Université de Versailles Saint-Quentin-en-Yvelines

Ph.D Dissertation

A thesis submitted to the graduate faculty in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

by

Ivan BEDINI

Thesis Title:

Deriving ontologies automatically from XML Schemas applied to the B2B domain

PhD Co-Advisors:

Georges GARDARIN	Université de Versailles	Thesis Director
Benjamin NGUYEN	Université de Versailles	Co-advisor
Thierry BOURON	France Telecom	Co-advisor

Program of Study Committee:

Jérôme EUZENAT	INRIA Grenoble Rhône-Alpes	Rapporteur
Chantal REYNAUD	Université Paris-Sud	Rapporteur
Luigi LANCIERI	Université Lille 1	Examiner

January 15th 2010

To all my little big family!

Acknowledgments

There are countless people who have supported, directed, assisted and encouraged me in completing this PhD, and that I would like to thank. It would require several pages to express my deepest gratitude to all of them and, needless to say, my thankfulness extends much further than the simple words that follow.

Firstly I am profoundly grateful to Prof. Georges Gardarin for accepting to supervise my work and for always having the right suggestions during any discussion we had. I must say that to have Georges as advisor has been a real pleasure and a great honour for me.

I thank Orange Labs as a whole to have given me the opportunity - and have provided the funds - to carry out this PhD. My thanks are especially addressed to: Thierry Bouron who has accepted to follow me in this experience as scientific director; Thierry Lemoisson, my research unit manager, who has believed in my project; Olivier Bouillant, Jean-Pierre Daquin and Frédéric Delmond who, one after the other as research laboratory managers, have supported me; Bruno Choquet, Jerôme Vinesse, Vincent Louis, Franck Panaget and Fayçal Boujemaa who have accepted my research work in their respective research programs through the years.

I would like to thank the University of Versailles Saint-Quentin, and particularly Benjamin Nguyen, my co-advisor, who always came up with very good ideas and suggestions and who was very patient with me (particularly with my poor written English); Karine Zeitouni and Laurent Yeh for the enriching exchanges we had on occasions and Chantal Ducoin for her kindness and her constant, useful administrative assistance.

I would like also to express my gratitude to Chantal Reynaud and Jérôme Euzenat for serving on my Thesis committee.

A special thank to Fabrice Bourge, my colleague and friend, who accomplished several tasks; he helped me a lot at the beginning, in the middle of the work and during the final achievement.

I have a very special thought also for Luigi Lancieri and Alain Léger for their encouraging guidance, their kindness and their suggestions that permitted me to rightly begin the work.

I am grateful to the Servery European research project for the opportunity and the luck I had of taking part as a member of different working groups. Especially I would like to thank Stéphanie Fodor and Mathieu Boussard for their leadership.

I would like to thank Emmanuel Olivier and Yvan Coquelin for all the discussions we had and for their concrete contribution to the implementation phase.

I would like to extend my thoughts to Pierre Bregant-Belin, Tiphaine Marie, Francis Berthomieu, Fabrice Jeanne, Sebastien Picant, Patrick Grohan, Emmanuel Bertin, Nassim Laga, Jacques Madelaine for all the enriching discussions we had and for having "lent me their ears" at some point of my work.

I would like to thank Michelle Harel for her last minute useful help during my final fight with uncooperative printers.

Last but not least my thanks go to my family of course, to whom I dedicate this achievement: my parents who instilled a thirst for excellence in me; Alice who, with love and comprehension, has always pushed me to pursue a PhD; my son Sasha who, even during a period of difficult health condition, has shown me the way of acceptance and serenity in such situations; my daughter Lisa, who was born at the beginning of the PhD, and has always brought a new smile to sleepless nights; my daughter Zélie who, like the last piece of a puzzle, is the element without which the picture is not perfect.

Abstract

Computer mediated networks play a central role in the evolution of Enterprise Information Systems. However the integration of data in networked systems still remains harder than it really should be. In our research we claim that Semantic Web technologies, and specifically ontologies, are well suited to integrate this domain to fulfil current approaches and achieve the needed flexibility. For this we address the first step toward the business semantic communication with a system that overcomes some of the existing lacks in the state of the art and provides a new approach for the automatic generation of ontologies from XML sources. We show the usefulness of our system by applying our theory to the B2B domain and producing automatically ontologies of relevant quality and expressiveness.

Résumé

La communication entre systèmes d'information d'entreprise joue un rôle central dans l'évolution des processus d'affaire. Pourtant l'intégration des données reste complexe : elle exige un effort humain considérable, surtout pour les connexions d'applications appartenant à différentes entreprises. Dans notre recherche nous affirmons que les technologies du Web Sémantique, et plus particulièrement les ontologies, peuvent permettre l'obtention de la flexibilité nécessaire. Pour cela le système que nous avons définit permet de surmonter certains manques dans l'état de l'art actuel et réalise une nouvelle approche pour la génération automatique d'ontologies à partir de sources XML. Nous montrons l'utilité du système en appliquant notre théorie au domaine du B2B pour produire automatiquement des ontologies de qualité et d’expressivité appropriée.

Contents

List of Figures

List of Tables

List of Listings

Introduction

Interesting applications rarely live alone. Whether the sales application must interface with the inventory application, or the inventory application must connect to the supplier's application, or the simple mobile calendar must synchronize with the professional calendar, applications require efficient and effortless integration with others. Passing to the scale of enterprises applications the integration still remains harder than it really should be. Enterprises are typically comprised of hundreds of applications that are custom built, acquired from third parties or a combination of both. Moreover it is not uncommon to find an enterprise with several Web sites, many instances of enterprise software, and countless departmental solutions. The integration of these application systems becomes a real challenge that requires a considerable human effort, especially if we aim at the connection of applications belonging to two different enterprises. This last use case refers to what is called Business to Business (simply B2B).

In the book *Enterprise Integration Patterns*, Gregor Hohpe [1] clearly formalizes problems we have with messaging-based application integration. He provides a very complete list of 65 patterns that aim at defining a common vocabulary used to build enterprise messages integration solutions. To highlight the main problems we have in application integration, we identify exchange patterns at three main interoperability levels: the communication channel, the message format, and the message content level. When analysing these three levels it appears that the communication channel, which assures the physical connection among applications, has evolved from point to point private networks to the World Wide Web communication layer. Thus every enterprise now has easy access to the communication channel, which less and less constitutes a barrier to the application integration development. Also the message format layer, which constitutes the message protocol adopted to exchange messages, seems to reach some stability. Even when an enterprise must handle different protocols and data formats, existing enterprise software systems are often capable of offering run time transformation or to adapt the application dynamically. This again does not constitute a main obstacle. What really constitutes the core issue is the integration of data at message content level. What information an application must handle and what is the meaning of the exchanged information are the two remaining core problems to be solved.

Throughout our thesis we investigate the adoption of Semantic Web-related technologies, as defined by Berners-Lee *et al.* [2] and by Motta *et al.* [3], to complement the current B2B approaches so as to allow a more dynamic set up and execution of electronic business exchanges. In fact B2B represents an interesting use case for Semantic Web technologies. The advent of XML along with Web Services, and more generically with the Service Oriented Architecture (SOA), certainly has contributed greatly to the development of standards-based integration solutions. However the large adoption of these technologies has also provoked a new fragmentation in applications development. As a result standardisation addresses only parts of the integration challenge. The frequent claim that XML is the lingua franca for system integration is somewhat misleading; indeed it does not imply common semantics and its adoption have led to the generation of countless dialects and languages which cannot be understood and integrated directly by machines. This problem is reflected in the many existing B2B standards that we present and analyse in this document.

In this context, we have positioned the core of our thesis on standards integration. We state that using this great number of proposed XML formats, although they are somewhat heterogeneous, it is possible to derive automatically a semantic common knowledge representation that: i) improves performances and capacity of automatic matching systems and; ii) can be used to generate a reusable knowledge to generate ontologies dynamically. Although it may seem somehow trivial at first, the issue interferes with several research areas that must be partly considered before focusing on the core topics. These research areas are:

- Enterprise Application Integration, e-business and B2B.
- Ontology Engineering: Ontology Learning, Ontology Matching, Ontology Alignment, Ontology Merging and building methodologies.

On the automation aspect we must also consider the following areas:

- Information retrieval, in particular Information Extraction, Text Mining, Concept Analysis, Clustering
- Semantic Web, Natural Language Processing and Knowledge Representation.
- Data Integration including Data Modelization and Schema Matching.

Motivation and Aim

The thesis applies the Semantic Web techniques to the B2B domain. Thus we first provide an overview of both domains. The state of the art and the consequent B2B-related analysis highlights the fact that mainly XML Schemas are used and maintained. Current research in the field of Semantic Web related to the more generic e-business domain focuses on product classification, such as works provided by Corcho *et al.* [4] and by Martin Hepp [5]. Although they provide a valuable work that contributes to the development of the enterprise integration domain, we show that they focus more on providing a taxonomy for the e-commerce through semantics catalogues of reference rather than building a reference knowledge for business messages definition. In this specific area existing B2B ontologies are still in a proof of concept phase, but as far as we know, no real business transaction solutions are integrated with the help of ontology concepts. In this thesis we first address the generation of semantics-based tools for the B2B domain. Then we provide a system that facilitates the human task of producing such knowledge. This could lead to the creation of a new generation of systems that produce semantically well formed business documents. As a consequence automatic systems aiming at direct data integration at run time could emerge and be more efficient. The lack of semantics attached to documents constitutes the first barrier to the realization of such systems.

In recent years the Semantic Web community has been very active and productive in this research field. One of its main purposes is to provide a meaningful representation of data over the web such that machines are capable of rightly interpreting data. In this research area a great amount of work is dedicated to improving ontology engineering. This includes techniques to discover correspondences and to match similar concepts automatically. With such tools it is simple to imagine some of the benefits enterprises could obtain. This led us to adopt these technologies and to try to contribute to fill existing gaps. We show that among observed approaches to the automatic ontology generation problem, those adopting a framework integrating an intermediary semantic model better automate ontology generation. Furthermore throughout the entire analysis, we observed that the extraction of ontological knowledge from XML sources is viable, as shown for example by the solution proposed by Giraldo and Reynaud [6]. But few systems provide advanced software to this purpose; this is an important lack to overcome.

We believe that focusing over the matching problem is probably the key research challenge to overcome the ontology generation process automation. As shown by the Ontology Alignment Evaluation Initiative ([7], [8], [9]), there are already a lot of notable ongoing works on this topic that seem acquiring interesting results. Matching systems can obtain real benefits from the adoption of an external resource and thus improve results and execution time performances (Aleksovski *et al.* [10], [11], Giunchiglia *et al.* [12]). However, as asserted by Euzenat and Shvaiko [13], few solutions still use this kind of knowledge. Yet we noticed that solutions adopting an external resource implicitly assume it exists in compatible format and semantics. But such external resource either is an upper ontology that often is inadequate for the application domain or is a domain specific formal ontology that is difficult to find, if it exists at all.

These observations motivated us to focus on the development of a specific semantic model capable of retaining relevant information that covers the matching need. This model is an essential prerequisite for matching and merging systems.

More precisely, the overall goal of our thesis has been to determine a solution facilitating the generation of domain-specific formal knowledge that can be used by matching systems. It is centred on an efficient algorithm that quickly discovers correspondences among entities and resolves alignment conflicts. Thus our system is capable of generating dynamic ontologies. This goal is reached by achieving the following objectives:

- extract conceptual knowledge from a large source corpus composed by XML Schemas;
- build a formal meta-model capable of managing the extracted information and of producing a common view of input sources;
- manage incremental addition of sources;
- generate an expressive ontology in standard language (OWL);

- implement fast, scalable and reliable algorithms.

To meet these objectives we have collected more than 3000 representative XML Schemas defining B2B messages. We have developed a generic system that is able to extract information from all these files. The extracted knowledge is formalized in a semantic model that we have used to provide specific information to matching systems. The outcome is a system capable of generating automatically a general semantic model that can be used to produce a first ontology skeleton in OWL format.

Main contributions

This thesis proposes mainly a semantic data model for ontology support, a methodology for extracting knowledge from XML schemas, and a system integrating the algorithms and the methodology to automatically generate ontologies. In particular we show how the conceptualised knowledge is completely obtained by an automate, and how it is used to improve the matching operation and to dynamically generate an ontology. The salient results of our work are:

- Validation of semantics and structures of incoming XML sources;
- Definition of an automatic ontology generation process;
- Definition of a specific intermediary Semantic Data Model;
- Information extraction from a large set of XML Schemas;
- Conceptualization of XML Schemas using our semantic data model;
- Generation of the Similarity Network as reference knowledge for matching/merging concepts;
- Dynamic generation of OWL ontology using instances of our model;
- Useful graphical interface and meaningful graphical representations of concepts and relationships.

Overall the main contributions of our research are: i) we provide an advanced information extraction software for XML Schema sources; ii) we improve performances of existing matching systems; iii) we increase the capacity of systems to automatically generate well defined semantic knowledge, thus lowering the human "bottleneck". The following paragraphs give some insight into these contributions.

We provide a new approach to the automation of ontology generation. This approach is based on an ontology generation life-cycle process that aims to delineate the main phases that bring to the generation of an ontology automatically. The life-cycle considers the possibility to add incrementally new sources that is fundamental to provide the necessary suppleness to an automatic approach.

We define a specific intermediary Semantic Data Model for Ontologies (SDMO) aiming at representing valid background knowledge for the automatic construction of ontologies and for semantic matching systems. We show that matching multiple sources is a different operation from operating over only two sources at once which is the usual approach. To address this issue we use instances of our model to maintain a "network of similarities" among concepts that is capable of providing the most appropriate one(s) in a generic context. Consequently the automatic ontology generation is a mapping of SDMO models to OWL.

An "XML miner" component has been developed to capture as many concepts and relationships as possible from XML Schema sources. Not only this is a specific component dedicated to our system, but it can also constitute a living stand alone module for other systems. In our specific implementation it aims at building instances of the model [14]. We show by practical experiences that our engine is well suited with respect to other existing solutions and is capable of getting relevant conceptual knowledge. Hence we have collected several XML Schemas from the B2B domain and obtained a corpus composed of more than 3000 files coming from 25 families (each family corresponding to a separate B2B standard body). We used this corpus in a preliminary phase to study general practices on XML Schema definitions and to validate the starting point of our approach. It turns out that XML Schemas sources provide a rich set of semantics content that can actually be used as input to build the ontology. We show that we are capable of providing a first level basic taxonomy (a sort of controlled vocabulary) from XML schemas [15].

We provide a first prototype that implements the greatest part of our theory. It brings together: the XML miner engine; the semantic data model; a procedure that queries the model to merge extracted concepts [16]; a graphical interface that permits a useful visualization of results; and the process to derive automatically an ontology ([17], [18]). Although the implementation remains a prototype, it has been sufficient to produce several tests demonstrating the soundness and power of our approach. As we will show, the system is able to produce and maintain instances of the model in an acceptable computation time. It is scalable enough to target a larger corpus than what we have been able to collect. Concerning quality results of our system, we have been able to define a small corpus of XML Schemas on which we measure expected precision and recall. It turns out that our approach is also viable in this aspect.

Finally, although it was initially targeted for the B2B domain, we have developed a generic component that can extract information from any XML Schema, regardless of its application domain. The only specific elements are the dictionary of abbreviations and a list of stop words. These pieces are external to the module and can be changed easily. Another aspect is the integration of an advanced graphical view of the generated set of concepts and relationships. This is not a completely new element, nevertheless it remains a real plus that facilitates the understanding of the semantic data model instances.

Thesis Outline

This document is divided in five chapters: The first one presents all background information regarding the Semantic Web, introduces the State of the Art concerning ontology construction and illustrates the main problems we address. The second chapter analyses the B2B architecture, its limitations and the aim of our research. The third chapter describes the semantic data model we have conceived to reach our goal. The fourth Chapter details the information extraction process from XML Schemas and proposes its conceptualization. The last part describes our implementations and the main results from our experiments. With more details, this thesis is organized as follows:

- **Chapter 1** provides an overview of the background information about ontologies and presents the main problem addressed by our thesis, the automatic generation of ontologies. Precisely :
    - Section 1.1 provides a short overview of the Semantic Web, its technologies, details the definition of ontology and depicts the Web Ontology Language (OWL).
    - Section 1.2 introduces the State of the Art concerning the automatic ontology generation and compares the proposals. Moreover it presents our approach to the automatic ontology generation problem as a multi-step process to follow to gather the final ontology according to a well-defined life-cycle.
    - Section 1.3 focuses over the matching problem and the associated algorithms.
    - Section 1.4 concludes this Chapter providing the main directions followed in our Thesis.
- **Chapter 2** does a little step behind to present the B2B domain. It introduces the B2B domain, mainly focusing on its weaknesses and problems. In details:
    - Section 0 presents the B2B domain, the components of a typical architecture, an analysis of the most common approaches. Then, it introduces the B2B standards which constitute the corpus from which we extract semantics to produce ontological knowledge.
    - Section 2.2 undertakes the question of why to use ontologies in the domain and tries to provide elements for the answer. Moreover, we outline the requirements of B2B ontologies.
    - Section 2.3 surveys existing B2B ontologies.
    - Section 2.4 concludes this Chapter and leaves the hand to our system.
- **Chapter 3** presents SDMO, our semantic model defined to maintain the collected information from the extraction phase.
    - Section 3.1 defines in detail our model with informal and formal descriptions.
    - Section 3.2 traces our direction to provide an ontology starting from the defined model. For this we specify the mapping from SDMO to OWL.
    - Section 3.3 depicts related works, different models that already have been defined in the domain of ontology construction. We also provide the evaluation and benefits of our approach.
    - Section 3.4 concludes this Chapter and highlights the main advantages of our model.
- **Chapter 4** aims at producing conceptual knowledge from non conceptual one. We present our analysis over the given XML corpus and validate our starting hypothesis that XML Schemas well fit the minimal exigency to have good quality input source. Furthermore we provide all details about the conceptualization operation and a theoretical evaluation of our

system with respect to others, showing that our system performs well the information extraction.

- Section 4.1 introduces the XML Schema standard to provide a basic knowledge to the reader. It has not the ambition to explain the whole XML model but to inform the reader on the basics. For this the W3C provides a more complete and rich collection of documents.
- Section 4.2 goes further into XML B2B standards and provides some interesting figures about usage and practices of XML Schemas in this domain, on semantics as well as XML structures.
- Section 4.3 already tries to produce a first B2B taxonomy starting from simple semantics extracted. We show how a B2B vocabulary arises naturally by the integration of the different standards; however, it is not enough to produce an ontology.
- Section 4.4 provides a conceptualization of XML Schema sources using our model. Moreover, it suggests a basic theoretical evaluation of our approach with respect to others.
- Section 4.5 gives some starting elements for the evaluation of an input source to decide using it or discarding it. This is because including bad information decreases the quality of the final result.
- Section 4.6 is a conclusion.

- **Chapter 5** presents Janus, our implementation
  - Section 5.1 presents Janus, the final tool performing knowledge extraction, ontology generation and visualization. It implements our model and follows the proposed life-cycle process. Moreover, the chapter depicts some few issues we have to resolve before implementing and the choices we made.
  - Section 5.2 goes beyond into some implementation details and provides the main algorithms for the construction of the "Similarity Network". Moreover, it details the frequency measure used.
  - Section 5.3 illustrates some integration problems and details the adopted solution. It also describes the integration algorithm that uses SDMO to unveil concepts similarities.
  - Section 5.4 shows a part of the tests we have performed and discusses the main results.
  - Section 5.5 concludes this chapter and provides an overall analysis of obtained results.
- Finally we summarize the thesis, discuss its contributions and provide a discussion of the approach and future directions for this work.
- **Appendix A** details the mapping we propose to OWL starting from SDMO.

Chapter 1.

Automatic Ontology Generation Problem

The current trend of application management is related to dynamic changes, system flexibility, and execution time performances, which implies that a considerable quantity of parameters change more and more quickly. As a consequence, current adopted knowledge representation, like UML [19] or XML Schemas [20], and human-based approaches to knowledge engineering show natural limits and need more advanced solutions. So if the first step is to adopt a more expressive language, like that one offered by ontologies that might improve the machine interpretation capacity, the second is to provide also more automate systems to leverage the human "bottleneck".

Throughout this Chapter we analyse existing systems related to ontology generation automation. We have investigated most of existing solutions aiming the automatic ontology generation and the overall approach we have followed during this overview is try to answer to the following questions:

- Is there already an existing system that can automatically construct ontologies from large amount of data sources?
- If an existing system does not exist, how can we use parts of existing systems in order to propose a system that achieves this goal?
- Are there any extra parts that need to be developed?

To conduct this analysis we split existing systems following their overall approach that we have categorized in four main types: direct transformation, external resource integration, intermediary model integration and framework approach. The evaluation of the systems is based on a process for automatic ontology construction that we propose. For that we use the described steps of the process as discriminating element. Throughout the overview we show that few systems really focus the global problem of the automatic ontology generation. Even less propose information extraction from semi-structured knowledge like XML that, as we will see in Chapter 2, is our first requirement. Another rising problem is the fact that ontology generation is almost limited to one or two input sources at once. Nevertheless all these experiences constitute a very interesting and helpful information that will help us to understand current problems and best approaches to follow.

Before starting with the presentation of existing systems for the automatic generation of ontology, we precede this Chapter with a short overview of the Semantic Web and its main relevant technology with respect to our research. And we finish with a focus over the matching problem. In detail this Chapter is outlined as follows: in Section 1.1 we provide a presentation of Description Logic and ontologies which are the "common thread" of our work. For this, after some basic definitions, we briefly introduce RDF/S and OWL W3C standards that seem reaching wide usage and success as ontology formalisation format. Next, in Section 1.2 we provide our overall approach to the automation of the ontology generation, the state of the art of current automation systems and the analysis of the visited solutions. After, in Section 1.3, we focus on the matching process, which is one of the most relevant parts that we retain important to further improve. Finally we summarize this Chapter in the conclusion section.

1.1 Ontology Representation

1.1.1 Semantic Web

The Semantic Web [2] is an extension of the current Web in which information is given with well-defined meaning, better enabling computers and people to work in cooperation. This is realized by marking up Web contents with properties, and relations, in a reasonably expressive markup language with a well-defined semantics.

In such a context, some languages also known as Semantic Web languages are used to represent information about resources on the Web. This information is not limited to Web resource description, but can be about anything that can be identified. Uniform Resource Identifiers (URIs) are used to uniquely identify entities. For example, it is possible to assign a URI to a person, to the company she works for, to the car she owns. Therefore relations between these entities can be written and shared on the Semantic Web in unambiguous way. A stack of languages has been published as W3C recommendations to be used on the Semantic Web. We summarize these languages and their goals in the following paragraphs.

1.1.2 Definition of Ontology

There have been many attempts to define what constitutes an ontology [21], [22], [23], [24], [25], [26] but perhaps the best known (in computer science) is due to Gruber [27] [28]:

An ontology is an explicit specification of a conceptualization.

In this context, a **conceptualization** means an abstract model of some aspect of the world, taking the form of a definition of the properties of important concepts and relationships. An **explicit specification** means that the model should be specified in some unambiguous language, making it amenable to processing by machines as well as by humans.

From this broad definition, Borst [29] and Fensel [30] emphasize the fact that there must be **agreement** on the conceptualization that is specified. The reason for including this is that the ability to reuse an ontology will be almost null when the conceptualization it specifies is not generally accepted; this requires adding that the conceptualization should be shared. Furthermore, Guarino [31] suggests the opportunity to develop different kinds of ontology according to their level of generality, as shown in Figure 1.1 (see [32] for a more detailed discussion).

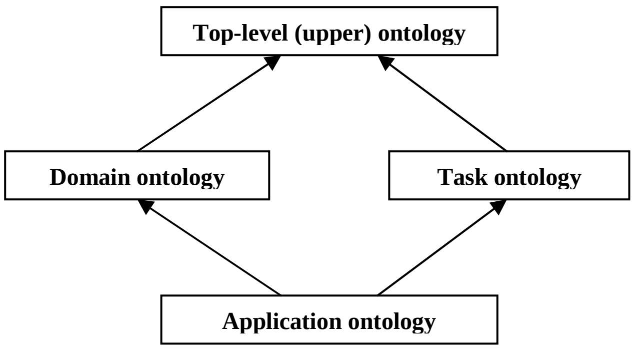


Figure 1.1 – Kinds of ontologies, according to their level of dependence on a particular task or point of view (thick arrows represent specialization relationships).

Figure 1.1 distinguishes three levels of ontologies as follows :

- **Top-level ontologies** describe very general concepts like space, time, matter, object, event, action, etc., which are independent of a particular problem or domain; it seems therefore reasonable, at least in theory, to have unified top-level ontologies for large communities of users.
- **Domain ontologies** and **task ontologies** describe, respectively, the vocabulary related to a generic domain (like medicine, or automobiles) or a generic task or activity (like diagnosing or selling), by specializing the terms introduced in the top-level ontology.
- **Application ontologies** describe concepts depending both on a particular domain and task, which are often specializations of both the related ontologies. These concepts often correspond to roles played by domain entities while performing a certain activity, like replaceable unit or spare component.

Thus, Ontologies glue together three important requirements to consider when developing a conceptual model: (i) they aim at consensual knowledge, their development require a cooperative process, and they should deal with pragmatics reasons (e.g., limiting complexity and dimension). (ii) They formalize semantics for information, consequently allowing information processing by a computer. (iii) And finally, formal ontologies implicitly use real-world semantics, which makes it possible to link machine processable content with meaning for humans.

There are several languages on which ontology can be expressed, but most of them share many structural similarities and kinds of entities. Below these common components are introduced with simple examples in turn:

- **Classes** or **concepts** are the top entities, corresponding to types of real world objects (e.g. *Person* or *Motorbike*)
- **Individuals** which are instances of classes, also called objects, are the basic or "ground level" objects (like *MotoGuzziV7* is an instance of the class *Motorbike*).
- **Relations** are ways in which classes and individuals can be related to one another, like Mark *is child of* Helen.
- **Datatypes** specify the kind of values on which an object is expressed; they can be simple value (like *string* or *integer*) or composed ones (as an address).
- **Attributes** which are aspects, features or parameters that objects (and classes) can have.
- **Restrictions** formally stated descriptions of what must be true in order for some assertion to be accepted as input (e.g. *All Persons having at least 2 children).*
- **Axioms** which are assertions in a logical form that together comprise the overall theory that the ontology describes in its domain of application.

(e.g. $Offer \equiv \forall priceOffer.Price \sqcap \forall interfacedBy.Service$)

Formally an ontology o is at least a tuple $o = (\mathcal{C}, \mathcal{R}, \mathcal{I}, \mathcal{D}, \subseteq)$ such that:

- $\mathcal{C}$ is the set of classes or concepts;
- $\mathcal{R}$ is a set of relations;
- $\mathcal{I}$ is the set of classes' instances (also called individuals);
- $\mathcal{D}$ is the set of Data Types;
- $\subseteq$ is a binary relation over entities belonging to $\mathcal{C}$, $\mathcal{R}$ and $\mathcal{D}$, called *specialisation*;

This definition does not include restrictions and axioms, except for generalization. It can be extended with other specific relationships and with constraints between classes and between instances, depending on the expressivity of the formalization language.

Now, if it is humanly relatively simple to represent and understand an ontology, to provide a machine processable language capable of undertake reasoning features over such a knowledge representation remains difficult. For this reason, several ontology definition languages exist, but we have focused our attention over ontology formalization based on Description Logics and their formalization following the W3C standards. These logics were created from the attempts to formalize semantic networks and frame based systems. They provide powerful formal description of concepts and roles (relations). Semantically they are founded on predicate logic, but their expression power is limited to be enough for practical modelling purposes and to have good computational properties such as decidability. This framework thus offers the basis that enables certain kinds of automated reasoning with formal ontologies. This is one of the best advantages offered by Description Logic based ontologies in respect with others knowledge representations.

1.1.3 Description Logic

It is acknowledged that Description Logics have heavily influenced the development of Semantic Web languages. For example, RDF-S can even be described as a relatively inexpressive Description Logic while OWL (both RDF-S and OWL are presented below) is in fact an alternative syntax for a very expressive Description Logic.
Description Logics (henceforth DL) [33] are a family of knowledge representation languages which can be used to represent the concept definitions in a structured and formally well-understood way. Knowledge representation systems based on DLs are drawn using the so-called **TBox** (terminological box) and the **ABox** (assertional box). The TBox describes terminology, i.e., the ontology in the form of concepts and roles definitions (i.e., relations between concepts), while the ABox contains assertions about individuals using the terms from the ontology. Concepts describe sets of individuals, roles describe relations between individuals. For example, the statement "*Every employee is a person*" belongs in the TBox, while "*Bob is an employee*" belongs in the ABox.

There are many varieties of Description Logics and there is an informal naming convention, roughly describing the operators allowed. In Table 1.1 are listed some labels for a logic expressivity.

$\mathcal{F}$	Functional properties
$\mathcal{E}$	Full existential qualification
$\mathcal{U}$	Concept union
$\mathcal{C}$	Complex concept negation (allows negation of concepts that are not atomic)
$\mathcal{S}$	An abbreviation for $\mathcal{ALC}$ with transitive roles. Where $\mathcal{AL}$ Attributive language
$\mathcal{H}$	Role hierarchy (subproperties)
$\mathcal{R}$	Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness
$\mathcal{O}$	Nominals. (Enumerated classes of object value restrictions)
$\mathcal{I}$	Inverse properties
$\mathcal{N}$	Cardinality restrictions
$\mathcal{Q}$	Qualified cardinality restrictions
$(\mathcal{D})$	Use of datatype properties, data values or data types

Table 1.1 – DL operators and naming conventions

Before introducing DLs constructors, we recall some main notational conventions as adopted in [33]. The letters *A*, *B* will often be used for atomic concepts, and *C*, *D* for concept descriptions. For roles the letters *R*, *S* are used, and for functional roles (features, attributes) the letters *f*, *g*. Nonnegative integers (in number restrictions) are often denoted by *n*, *m*, and individuals by *a*, *b*. These conventions are followed when defining syntax and semantics and in abstract examples. In concrete examples, the following conventions are preferred: concept names start with an uppercase letter followed by lowercase letters (e.g., Human, Male), role names (also functional ones) start with a lowercase letter (e.g., *hasChild, marriedTo*), and individual names are all uppercase (e.g., CHARLES, MARY).

In Table 1.2, the two first columns illustrate the DLs constructors as well as their syntaxes. The third column illustrates their semantics. The various description logics differ from one to another based on the set of constructors they allow, as shown in the fourth column.

Elementary descriptions are *atomic concepts* and *atomic roles* (also called *concept names* and *role names*). Let N_C be the set of concept names and N_R the set of roles. These are defined only by the word that is their concept name. And N_A is the set of atomic concepts (thus $N_A \subseteq N_C$). Complex descriptions can be built from them inductively with *concept constructors* and *role constructors*.

Name	Syntax	Semantics	Symbol
Atomic concept	A	$A^I \subseteq \Delta^I$	$\mathcal{AL}$
Top (universal concept)	$\top$	Δ^I	$\mathcal{AL}$
Bottom (bottom concept)	$\perp$	$\varnothing$	$\mathcal{AL}$
Intersection	$C \sqcap D$	$C^I \cap D^I$	$\mathcal{AL}$
Union	$C \sqcup D$	$C^I \cup D^I$	$\mathcal{U}$
Negation	$\neg C$	Δ^I n C^I	$\mathcal{C}$
Value restriction	$\forall R.C$	$\{a \in \Delta^I \mid \forall b. (a, b) \in R^I \rightarrow b \in C^I\}$	$\mathcal{AL}$
Existential quant.	$\exists R.C$	$\{a \in \Delta^I \mid \exists b. (a, b) \in R^I \wedge b \in C^I\}$	$\mathcal{E}$
Unqualified number restriction	$\geqslant nR$ $\leqslant nR$ $=nR$	$\{a \in \Delta^I \mid \|\{b \in \Delta^I \mid (a, b) \in R^I\}\| \geq n\}$ $\{a \in \Delta^I \mid \|\{b \in \Delta^I \mid (a, b) \in R^I\}\| \leq n\}$ $\{a \in \Delta^I \mid \|\{b \in \Delta^I \mid (a, b) \in R^I\}\| = n\}$	$\mathcal{N}$
Qualified number restriction	$\geqslant nR.C$ $\leqslant nR.C$ $=nR.C$	$\{a \in \Delta^I \mid \|\{b \in \Delta^I \mid (a, b) \in R^I \wedge b \in C^I\}\| \geq n\}$ $\{a \in \Delta^I \mid \|\{b \in \Delta^I \mid (a, b) \in R^I \wedge b \in C^I\}\| \leq n\}$ $\{a \in \Delta^I \mid \|\{b \in \Delta^I \mid (a, b) \in R^I \wedge b \in C^I\}\| = n\}$	$\mathcal{Q}$
Role-value map	$R \subseteq S$ $R = S$	$\{a \in \Delta^I \mid \{\forall b. (a, b) \in R^I \rightarrow (a, b) \in S^I\}$ $\{a \in \Delta^I \mid \{\forall b. (a, b) \in R^I \leftrightarrow (a, b) \in S^I\}$	
Agreement and disagreement	$u_1 = u_2$ $u_1 \neq u_2$	$\{a \in \Delta^I \mid \exists b \in \Delta^I . u_1^I(a) = b = u_2^I(a)\}$ $\{a \in \Delta^I \mid \exists b_1, b_2 \in \Delta^I . u_1^I(a) = b_1 \neq b_2 = u_2^I(a)\}$	$\mathcal{F}$
Nominal	I	$I^I \subseteq \Delta^I$ with $\|I^I\| = 1$	$\mathcal{O}$

Table 1.2 – Some Description Logic concept constructors.

The semantics of a concept description (third column of Table 1.2) is defined in terms of an **interpretation** $I = (\Delta^I, \Delta)$, which consists of a nonempty set Δ^I, the domain of the interpretation, and an interpretation function, which associates to each concept name $A \in N_C$ a subset $A^I \subseteq \Delta^I$ and to each role name $R \in N_R$ a binary relation $R^I \subseteq \Delta^I \times \Delta^I$. Additionally, the extension of Δ^I to arbitrary concept descriptions is defined inductively as shown in the third column of Table 1.2.

For example given a set of delivered invoices, an interpretation of such set could be the subset of invoices paid by Acme Inc.

More in detail we define **terminological axioms** as the first component of a DL based knowledge base K, which in the most general case have the form $C \sqsubseteq D$ (resp. $R \sqsubseteq S$) called *inclusions*, or $C \equiv D$ (resp. $R \equiv S$) called *equalities*.

In DL an equality whose left-hand side is an atomic concept is a *definition*. Definitions are used to introduce **symbolic names** for complex descriptions. For instance, let us simply assume that a Supplier is itself a company having another company as customer; in this case Supplier and Customer are the symbolic names for the following axioms:

$$\text{Supplier} \equiv \text{Company} \sqcap \text{hasCustomer.Company}$$

$$\text{Customer} \equiv \text{Company} \sqcap \text{hasSupplier.Company}$$

A symbolic name can also be used as abbreviation in other descriptions, such as:

$$\text{BusinessPartner} \equiv \text{Customer} \sqcup \text{Supplier}$$

So, if no symbolic name is defined more than once, a *terminology* T (also **TBox**) is the finite set of such definitions. That means that for each atomic concept A there is at most one axiom in T whose left-hand side is A. Normally a concept appearing only in the right-hand side of a set T is also referred as a *primitive* concept.

1.1.4 Inferences with Ontologies

A knowledge representation based on DLs is able to perform specific kind of reasoning. This means that given a knowledge base, denoted as a pair $K = \langle T , A \rangle$, where as already mentioned above, T are TBox while A, the second component is the so called *world description* or **ABox**. Finally K contains implicit knowledge that can be made explicit through inferences.

Standard inferences can be done with ontology representations based on DLs. Based on DL semantics and the terminological knowledge T of a knowledge base K, basic DL inferences on T are the following: satisfiability, subsumption, equivalence and disjointness [33] on T.

Definition (Satisfiability, Subsumption, Equivalence and Disjointness)

- **Satisfiability**. A concept C is satisfiable with respect to T if there exists an interpretation I of T such that C^I is nonempty. In this case we say also that I is an interpretation of C.
- **Subsumption**. A concept C is subsumed by a concept D with respect to T iff $C^I \subseteq D^I$ for every interpretation I of T. In this case we write $T \models C \sqsubseteq D$ (we also say that C *specializes* D).

- **Equivalence**. Two concepts C and D are equivalent with respect to T iff $C^I = D^I$ for every interpretation I of T. In this case we write $T \models C \equiv D$.
- **Disjointness**. Two concepts C and D are disjoint with respect to T iff $C^I \cap D^I = \varnothing$ for every interpretation I of T.

Such basic inferences are required not only to maintain and to guarantee consistency of DL knowledge bases but also to classify them. For instance, the TBox classification aims at placing a new concept in the suitable place in a taxonomic hierarchy according to the partial order induced by subsumption relationships among the other defined concepts.

1.1.5 RDF and RDF-S

The W3C recommendation Resource Description Framework (RDF) [34] is a first level of knowledge representation formalism. Basically speaking, the RDF data model is based upon the idea of making statements about resources, in particular, Web resources, in the form of subject-predicate-object expressions. These expressions are known as triples in RDF terminology. Triples are statements that contain a subject, a predicate, and an object. RDF can be viewed as an application neutral data model. RDF representations are depicted as directed labelled graph, as illustrated in Figure 1.2.

The subject of an RDF statement is either a Uniform Resource Identifier (URI) or a blank node, both of which denote resource. Resources indicated by blank nodes are called anonymous resources. They are not directly identifiable from the RDF statement. The predicate is a URI which also indicates a resource, representing a relationship. The object is a URI, blank node or a Unicode string literal.

In a triple a resource, the **subject**, is linked to another resource, the **object**, through an arc labelled with a property. The triple is also called a **statement**. Notice that the object can be a value or a resource, which can have in turn properties/attributes.

- *Statement = <object, subject, predicate>*
- *Statement: <The supplier, of http://www.LDLC.com/printers/#EPCCode, is HP>*

That read in a more human form becomes: *HP is the supplier of the printer #EPCCode*.

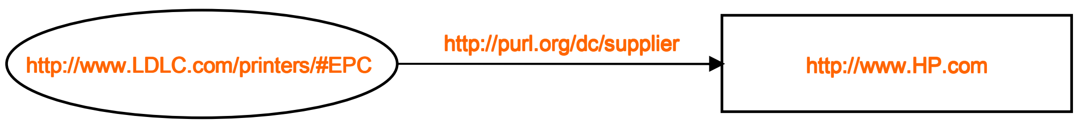

http://www.LDLC.com/printers/#EPC
http://purl.org/dc/supplier
http://www.HP.com

Figure 1.2 – Example RDF statement graphical representation

That in XML formalization becomes:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:si="http://www.LDLC.com/siteinfo#">
  <rdf:Description rdf:about=" http://www.LDLC.com/printers/#EPC">
    <si:supplier>http://www.HP.com</si:supplier>
```

```
7:   </rdf:Description>
9: </rdf:RDF>
```

Listing 1.1 – RDF /XML document example

RDF Schema (RDF-S) [35] is a collection of RDF resources that can be used to describe properties of other RDF resources. Unlike its name suggests, RDF-S is not a schema that imposes specific constraints on the structure of a document, but instead it provides information about the interpretation of the statements given in an RDF data model. In this regard, RDF-S has similarities to frame based languages. Finally, following their original scope, RDF and RDFS are languages for describing the organization of resources on the Web.

1.1.6 OWL - the Web Ontology Language

The Web Ontology Language (OWL) [36], [37] is one of the most expressive standardized Semantic Web languages. It is layered on top of RDF and RDF-S. OWL is a family of knowledge representation languages based on DLs. OWL languages are well-founded, useful and efficient enough for being the basis of knowledge representation for the Semantic Web, and thus for representing ontologies. OWL can be used to define classes (unary relations) and properties (binary relations) as in RDF-S but also provides constructs to create new class descriptions as logical combinations (intersections, unions, or complements) of other classes, define cardinality restrictions on properties and so on. OWL has three different levels of expressiveness: OWL-Lite, OWL-DL and OWL-Full. Each of these sublanguages is a syntactic extension of its simpler predecessor. OWL-Lite and OWL-DL differ from OWL-Full in such a way that they define certain constraints on RDF and RDF-S to be compatible with the traditional semantics of Description Logics. Reason for this differentiation is to look for in the decidability and computational complexity of the underlying DL w.r.t. reasoning techniques.

Constructor	**DL Syntax**	**Example**
`intersectionOf`	$C_1 \sqcap \ldots \sqcap C_2$	BusinessPartner ⊓ Customer
`unionOf`	$C_1 \sqcup \ldots \sqcup C_2$	Customer ⊔ Supplier
`complementOf`	¬C	¬Customer
`one of`	$\{x_1\} \sqcup \ldots \sqcup \{x_2\}$	{Orange} ⊔ {Telefonica}
`allValuesFrom`	∀P.C	∀hasCustomer.Manufacturer
`someValuesFrom`	∃P.C	∃hasSupplier.Commerce
`maxCardinality`	nP	≼1hasCustomer
`minCardinality`	≽nP	≽2hasSupplier

Table 1.3 – Some OWL Class constructors and relative DL syntax

x With respect to Description Logic in OWL jargon a *class* is referred to as a *concept* in Description Logic, while a *property* is a *role* in Description Logic. Some of the constructors supported by OWL, along with the equivalent Description Logic syntax, are summarised in Table 1.3.

An OWL ontology consists of a set of axioms based on constructors. Table 1.4 summarises axioms (DL descriptions) supported by OWL. These axioms make it possible to assert subsumptions or equivalence with respect to classes or properties, the disjointness of classes, and the equivalence or non-equivalence of individuals (resources).

Axiom	**DL Syntax**	**Example**
`subClassOf` (concept inclusion)	$C \sqsubseteq D$	Supplier ⊑ BusinessPartner ⊓ Customer
`equivalentClass` (concept equivalence)	$C \equiv D$	Man ≡ Human ⊓ Male
`disjointWith`	$C1 \sqsubseteq \neg C2$	Male ⊑ ¬Female
`sameAs`	$\{x\} \equiv \{y\}$	{OrangeLabs} ≡ {FTR&D}
`differentFrom`	$\{x\} \sqsubseteq \neg\{y\}$	{FranceTelecom}⊑ ¬{FinancialTime}
`subPropertyOf` (role inclusion)	$R \sqsubseteq S$	hasSupplier ⊑ hasBusinessPartner
`equivalentProperty` (role equivalence)	$R \equiv S$	cost ≡ price
`inverseOf` (role transitivity)	$R \equiv S^{-}$	hasCustomer ≡ hasSupplier^{-}
`transitiveProperty`	$P+ \sqsubseteq P$	ancestor+ ⊑ ancestor
`functionalProperty`	$\top \sqsubseteq \leq 1P$	$\top \sqsubseteq \leq$ 1hasEmployer
`inverseFunctionalProperty`	$\top \sqsubseteq \leq 1P^{-}$	$\top \sqsubseteq \leq$ 1hasEmployee^{-}
`concept instantiation`	$c \in D$	{FranceTelecom} ∈ TelecomOperator
`role instantiation`	$\langle a,b\rangle \in R$	

Table 1.4 – Some OWL axioms and relative DL syntax

We provide in Listing 1.2 a simple example using OWL XML syntax of the declaration of the following assertion:

$$\text{Company} \sqcap \forall \text{hasSupplier}.(\text{Manufacturer} \sqcup \exists \text{hasSupplier}.\text{Manufacturer})$$

i.e., the set of companies which all suppliers are either manufacturer or have themselves a supplier which is a manufacturer.

```
<owl:Class>
  <owl:intersectionOf rdf:parseType=" collection">
    <owl:Class rdf:about="#Company"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasSupplier"/>
      <owl:toClass>
        <owl:unionOf rdf:parseType="collection">
          <owl:Class rdf:about="#Manufacturer"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasSupplier"/>
            <owl:hasClass rdf:resource="#Manufacturer"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:toClass>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Listing 1.2 – OWL XML syntax example

According to the DL naming convention presented in Table 1.1, and except for individuals and datatypes, the constructors and axioms of OWL can be translated into $\mathcal{SHIQ}$. In fact, OWL Lite is equivalent to $\mathcal{SHIN(D)}$ and OWL DL is equivalent to $\mathcal{SHOIN(D)}$ Description Logic. The ability to use DL reasoners to provide reasoning services for OWL applications was one of the motivations for basing the design of OWL on a DL. Several ontology design tools, both "academic" and commercial, now exploit the correspondence between OWL and $\mathcal{SHOIN(D)}$ in order to support ontology design and maintenance by, for example, highlighting inconsistent classes and implicit subsumptions relationships.

1.1.7 Synthesis

Throughout this section, we have introduced the meaning of an ontology as knowledge representation and RDF/OWL, one of the most powerful formalisation language to define ontology based on Description Logic. There exists several languages for ontology formalization, but in our system, we decided to adopt OWL. This language, originally designed to model Web resources, provides a reasoning system that can be used to automatically detect models inconsistencies and inferences. As we will show in the following Chapters, an OWL ontology can be also useful to formalize business exchange messages to improve enterprise application systems interoperability, like what is done in [38]. However a more expressive language also means more complex design task. For this reason, in the following sections, we investigate and present possible way to automate at least a part of the ontology generation, in order to leverage as much as possible human involvement.

1.2 Automatic Ontology Generation Overview

After a brief introduction to ontology and related language formalisms, we now analyse some approaches to the automatic ontology generation process. The methodologies proposed in the literature focus on different aspects of working with ontologies. For example, some approaches propose a general schema to be followed when constructing ontologies, some have an emphasis on the cooperative ontology construction by a group of knowledge engineers. In any case, it appears that ontology generation processes are human-centric, such as OTK [39], METHONTOLOGY [40], DILIGENT [41] or Neon Methodology [42] which target ontology engineers and not machines. Thus most approaches to ontology generation are mainly hand-made by domain experts, but as explained in the beginning of this chapter, hand-made methods are not our concern.

Our interest is in the automation of the ontology construction process. It is motivated by the fact that an ontology brings out a rich knowledge representation that in most cases can be difficult to build and maintain manually, above all if we consider distributed environment where base knowledge can change over the time. In these contexts it is simpler for someone in charge to build and maintain a domain ontology to be assisted by tools that can at least produce automatically a skeleton of ontology, or integrate new information on the fly, leaving at most a final refinement and validation. A more ambitious goal should state that ontologies could be defined by retrieving information sparse over the Web, but as we will see throughout the following overview, although there are some fully automatic systems, they still work under limited circumstances and have low performance, that still highly constraints the possibility to have generic automatic ontology generators. Some of these constraints are due to the lack of a formal reference knowledge model inherent with the domain of interest or of well defined source corpora from which it is possible to apply simple transformation rules.

1.2.1 Existing *State of the Art*

The literature offers several *State of the Art* on ontology learning and more specifically on ontology matching that focus on techniques and tools evaluations. Among them, we can cite the paper from Mehrnoush and Abdollahzadeh [43] which proposes a complete framework for classifying and comparing ontology learning systems. The authors propose six main categories (called *dimensions*) as follows: **elements learned** (concepts, relations, axioms, rules, instances, syntactic categories and thematic roles); **starting point** (prior knowledge and the type and language of input), **pre-processing** (linguistic processing such as deep understanding or shallow text processing); **learning methods** including also an evaluation about the degree of automation (manual, semi-automatic, cooperative, full automatic); the **result** (ontology vs. intermediate structures and in the first case the features of the built ontology such as coverage degree, usage or purpose, content type, structure and topology and representation language); and finally **evaluation methods** (evaluating the learning methods or evaluating the resulted ontology).

We share the most part of the conclusion of their analysis, especially regarding the importance of input sources, which of course are essential to the automation process and highly influence the result of the final learned ontology. In fact ontology learning systems extract their knowledge of interest from inputs, which can differ by type and language (e.g., English, German or French). Types can be **structured data** like already existing ontologies, some schemata or lexical semantic nets such as WordNet. Other sources for ontology learning systems are **semi-structured data** such as dictionaries, HTML and XML schemas and DTDs (document type definitions), which probably constitutes in the Web environment the most hot topic today. Finally, the most difficult type of input from which to extract ontological knowledge is the **unstructured** ones (e.g., free text). Tools that learn ontologies from natural language exploit the interacting constraints on the various language levels (from morphology to pragmatics and background knowledge) in order to discover new concepts and stipulate relationships between concepts [44]. Finally the authors of [43] assert that the first two kinds of input data are more appropriate to build ontologies for the Semantic Web, thus with DL implications, while the latter is more adapted to build more general lexicons such as taxonomies or dictionaries.

They also identify some open problems to be considered to improve the field, in particular: (i) the way to **evaluate** ontology learning systems, currently evaluated only on the basis of their final results; no measure is defined for specific parts of the learning process proving the accuracy, efficiency, and completeness of the built ontology. (ii) **Full automation** of ontology learning process is not described yet and integrating successful **modules** to build complete autonomous systems may eliminate their weaknesses and intensify their strengths. (iii) At last, moving toward **flexible** neutral ontology learning method may eliminate the need for reconstruction of the learning system for new environments.

Moving forward the automation process to enter in more technical surveys, in [45] authors provide a comprehensive tutorial and an overview on learning ontology from text. Rahm *et al.* [46] present an overview on techniques used for the schema matching automation. Euzenat *et al.* in [47] provide a detailed overview and classifications of techniques used for ontology alignment and a *state of the art* on existing systems for ontology matching/alignment, probably the best known software at present. From the book *Ontology Matching* by Euzenat and Shvaiko [13], which surely represents the most complete work in the current literature around the matching theme, beyond techniques are presented theoretical aspects and definitions involved into the matching process as well as their evaluation measures. As last, let us cite the survey presented by Castano *et al.* [48], which provides a comprehensive and easily understandable classification of techniques and different views of existing tools for ontology matching and coordination.

All these works provide a real detailed overview on ontology generation tools and aspects of possible automation, at least for some specific tasks. Indeed, even if the frontier between matching and generation tools is not always clearly definable, we can say that except the first one, all referred papers mainly focus on the matching step but do not cover the whole ontology automation process. We can also add that the matching problem is probably the most challenging part and this is the reason why we analyse it more deeply in Section 1.3 below. The overview proposed below focuses on different approaches of the process adopted for the automation, to provide full automation standpoint for ontology generation process and to highlight successful modules to build, in order to have complete autonomous systems integrating them.

1.2.2 Automatic Ontology Generation Life-Cycle

Automated generation provides a fundamentally different approach to ontology creation than manual construction by a designer. As we will see the majority of papers in this area propose methods to extend an existing ontology with new concepts, using natural language processing, statistical, and machine learning techniques. In the last few years most work has been developed under the names of *Ontology Mapping and Alignment, Ontology Merging* and *Ontology Integration* [49] (see also Section 1.3 for more details about the difference between these terms). Some results can be considered for our goal. For instance the PROMPT [50] and ANCHOR-PROMPT [51] systems were originally designed for assisting knowledge engineers in the process of merging and aligning ontologies. The system provides different heuristics for suggesting mappings to the users and identifying the concepts and roles to be merged. The FCA-Merge [52] method for ontology merging is based on Formal Concept Analysis techniques. The approach taken by the authors is “extensional”, in the sense that it is based on objects/individuals which appear in both ontologies to merge. Concepts having the same individuals are then supposed to be merged. The generation of the merged ontology from the concept lattice is semi-automatic and requires human interaction. The GLUE [53] system uses machine learning techniques for discovering mappings. Given two ontologies to be merged, for each concept in one ontology GLUE finds the most similar concept in the other ontology. GLUE exploits the information stored in both the TBox and the data. H-Match [54] is an automated ontology matching system that has been designed to enable knowledge discovery and sharing in open networked environments. It takes as input two ontologies and outputs a set of correspondences between concepts having the closest meaning. The H-Match approach is based on a weighted sum of different affinity measures that yield in a final measure called similarity affinity. Finally, based on thresholds, the best set of similarity affinities is returned to compose the final alignment. Moreover it proposes a dynamic setting that permits to adapt the matching strategy at run-time.

Although the interest of matching algorithms proposed by these systems, we have not included in this survey most of them because they do not support automation for the whole design process. They assume inputs composed by two sets of entities, mostly well formed ontologies, and do not consider the interpretation of a large input corpora from which could be derived ontological knowledge (i.e., axioms, concepts, roles, etc.). Moreover the ontology evolution step is out of their target. On the opposite, we describe a general approach for **automatic ontology construction** which consists of a sequence of phases that are to be followed during automatic ontology construction. If necessary, some of the steps have to be repeated until a satisfactory result is achieved. Sometimes, the individual steps can (should) be supported by automated validation techniques.

The process is depicted in Figure 1.3. The five proposed steps are:

- **Information Extraction**. This step is responsible for the acquisition of information needed to generate the ontology (concepts, attributes, relationships and axioms) and to handle the different source formalisms. Input sources can be of many kinds: structured, semi-structured or unstructured. Techniques for information retrieval and extraction can be of different types, such as NLP (Natural Language Process) techniques (for unstructured corpora such as text documents), clustering, machine learning, semantics, morphological or lexical and more often a combination of them. Large corpora can be grouped in different clusters. Normally the extracted information is formalized in an adequate format that makes sources descriptions uniform and facilitates the following tasks.
- **Analysis**. This step focuses on the matching and alignment of formalized input sources. This step requires: matching techniques, as morphological and lexical analysis of labels; a semantic analysis to detect synonyms, homonyms and other relations of this type; an analysis of concept structures to find hierarchical relationships and identify common attributes; techniques based on reasoners to detect inconsistencies.
- **Generation**. This stage deals with the merging/integration problem, if appropriate, and the formalization of the specific format adopted in previous tasks in a more general

ontological format, such as OWL. The merging task is often driven by heuristics and rules.

- **Evolution**. Depending on the usage, an ontology is often not a static description of a domain, but with the time the ontology may also require some changes (for example in professional exchanges a new partner can arise in a business collaboration and require a dynamic integration of his business semantics). A number of concepts as well as properties, relationships, and other parameters can be added or modified. As shown in Figure 1.3, the whole process is considered to be a cycle where the evolution step is responsible of managing changes in a compatible way. This operation is considered as an addition of new requirements and as such could be followed by a new step of information extraction, if new resources are not yet in the required format, or directly by the analysis step in order to provide new matches and alignments. Anyway, this step evaluates the ability of tools to solve and take care of the change problems.
- **Validation**. All previous steps may introduce wrong concepts and relationships, thus a validation task of the final result is needed. Conversely, a validation task can be introduced at the beginning of each task to verify input correctness and at the end of each step to verify the consistency. This step is often done by hand, but in some cases validation can be automated or simply supervised.

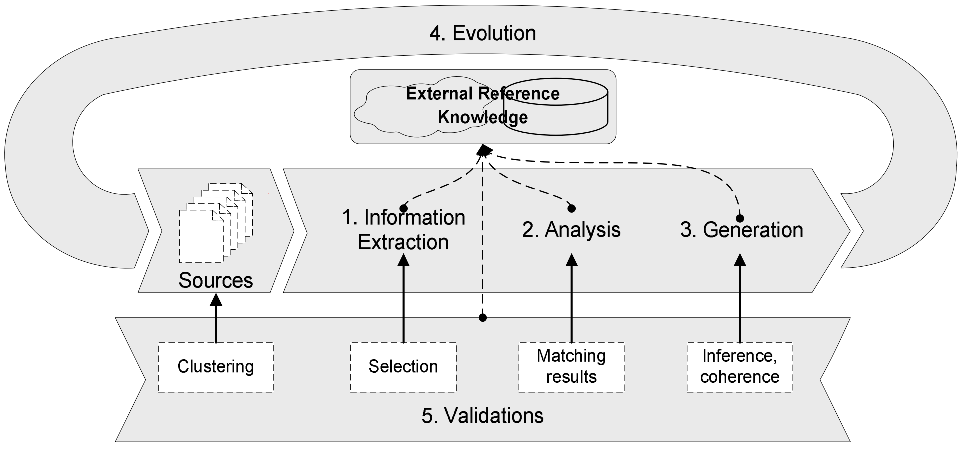


Figure 1.3 – Ontology generation life-cycle

In the following, we group the various considered systems in different subsections according to their focus. As often when classifying works, the border line is not always well defined and in our case applications can present more aspects, therefore we share works with respect to their automation approach rather than with regards to the techniques they implement. In fact we support the thesis that there is not a single technique to develop, but that only an appropriate mix of techniques can bring us to our goal.

1.2.3 Direct Transformation Approach

Several works propose direct **transformation**, schematically depicted in Figure 1.4, from input sources format to an ontological language. The transformation is merely done over a predefined mapping table from the conceptual information represented by the source format, such as XML schemata or conceptual model like UML. Applications of this approach make the hypothesis that concepts and relationships are already well defined in the input source and often they do not change the starting information model richness. What is interesting here is that they show that the ontology format representation subsumes other common knowledge representation, such as XML or UML. They also propose software that simply produces this transformation. Experiences show that this approach presents a high degree of automation, even if the final result is generally a light ontology. (However it still remains an interesting result to know that if we are confronted with two different representation formats, the solution is not always complex).

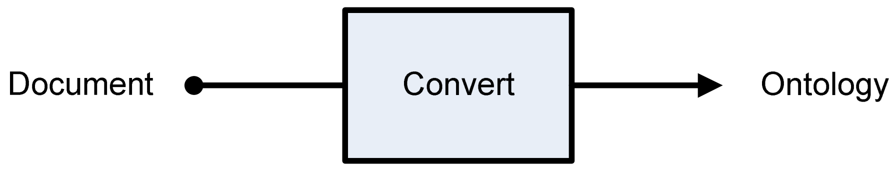


Figure 1.4 – Ontology generation direct transformation approach

XSD	OWL
xsd:elements, containing other elements or having at least one attribute	owl:Class, coupled with owl:ObjectProperties
xsd:elements, with neither sub-elements nor attributes	owl:DatatypeProperties
Named xsd:complexType	owl:Class
Named xsd:SimpleType	owl:DatatypeProperties
Xsd:minOccurs, xsd:maxOccurs	owl:minCardinality, owl:maxCardinality
xsd:sequence, xsd:all	owl:intersectionOf
xsd:choice	combination of owl:intersectionOf, owl:unionOf, owl:complementOf

Table 1.5 – XSD to OWL correspondences

1.2.3.1 Mapping XML to OWL Ontologies

Sören Auer *et al.* [55] of the University of Leipzig (Germany) have developed a tool that converts given XML files to OWL format. It is based on the idea that items specified in the XSD file can be converted to ontology classes, attributes and so on. Table 1.5 shows in detail the mapping between these two formalisms. Technically they have developed four XSLT[1] instances to transform XML files to OWL, without any other intervention on semantics and structures during the transformation. Finally

[1] Extended Style Sheet Transformations - http://www.w3.org/TR/xslt

the OWL file (read ontology) is automatically generated, but under the assumption that concepts were already correctly represented in the source file. This method has been also applied to the Ontowiki platform [56].

1.2.3.2 OWLMap

Matthias Ferdinand *et al.* [57] also propose direct mappings from XML Schema to OWL. Furthermore they describe mappings from XML to RDF, but these mapping are independent of each other. That means, that OWL instances have not necessarily to suit to the OWL model, because elements in XML documents may have been mapped to different elements in OWL.

1.2.3.3 UML to OWL

Dragan Gasevic *et al.* [58] advocated the use of UML profiles to extend the possibilities of representation of UML. In this way they get a larger UML representation that overcomes its limitations and that can be *translated* into OWL, again through a system of XSLT instances. As before the hypothesis is that the source of the transformation is complete and well-defined by an expert at an early stage to represent the ontology, the subsequent ontology generation is performed automatically.

1.2.3.4 Semi-automatic Ontology Building from DTDs

Within the PICSEL project, a collaboration between INRIA Future and France Telecom, Giraldo and Reynaud [6] have developed a semi-automatic ontology generation software for the tourism industry domain extracting information contained in DTD files. This experience is interesting because it goes further, in respect to the XML to OWL transformation seen previously, and shows that *tags and structure of XML files have sufficient information to produce an ontology*. What makes their solution semi-automatic is the fact that the detection of abbreviations or false positives[2] is left to an expert during the ontology validation task. This experience is really close to the use case adoption proposed in Chapter 2, but is limited to the sole domain of tourism, which is defined in advance with great precision, and therefore the detection of relevant concepts does not produce conflicts between different representations.

1.2.4 External Resource Integration Approach

Some works are based on **external knowledge** resource to build or enrich a domain ontology, a simple schema is presented in Figure 1.5. This approach can be also divided in two sub-approaches. One aims to produce a sub-ontology from a main upper ontology, while the second refines/enriches retrieved ontological knowledge from a more detailed external resource. In both cases some *seeds* are either manually or automatically defined from the input source, and the external resource is queried in order to derive new knowledge.

[2] A false positive is a misjudgement detection of a program.

This category may sometime overlaps a mining based approaches because techniques applied to retrieve seeds and to interpret queries on the Web can be similar; nevertheless we classify here experiences with an approach closer to the integration of external dictionaries, existing ontology or from a more general knowledge resource, like WordNet [59] or the Web.

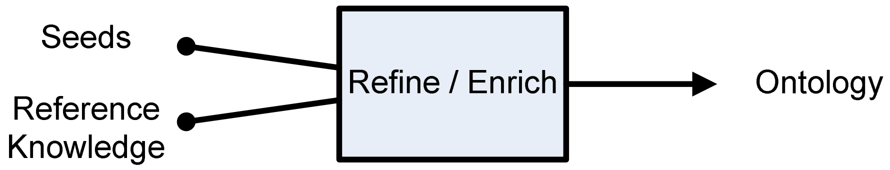


Figure 1.5 – Ontology generation external resource integration approach

1.2.4.1 SALT

D. Lonsdale *et al.* of Brigham Young University, England, propose a process to generate domain ontologies from text documents [60]. Their methodology requires the use of three types of knowledge sources which are: 1) a more general and well defined ontology for the domain, 2) a dictionary or any external resource to discover lexical and structural relationships between terms and 3) a consistent set of training text documents. With these elements they are able to automate the creation of a new sub-ontology of the more general ontology. User intervention is required at the end of the process because it can generate more concepts than required. This behaviour is acceptable because the withdrawal of false positives is easier than adding missing concepts. The authors state that with a large set of training documents their solution can achieve really good results. However the hypothesis of having an upper ontology well defined beforehand proves that the **NLP approach can be used in complement** of the automatic ontology generation process.

1.2.4.2 Learning OWL ontologies from free texts

He Hu and Da-You Liu from Renmin and Jilin University, China, have developed an automatic generation [61] based on the analysis of a set of texts followed by the use of WordNet. The analysis of the corpus considers words as concepts. These words are then searched in WordNet to find the concepts associated with them. The ontology generation seems to be one of the most automated, but no details of how the terms are extracted from the body text as well as any qualitative assessment of the work are provided. Nonetheless, it remains an interesting experience to the extent it demonstrates once again that automation is easier if a *more general reference knowledge* already exists, which the authors argue can be represented by WordNet.

1.2.4.3 Design of the Automatic Ontology Building System about the Specific Domain Knowledge

Hyunjang Kong *et al.* [62] of the University Chosun, Korea, have developed a method based on WordNet. In this method, WordNet is used as a *general ontology* from which they *extract a subset of* *"concepts"* to build a domain ontology. For example, consider a user trying to generate an ontology on wine. The software will query WordNet using this term and create classes of concepts based on the results of the query. After this initial pass, the user can extend the ontology by entering new concepts to be included. The ontology is then exported in OWL format. Depending on the quality of the starting knowledge resource, this approach will be more or less satisfactory. It is also dependant on the targeted area.

1.2.4.4 Domain-Specific Knowledge Acquisition and Classification Using WordNet

Dan Moldovan and Roxana Girju from the University of Dallas expose a method for generating ontologies [63] based on WordNet. The approach is almost the same as the previous [62], a user defines some "seeds", i.e. concepts of the domain, but with the difference that if a word is not found in WordNet then a *supplementary module will look for it over the Internet*. Then linguistic and mining techniques extract new "concepts" to be added to the ontology. This method automatically enriches its corpus retrieving sentences about the seeds of the ontology that were not found in WordNet. User intervention is necessary here to avoid incongruous concepts.

1.2.4.5 Enriching Very Large Ontologies Using the WWW

Agirre *et al.* [64] have developed a strategy to enrich existing ontologies using the *WWW to acquire new information*. They applied their approach to WordNet, which is often accused of two flaws: the lack of certain links between concepts, and the proliferation of senses for the same concept. The method takes as input a word which one wants to “improve” the knowledge. WordNet is questioned about this word, and the different meanings of words are used to generate queries for the web. For each query, that constitutes a “group”, different search engines are queried and the first 100 documents are recovered. Terms frequencies are then calculated and compared with each group, and of course the winning group, (i.e., sense), for the concept is the one with the highest frequencies. In addition a *statistical analysis* is performed on the result, in order to estimate the *most common meaning* of the concept. This method alone cannot be adopted to build ontologies, but it has the merit to be able to iterate with an external knowledge base to provide further information that may be used for the validation task of an ontology in absence of human intervention.

1.2.4.6 A new Method for Ontology Merging based on Concept using WordNet

Miyoung Cho *et al.* [65], from Cheju Universities in Korea, present the problem of proximity between two ontologies as a choice between alignment and merging. The first case is limited to establishing links between ontologies while the second creates a single, new ontology. With their experience they directly merge two ontologies based on WordNet. For this they use two approaches in their method that they call the horizontal approach and the vertical approach. The horizontal approach first checks relationships between concepts of the “same level” in the two ontologies and merges or ties them as defined by WordNet. The vertical approach completes the merging operation for concepts with “different levels”, but belonging to the same branch of the tree. In this case they fill the resulting ontology with concepts from both ontologies and do not make a choice. A similarity measure is calculated in order to define the hierarchy between these concepts in the resulting tree. Figure 1.6 shows an example this kind of matching, where C_1 and C_4 of O_1 are mapped to their equivalent concepts in O_2, while C2, C3 have not direct equivalence. Thus the vertical approach is applied to the remaining concepts in order to define a concept hierarchy among them, and finally merged as illustrated always in Figure 1.6 in the right side.

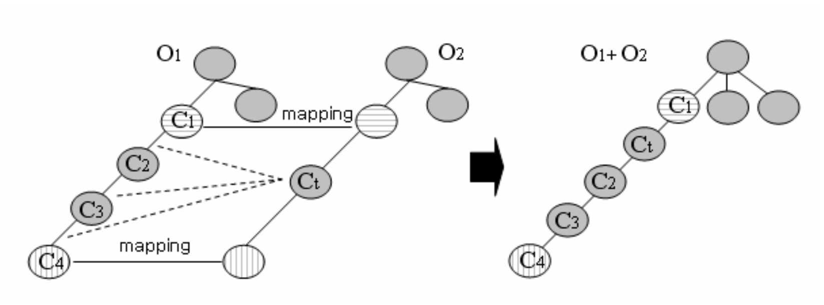


Figure 1.6 – Sample of Vertical approach merging using similarity measure

This method, while not providing an adequate solution to automation, does provide a *purely semantic approach* to the merging solution.

1.2.4.7 A Method for Semi-Automatic Ontology Acquisition from a Corporate Intranet

Similar to [61], Joerg-Uwe Kietz, Alexander Maedche and Raphael Volz [66] describe a generic approach for the creation of an ontology for a domain based on a source with multiple entries which are: a generic ontology to generate the main structure; a dictionary containing generic terms close to the domain; and a textual corpus specific to the area to clean the ontology from wrong concepts.
This approach combines several input sources, allowing great generality and a better reliability of the result. The user must manually check the ontology at the end of the generation process.

1.2.5 Ontology Generation Intermediary Model Approach

Another approach is to use an **intermediary representation** of input sources, presented in Figure 1.7. Sources are mined and interpreted in order to produce a more generic format to be further transformed into ontology. The kind of intermediary format depends on the type of input source. First, if it is an unstructured corpora it is mainly represented by a list of words which constitute candidate concepts; later by the integration of an external resource it can be enriched (as already showed in the approach above) in order to get ontology knowledge. Second, the intermediary format can be a conceptual or semantic model which provides a higher level of flexibility when we are in presence of more than one group in input sources (to integrate two or more schemas). In such a case, each input cluster is transformed in the concept model, on which matching and merging operations are applied, before to obtain the final ontology.

A lot of experiences focused on unstructured sources, like text documents or web pages; they use Natural Language Processing (NLP) techniques. These experiences tell us that recovering structured concepts from unstructured documents still requires human assistance and that mining techniques from natural text can be used only in complement with other existing structured knowledge representations.

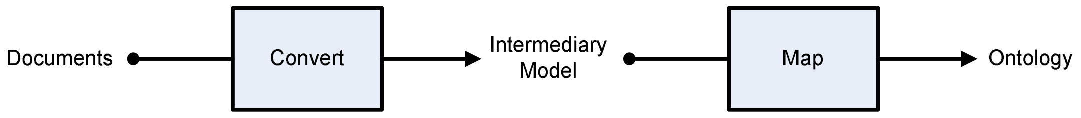


Figure 1.7 – Ontology generation intermediary model approach

1.2.5.1 TERMINAE

Biebow and Szulman [67] of the University of Paris Nord presented the TERMINAE method and tool for building ontological models from text. Text analysis is supported by several NLP tools (such as LEXTER [68]). The method is divided into 4 stages: corpus selection and organisation; linguistic analysis with the help of several NLP tools; normalization according to some structuring principles and criteria; formalization and validation. An expert is called to select the most important notions (concepts) for the targeted ontology from the list of *candidate terms extracted by the tool* and to provide a definition of the meaning of each term in natural language. The new terminological concept finally may or may not be inserted into the ontology, depending on the validity of the insertion.

1.2.5.2 A method to build formal ontologies from text

Originating from the same University, Jerome Nobécourt has developed a method [69] based on TERMINAE that allows an automation of the insertion of concepts into the ontology by the adoption of ***successive refinements*** of the selected concepts: while the classic TERMINAE approach requires the hypothesis that the ontology is a static property of the domain, the latter introduces a more dynamic environment for domain ontology.

1.2.5.3 Ontology Construction for Information Selection

Latifur Khan and Luo Feng of the University of Texas demonstrated a method to automatically construct an ontology from a set of text documents [70]. Their overall mechanism is as follows: 1) terms are extracted from documents with text mining techniques (i.e. removed stop words, words stemm and *tf*idf* calculation); 2) documents are grouped hierarchically according to their similarity using a modified version of SOTA algorithm[3] and then; 3) a method based on the Rocchio algorithm[4]

[3] Joaquin Dopazo and Jose Maria Carazo. Phylogenetic reconstruction using an unsupervised growing neural network that adopts the topology of a phylogenetic tree. Journal of Molecular Evolution, Volume 44(2) :226/233, 02 1997.

[4] Thorsten Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In

is used to assign concepts to the tree nodes starting from leaf nodes. Concept assignment is based on WordNet hyponyms[5]. This experience introduces a new *bottom-up* approach for ontology generation that seems to produce good results without any human intervention. The bad news is that it also needs a more general ontology to define concepts for the targeted ontology, but as we can see, this is generally the case of all text mining based methods.

1.2.5.4 Learning concept hierarchies from text corpora using formal concept analysis

Cimiano *et al.* [71] address the learning of taxonomic relations from text corpora. The overall process of automatically deriving concept hierarchies from text is depicted in Figure 1.8. First, the corpus is part-of-speech (POS) tagged[6] and parsed, thus yielding a parse tree for each sentence. Then, verb/subject, verb/object and verb/prepositional phrase dependencies are extracted from these parse trees. In particular, pairs are extracted consisting of the verb and the head of the subject, object or prepositional phrase they subcategorize. Then, the verb and the heads are lemmatized, i.e. assigned to their base form. In order to address data sparseness, the collection of pairs is smoothed, i.e. the frequency of pairs which do not appear in the corpus is estimated on the basis of the frequency of other pairs. The pairs are then weighted according to some statistical measure and only the pairs over a certain threshold are transformed into a formal context to which Formal Concept Analysis is applied.

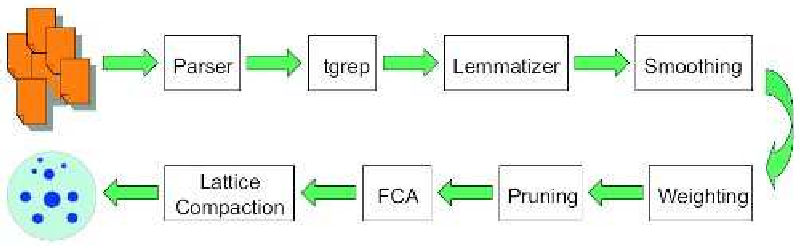


Figure 1.8 – Learning concepts hierarchies from text corpora overall process

The lattice resulting from this is transformed into a partial order which is closer to a concept hierarchy in the traditional sense. As FCA typically leads to a proliferation of concepts, the partial order is compacted in a pruning step, removing abstract concepts and leading to a compacted partial order which is the resulting concept hierarchy.

1.2.5.5 Generating an ontology from an annotated business model

The L3I laboratory of the University of Rochelle has developed a semi-automatic ontology generation process [72]. This process starts from a UML class diagram representation of the ontology domain,

Douglas H. Fisher, editor, Proceedings of ICML-97, 14th International Conference on Machine Learning, pages 143/151, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[5] A word that denotes a subcategory of a more general class. Opposite of hypernym.

[6] Part-of-speech tagging consists in assigning each word its syntactic category, i.e. noun, verb, adjective etc.

made by an expert that annotates the elements to be introduced into the ontology. This UML model is then transformed into ODM format[7] as pivot model before automatically generating the ontology in RDFS format. As in the previous case some degree of human intervention is needed at an early stage.

1.2.5.6 A Bottom-Up Approach for Integration of XML Sources

The solution proposed by Santos Mello *et al.* [73] [74] [75] shows an interesting level of automation with an approach really close to our needs. The ontology generation is viewed as a particular case of the integration of input XML data sources. Figure 1.9 illustrates the architecture of their solution, which is composed by three layers. From the Data Access Layer, the Mediation Layer receives the DTDs of the XML Access Modules. An XML Access Module is a functional unity that provides access to an XML data source. Each XML source keeps data instances that are in accordance to a DTD. Document databases and wrappers are examples of XML Access Modules. Based on the set of DTDs, the integration process is performed in two steps. In the first step, a local conceptual schema is generated as an abstraction of each DTD, through the DTD-Conceptual Schema Conversion module.

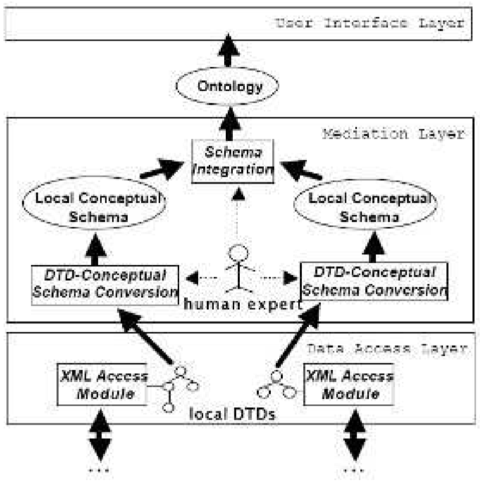


Figure 1.9 – Integration architecture centered on a Mediation Layer

This conceptual schema models DTD elements and attributes as related concepts with associated mapping information. The further human intervention validates mapping defaults. In the second step, local conceptual schemata are integrated to generate an ontology. The ontology provides a unified conceptual vocabulary for all DTD elements and attributes; it acts as a front-end for semantic queries originated from the User Interface Layer. The module that performs such task is called Schema Integration. During semantic integration, local concepts are mapped (based on an analysis of equivalencies and conflicts) to global concepts. The human expert intervenes again to select the best integration alternatives.

The conceptual model they use is necessary to reduce the complexity of the integration process, each DTD is so converted to a conceptual schema in the Canonic Conceptual Model (CCM). CCM is a

[7] Ontology Definition Metamodel – http://www.omg.org/ontology/

conceptual model suitable for semi-structured schemata representation based on ORM (Object with Role Model) [76] and ER (Entity-Relationship) [77] models. This model seems well fitting the matching of structured sources, but it is based on the hypothesis that input sources have the same level of granularity and thus simple correspondences, otherwise their underlying model is not adequate and needs improvement. Moreover authors claim that their approach is applicable to more than two input sources at a time, however no details and tests are provided, as well as implementations are missing to prove the feasibility of the whole approach.

1.2.6 Framework Approach

Solutions based on a **Framework** approach, simply represented in Figure 1.10, are generally more complete and produce best results. Often these kinds of solutions are delivered as part of an ontology editor and integrate different modules to achieve the goal. However seeing that each module can provide several options and parameters to set the integration of modules remains almost a human task.

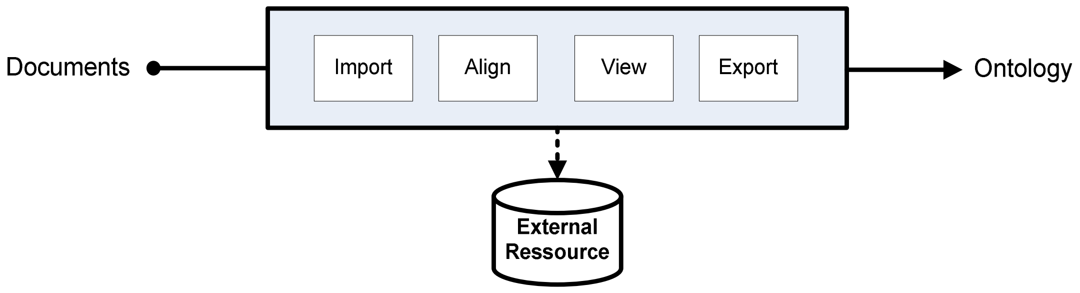


Figure 1.10 – Ontology generation framework approach

1.2.6.1 Symontox: a web-ontology tool for e-business domains

SymOntoX [78] is an OMS (Ontology Management System[8]) specialised in the e-business domain, which provides an editor, a mediator and a versioning management system. With SymOntoX the creation of the ontology is mainly done by an expert using the editor. But the framework contains a first step towards an easier generation: it contains high-level *predefined concepts* (such as Business Process, Business Object, Business Actor, etc.), as well as *different modules* used for ontology mapping and alignment to simplify the work of the expert. Here, ontology generation is merely assisted.

1.2.6.2 Protégé

Protégé [79] is a free open source, platform to design ontologies. Developed by the Stanford Medical Informatics group (SMI) at the University of Stanford, it is supported by a strong community and experience shows that Protégé is one of the most widely used platforms for ontology development and

[8] Ontologie Managment System. http://sw-portal.deri.at/papers/deliverables/d17_v01.pdf.

training. This software has an *extensible architecture* which makes it possible to integrate plug-ins[9]. Some of these modules are interesting and relevant to our case, like those from the PROMPT Suite [50]. They automate, or at least assist, in the mapping, merging and managing of versions and changes. Also the related project Protégé-OWL offers a library of Java methods (API-Application-Programming Interface) to manage the open-source ontologies formats OWL (Web Ontology Language) and RDF (Resource Description Language).

The glue between these pieces of software still remains human, yet program modules and libraries provide a fundamental basis for developing the automation of ontology generation.

1.2.6.3 Ontology Learning Framework

Alexander Maedche and Steffen Staab at the University of Karlsruhe, Germany, are contributors of several interesting initiatives within the ontology design field as well as the automation of this process, like the MAFRA Framework [80], Text-To-Onto [81] and KAON [82]. In this paper we focus on their framework for ontology learning [83].

They propose an ontology learning process that includes five steps (illustrated in Figure 1.11): import, extraction, pruning, refinement, and evaluation. This approach offers their framework a flexible architecture that consists of many extensible parts, such as: a component to manage different input resources, capable of providing information extraction from a large variety of formats (UML, XML, database schema, documents text and web); a library of algorithms for acquiring and analyzing ontology concepts; a graphical interface that allows users to modify the generated ontology, but also to choose which algorithms to apply and treatments to perform.

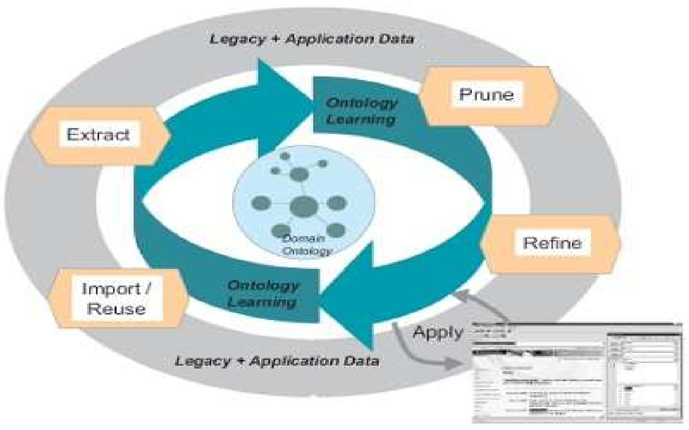


Figure 1.11 – Ontology Learning process steps

They bring together many algorithms and methods for ontology learning. Despite their framework not allowing a completely automatic generation process, they are the only researchers to propose a *learning process* close to a methodology for automatic ontology generation.

[9] A hardware or software module that adds a specific feature or service to a larger system.

1.2.6.4 LOGS

A group of researcher from Kansas State University has developed LOGS (Lightweight universal Ontology Generation and operating architectureS) [84]. They state that generating ontology automatically from text documents is still an open question. Therefore they developed LOGS with a modular architecture that integrates the core functionality that can be expected by automatic ontology building software. It consists of the following modules: document source parser, NLP engine, analyser, ontology engine, interface, integrator, ontological database and dictionary. It also contains other modules able to crawl an intranet, to refine the process of ontology design and a module implementing trial and error iterative analysis of related texts to find known patterns. Although no qualitative analysis is provided, the authors argue that they obtained significant results.

1.2.7 Comparative Analysis and Discussion

Works presented above are only a part of all studied experiences; nevertheless they represent a significant sample covering the essential steps and approaches in the automatic generation of ontologies.

Firstly we can note that modules implementing a step have a different degree of automation, which can not be measured exactly. However we can observe that **transformation approaches** are used to build ontology from structured or semi-structured sources, but with low degree of integration and matching tasks. Between them only the work from Giraldo *et al.* [6] implements a method to extract knowledge from more than one file at once; but it can still be considered as a single input cluster. Thus we can state that systems adopting a transformation approach can be adopted only to transform one cluster at once because they do not provide solution aiming the reconnaissance of similar information from different sources (merely clustering, alignment and merging solutions). Furthermore this approach requires human intervention at initial stage to select sources with compatible content. It reaches a good level of automation but low generality (applied to only one input at once) and high human implication at the early stage. In this approach sources are directly mapped to an ontological language, which can be used as a preliminary step before merging several input clusters to produce a larger common ontology.

Systems based on **external resources** are too much tied to the resource itself. As far as we know upper ontologies are not detailed enough to provide a real support for the automatic construction of a domain or application ontology. This makes difficult to generate ontologies from scratch with this approach. The usage of the Web is interesting, but such knowledge is too much heterogeneous in both format and content. Its adoption can entail others problems and makes things more complex than what they are. However it can be used as complement to refine a generated ontology, like the work done by Agirre *et al.* [64], or to validate resulting correspondences, like using a deductive approach from a query about contrasting correspondences. But as far as we know no system still implements such an approach. Human intervention is mainly needed at the starting point and at the end of the process, to define seeds and to filter results. To this end WordNet [59] surely deserves some special attention because we observe that it is an essential resource for the automation process systems. In fact it is used by large parts of works with different roles. The first is that of an electronic dictionary and thesaurus, which is fundamental. The next is that of a reference ontology, mainly by using its sibling or hierarchical terms discovery, with relationships like hyponym, meronym, holonym and hyperonym. But for this WordNet has the drawback of being too generic and not adapted to specific domain ontology development. Even so, it remains an important module to further be developed.

Approaches adopting an **intermediary model** gather a more flexible behaviour. This approach seems to be indispensable in the case where more than one input source is available. It permits to leverage different input formats and to highlight required information. The definition of such a model can be conceptual or object oriented, but it is often specific to implantation features. It is often adopted by advanced matching systems, but very few provide a public formalisation (this topic will be discussed in detail in Chapter 3). Human intervention is mostly needed to validate the final model instance, but generally the transformation from the model to the ontology language is error-safe. Disadvantages of this approach are: the double mapping, from the input source to the model and from the model to the ontology, which implies lost of efficiency; and the risk to lose knowledge not handled by the model.

The **framework** based approaches are the only one to execute each task of the life-cycle proposed in Section 1.2.2. The SymOntoX system provides some specific predefined construct for e-business ontologies. Protégé, like several other ontology editor, is able to integrate external modules and thus is able to manage several ontology generation requirements, even if its current graphical plug-ins are not scalable in presence of large ontologies. Thus as general rules this approaches is the best to follow for our goal, even if their usage is not allowed in run-time environment because it requires human intervention at each stage in order to provide the best module to be adopted.

Concerning **input sources**, information extraction can reach good results. The most studied input corpora are text documents. A lot of information can be extracted from this type of corpus source. Methods based on this corpus have the advantage to have a lot of resources, that can be found over Internet or an Intranet, and that several NLP and mining software are available. Nevertheless they require a most important human validation task and are preferred for defining a high level definition of concepts, or a taxonomy, which limits reasoning capability on resulting ontology. Structures, like classes, attributes and relationships, are mostly provided by other external resources not always available. Thus structured and semi-structures sources are better positioned to achieve our task. But unfortunately extract knowledge from large corpora for this kind of source remains a complex task and we did not meet any experience providing free tools or APIs that can be easily integrate in other application. Moreover information extraction from semi-structured sources, like XML, need further research work in order to exploit at best contained semantics to produce well defined ontologies also at semantic level, rather than provide a simple direct map of structural knowledge. This is true at least for those systems we have tried to extract ontology knowledge from XML files, like XML2OWL [55] and Mafra [80]. Derived concept names maintain exactly the same label used in sources, which often are abbreviations or incomprehensive tag names.

Matching and **alignment** modules are one of the most challenging tasks but, as testified by the different Ontology Alignment Evaluation Initiative [7], [8], [9] there is a lot of ongoing research on this matter. Always from the OAEI initiative we can also observe that more and more systems managed to produce better quality results over the years. This means that we can expect that they will be able to provide useful and integrable APIs for applications requiring this kind of intelligent piece of software embedded.

At present theoretical works as well as implementations for merging and source integration tasks are developed with two input ontologies. They make the strong hypothesis that **multi ontology** merging and matching is just a derived case of two inputs. But from some tests we have conducted it seems to be not always true and current algorithms are not efficient enough and scalable for combing the merge of two sources with others. We analyse this issue later in Section 5.1.1.

Evolution management is still rare. Some methods manage versions and other go further and provide automatic detection of changes. But in reality what we are really looking for, more than ontology generation, is also the possibility to manage dynamic environments. This can be done with ontology able to grow as sources are added incrementally and not a static adaptation of knowledge representation.

Validating an ontology means ensuring that the ontology is a good representation of the domain that it is supposed to model. Reasoning is at the basis of validation done automatically (or at least supported by automated tools). From the survey we observe, validation still remains human and only automatic consistency checking and some pruning methods have been implemented. However it is probable that in the few years to come most researches will be focused on this topic.

It is difficult to evaluate ontologies generated by systems. As seen in Section 1.1 a DL ontology deals with basic entities like concepts and roles, and with constructors and axioms defined over such basic entities. Between them at least high level concepts are derived from all methods. It is more difficult to say something about role and function derivations. Very few details are provided in reviewed papers. So we can at least affirm that systems based on mining texts like Cimiano *et al.* [71] more than concepts are also able to produce subsumption relationships (i.e. $A \sqsubseteq B$), which provides concepts hierarchy, and some concept equivalences (i.e. $A \equiv B$). Wordnet based techniques also discover some properties (like *part of*) with the usage of meronym relationships or also equivalences on individuals (i.e. *samaAs,* $\{x\} \equiv \{y\}$) based on synonyms. But as told above WorldNet is too generic and concepts can have more than one meaning, thus without context information resulting relationships can be false. Properties can be derived more naturally from structured sources, as shown from the XML2OWL experience, which provide also a basic map from XML schema structures to OWL union and intersection constructors.

The analysis of Table 1.6 below summarizes surveyed works, w.r.t. our approach to ontology generation automation life-cycle presented in Section 1.2.2. It should also be noted that qualitative results were not always available and when conducting this assessment only few tools presented in this table were both freely available and able to process XML Schema files (as required by the use case we were evaluating), and therefore specifically tested by us. These are Protégé, XML2OWL and MAFRA.

Despite this lack of availability, the purpose of this study is mainly theoretical, thus information obtained by public material was enough to perform at least a preliminary evaluation. Values are assigned to each step according to the following criteria:

- - – when step is not developed;
- O – for solutions using a semi-automatic approach ;
- + – for solutions where human intervention is optional;
- ++ – for solutions that show the best automation level.

	Extraction	Analysis	Generation	Validation	Evolution
Generating an ontology from an annotated business model	- Human	-	+ – No merging. Direct transformation using XSLT files.	- Human, upstream to the generation	-
XML2OWL	++ – Static table of correspondences	-	+ – No merging. Direct transformation using XSLT files.	- Human, upstream to the generation	-
UML2OWL	+	-	+ – No merging. Direct transformation using XSLT files.	- Human, upstream to the generation	-
Semi-automatic Ontology Building from DTDs	+ – automatic extraction from DTD Sources	O – structure analysis without alignment	+ – No standard ontology representation	- Human	-
Learning OWL ontologies from free texts	+ – Text sources. NLP techniques. WordNet as resource dictionary/ontology	-	+ – OWL format	-	-
Ontology Construction for Information Selection	+ -	-	+	-	-
TERMINAE	+ – Text sources. NLP techniques	O – Concept relationships analysis	+ – No standard ontology representation	- Human	-
SALT	++ – Text sources. NLP techniques. Multi entries.	+ – Similarity analysis of concepts	o – No standard ontology representation	o –Limited human intervention	-
A new Method for Ontology Merging based on Concept using WordNet	-	O	+ – Automatic merging. No standard ontology representation.	-	-
Design of the Automatic Ontology Building System about the Specific Domain Knowledge	o – Main concept defined by a domain expert.	-	+	-	-
Enriching Very Large Ontologies Using the WWW	+ – Enrich existing ontology	-	+	-	-
Domain-Specific Knowledge Acquisition and Classification Using WordNet	++ – Main concept defined by a domain expert.	O – Grammatical analysis of text	+	- Human	-
A Method for Semi-Automatic Ontology Acquisition from a Corporate	++ – NLP techniques. Multi entries source.	O – Meaning analysis of concepts	O	O – User required for undecidabe cases	o – Cyclic approach can manage evolutions

Intranet					
SymOntoX	-	+ – Matching analysis	+ - Provide some predefined concepts.	- Human	o – Manage versions, but still human.
Protégé (Mainly from plug-in)	+ – extraction from Relational DB and some XML format	++ – Matching and Alignment analysis.	o – Assisted merging. Export in several ontology formats.	- Human	+ – Ontology evolution detection
LOGS	++ – Text source analysis. NLP engine. Morphological and semantic analysis. Machine learning approach for rules.	+ – Similarity based on concepts and relationships analysis.	+ – Different format. Internal ontology structure based on a lattice.	O – Validation at the end of each module	-
Ontology Learning	++ – Extraction from several formats (XML, UML, OWL, RDF, text…). NLP, Semantic and lexical analysis. Multi entries source.	+ – Libraries for clustering, formal concept analysis and associations rules	+ - OWL and RDF/S	o - Assisted	-

Table 1.6 – Comparative analysis of methods

As final consideration we can say that most methods offer automations of only some steps of the generation process. Modular solutions, rather then monolithic applications should offer a better architecture for covering the larger part of the ontology life cycle, and to achieve this result it is essential to dispose of specialized program libraries to integrate in most ambitious applications.

1.3 The Matching Problem

As shown above the automatic ontology generation process requires a matching task to handle different representations of similar concepts. Different ontologies or sources need to be confronted and related to each other, either to produce a single integrated and reconciled ontology that deals with a larger domain of interest or to establish a connection, with a precise semantics, between the different inputs, which can remain distinct. This implicitly means that if we want to retrieve concepts from different input sources, the information retrieval and subsequent matching task must be applied to different source formats. Even when input sources are either well formed ontologies or XML Schemas, definitions can be similar but also heterogeneous, semantics different, and thus the discovery of correspondences is probably the most basic, and at the same time the most challenging task that must be conducted. In this section we deeply present the matching process, in order to clarify what we mean with it.

1.3.1 Matching Simple Items

Before entering in the whole matching process description, we present the basic problem behind it, which is the matching operation. For that we define a **matching operation** as the function that look for correspondences between two or more input sources. For the sake of simplicity we limit the formal definition to two input sources.

Definition 1: ***(Matching Operation).*** Given two non empty set of elements $S = \{e_1, \ldots, e_n\}$ and $S' = \{e'_1, \ldots, e'_m\}$ with $m,n > 1$, the matching operation is a function $f : S \rightarrow S'' \subseteq S'$ that defines a precise correspondence between elements belonging to the different sets. Thus we say that $f(e_i) = \{e'_x, \ldots, e'_y\}$ *(or* $eR_f e'$*)* if it exists at least an element $e \in S$ that holds with an element $e' \in S'$.

The aim of such operation is to identify a possible **alignment** *A*, if any, between given input sources. An alignment is made up of a set of correspondences, derived from a matching operation, between pairs of elements belonging to different input sources.

Current approaches to similarity (correspondence) discovery usually adopt algorithms realizing the matching operation, with exponential computational complexity order [85]. The simple example below shows how algorithms often proceed.

Let C_1, C_2 and C_3 be three sets of generic concepts that we want to align:

- C_1 = *{person, address, account}*
- C_2 = *{organization, location, manager}*
- C_3 = *{umbrella, washing machine, location}*

```
M1,2={(person,organization),(person,location),(person,manager),(address,organization),(address,loc
ation),(address,manager),(account,organization),(account,location),(account, manager)}

A1,2={(person,manager), (address,location)}
```

Listing 1.3 – List of matching couples between C1 and C2, and the resulting alignment A12

Normally a matching operation implements different algorithms to be executed for each pairs of entities belonging to different sets. Thus if we consider the firsts two sets C_1 and C_2 we must execute algorithms between the following set of possible matchings $M_{1,2}$ before discovering that there are only two mappings with real meanings: $A_{1,2}$ (see Listing 1.3). Consider adding the $M_{1,3}$ and $M_{2,3}$ matchings. The global alignment *A* is still composed by the same two matchings, while algorithm has been executed 27 times ($=3^3$). Thus if we consider *n* to be the average number of concepts for each set and *m* the number of sets to match, then the resulting computational complexity order is $O(n^m)$. This simple example shows the overall approach to the matching operation problem and at the same time it highlights the need for a rational approach when the input is composed by more than two input sets.

1.3.2 Known Matching Features

As shown in [86] and [85], classical matching approaches lack of efficiency. This can be explained by three main reasons: (i) the algorithm computational complexity order, as exposed in [85]; (ii) the fact that algorithms compute measures between every couple of items of ontologies to map, even when they do not have anything in common (like looking for similarities between "*umbrella and sewing* *machine*”[10]); (iii) the lack of memorization : a comparison is done every time two items are met (like a *“Sisyphean task”*[11]), regardless of what has already been calculated.

The problem of matching has been investigated not only in the ontology area, but more generally into the area of data and knowledge management ([87], [53], [89], [50]). Reference surveys on schema and ontology matching are given in ([47], [90], [91], [46], [48]).

As we can see from all these works, many researchers in the Semantic Web and Knowledge Engineering communities agree that discovering correspondences between terms in different sets of elements is a crucial problem. Sometimes two ontologies refer to similar or related topics but do not have a common vocabulary, although many terms they contain are related. So this complex task requires the application of several algorithms (w.r.t Definition 1, each algorithm realizes at least a matching operation) and once again we lose efficiency.

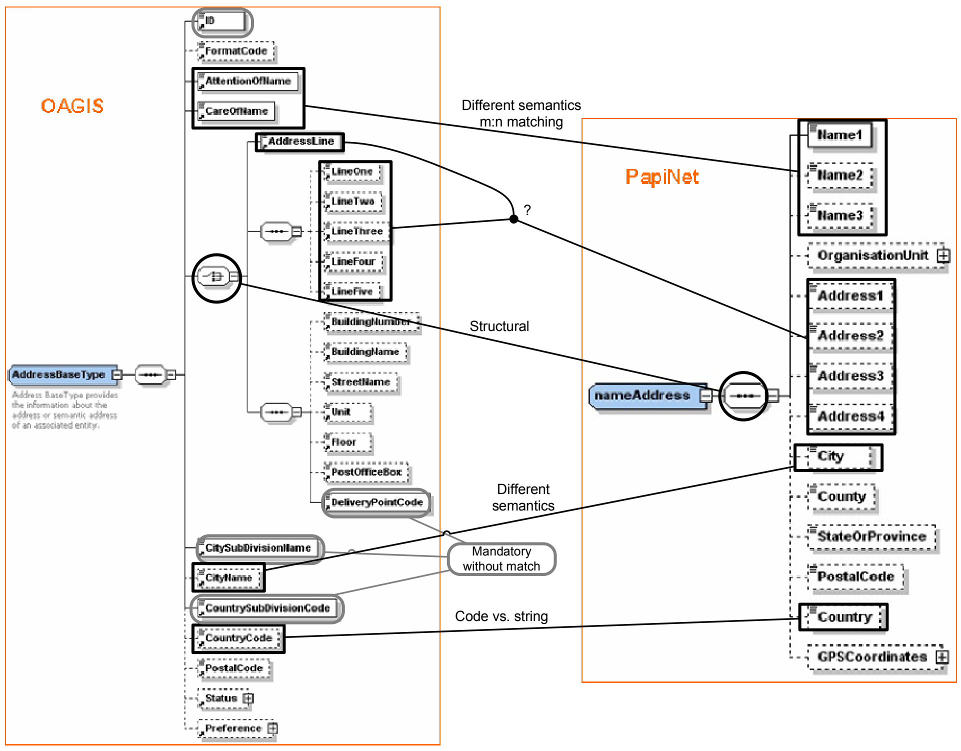


Figure 1.12 – Example of possible mismatchings between two XML Schemas definitions

Looking for correspondences between sets of elements more complex than that presented in the example above, Figure 1.12 illustrates a non exhaustive list of possible mismatching that can be hold between the definitions of a same high level concept expressed in XML Schema format. For instance

[10] Comte de Lautréamont, *Les Chants de Maldoror, VI, Roman*, 1869

[11] In Greek mythology Sisyphus was compelled to roll a huge rock up a steep hill, but before he reached the top of the hill, the rock always escaped him and he had to begin again (Odyssey, xi. 593).

the example shows two different vision of the concept *address* as defined by two B2B standards, OAGIS and Papinet. It is clear that although both of these standards are based on the "upper" standard UN/CEFACT CCTS, there are considerable differences in the resulting document fragments. This explains why we need more than one algorithm to discover possible similarities between two sets of elements. For this we provide a first classification of the nature of these algorithms categories: *syntactic*, *semantic*, and *structural*. A good process for matching discovery should cover at least these three categories and also implement a combination of them in order to improve results, as shown in [92] and [93]. As a result, a lot of time is spent computing these algorithms during the matching process.

1.3.3 The Matching Process

As already mentioned above matching problem can be approached from various standpoints and this fact is reflected by the variety of the definitions that have been proposed in the literature ([47], [48], [46], [94], [95], [13]). We observe that there are some recurring terms often leading to confusion and thus producing overlaps on the process definition. *Learning*, *matching*, *anchoring*, *alignment*, *transformation*, *mapping* and *merging* are almost used to this purpose. Figure 1.13 proposes a view about the role and sequence that each of these common terms play in the ontology "life-cycle" process.

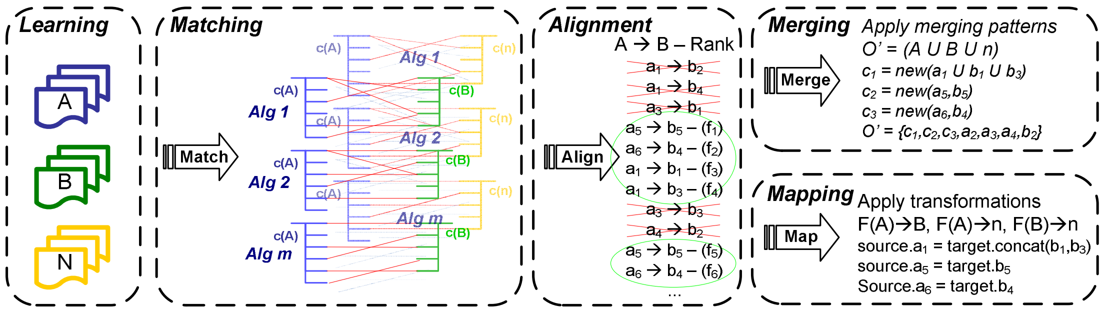


Figure 1.13 – Ontology learning, matching, alignment, mapping and merging phases

The *Learning* phase aims to extract knowledge information from sources handling their different representations. As output it provides a formal representation, sometimes an ontological view of inputs. From here we assume that we have two or more input ontologies. This term often refers to a larger operation that comprises the final ontology generation, but we prefer to use this term just to highlight the fact that ontological knowledge is mainly retrieved, thus learnt, at this stage of the process. The *Matching* phase realises similarity detections between input entities executing one or more algorithms. As described in the previously, the "matcher" (the application realising this phase) computes the algorithms for each couple of input entities and provides as output a list of the best matches found, selected on the base of parameters. The following *Alignment* phase tries to select the best set of correspondences between all those provided by the matcher. It permits to combine the different similarity algorithms executed previously and to provide a uniform view of correspondences, normally without inconsistencies. At this stage the match can be also contextualized, choosing a match rather than another because of heuristics practices or an existent upper ontology for the concerned domain suggests so.

Finally, according to the purpose, alignments can be used to merge input ontologies (*Merging* phase) or to transform instances of an ontology into another (*Mapping* phase).

This disambiguation permits us to well situate the problem that we want to address.

To our extent the *Matching process* considers only the matching phase described above. In our analysis we estimated that this is a core part that: i) mainly contributes to the computation time and; ii) is the most generic and thus reusable part. These are the main reasons that conduct us to look for a scalable solution to improve the whole ontology generation process in this phase.

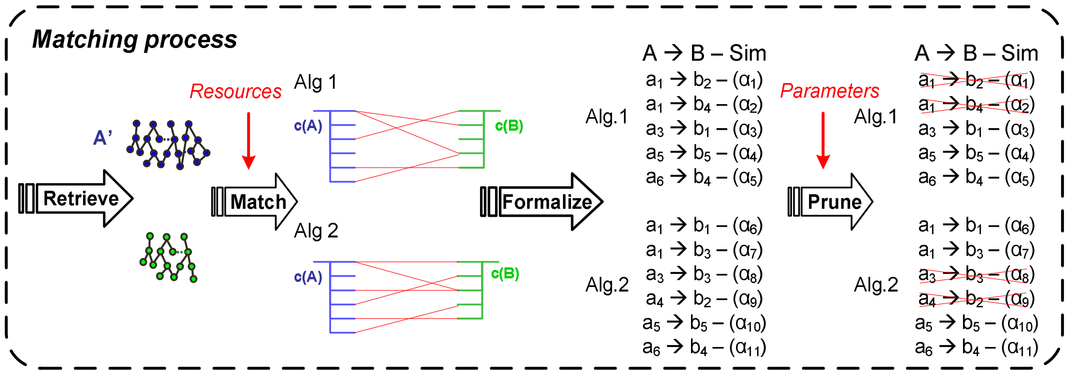


Figure 1.14 – Matching process details

As shown in Figure 1.14 the matching phase can be split in different steps. The *Retrieve* step takes as input information extracted from sources, and transforms this knowledge in an internal ontology matching format, an example in ([46], [49], [94]), sometimes called reference model ([96], [97]). In its simpler form it is a list of terms representing semantics of input entities, and in other cases it can be a more complex Galois lattice representation like in [52]. Subsequently the *Match* step is able to execute similarity algorithms and *Formalizes* results with a correspondent confidence value for each match found. Some algorithms, like synonymy detection, can also require external resources (e.g.: WordNet [59] or electronic dictionaries). Thresholds and some heuristic are used in the *Prune* step to filter sets of matches. Techniques for matching sources are really numbness and the survey published in [95] is a good reference for discover and compare them.

1.4 Conclusion

In this first Chapter we have introduced the problem of leveraging the human bottleneck to the growing necessity of more reactive knowledge management for enterprise applications. For this we have presented the Semantic Web approach to knowledge representation which is based on ontologies.

Among the different languages available nowadays we have suggested OWL as recommended formalism to follow.

After the introduction of Description Logic and OWL that are necessary to the understanding of our work, we have studied existing systems aiming the automation of ontology generation. This is motivated by two reasons. One is that the construction of ontologies brings a new level of complexity that might be facilitated by automating the great part of the generation process. Secondly the enterprise environment already offers a huge quantity of formalized knowledge that cannot be ignored and completely rewritten starting from scratch.

Throughout our analysis we have seen that for our requirements, systems adopting a framework approach with the integration of an intermediary conceptual model better perform the automation of the ontology generation. Furthermore all over the analysis we have also shown that the extraction of ontological knowledge from XML sources is viable. But one problem is that few systems are available and for us this is an important lack to overcome.

Afterward we have also made a focus over the matching problem showing that it is probably the most notable research challenge to overcome if we want to automate the process. Nevertheless already exist numerous of on going works on this topic that seems acquiring interesting results. Consequently we do not cover specifically this topic and we focus our research on a system that aims the improvement of matchers furnishing them valuable information to perform their task better.

Least but not last in this chapter we have presented our vision on the ontology generation life cycle that also represent our overall approach that we will follow in our research to achieve the automatic generation system.

Chapter 2.

The B2B Domain:

Approaches and Limitations

In this Chapter we introduce the domain of electronic professional exchanges and in particular the B2B (Business to Business), which is the original starting point of our research. We present current approaches to professional exchanges between businesses with a particular focus on their current limitations concerning data flow interchange.

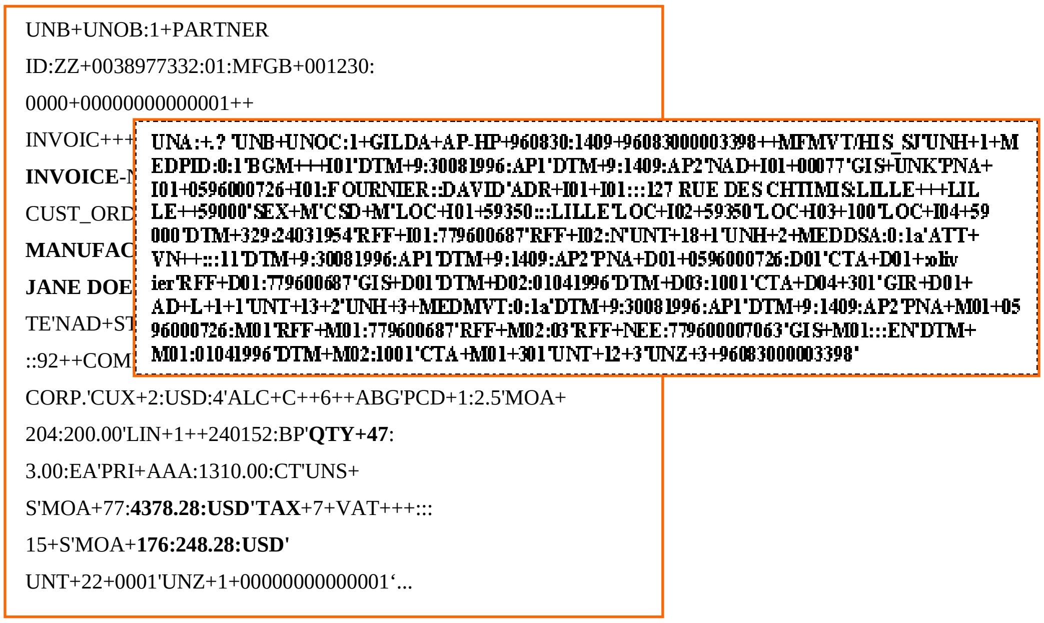
Following the SOA (Service Oriented Architecture) and SaaS (Software as a Service) paradigms, businesses are changing the way they collaborate with their partners, and consequently the requirements of enterprise applications are also changing. As we will show, among different problems present in the B2B architecture, the automation of business messages translation is one of the issues that can highly facilitate setting up dynamic electronic business relationships.

Currently most professional exchange integration scenarios are based on the complete transformation of business messages at design time following standardization approaches. Although this model works and businesses are able to exchange messages electronically, the effort to produce these standards appears too high and inadequate for more sporadic collaborations or for (smaller) firms that are unable to contribute to standardization. We claim that Semantic Web technologies are well suited to integrate the B2B architecture in order to fulfil the standardization approach and achieve the needed flexibility.

In Section 0 we provide an overview of the domain of professional electronic exchanges, restrained to B2B, with a short analysis of current practices, focusing on some of their weaknesses. We evaluate possible solutions to manage a more dynamic environment. Section 2.2 summarizes the main reasons that brought us to the decision to use Semantic Web technologies to simplify the setup of new business collaborations, and we add some new requirements that specific B2B ontologies should follow. Section 2.3 presents some relevant and already existing ontologies for the domain and Section 2.4 is a conclusion. One-Minute Electronic Professional Exchanges

When conducting a business relationship with its partners, any company, regardless of its size, seeks to increase its operational efficiency by improving the business processes and lowering costs. One way of reaching this goal is to automate the business processes to gain time and to reduce human intervention, therefore errors. Of course this applies to the operations performed both internally (inside the company) and externally (with other partners).

Since the 1960s, an important effort has been made to try to define standard data formats so that business partners could exchange structured business data via automated means, i.e. directly between computer-supported business applications [98]. Over the years numerous Electronic Data Interchange (EDI) [99] standards have been defined to enable interoperability. However traditional EDI suffers from barriers such as development and utilisation cost, long standardisation processes and critical user mass [100]. As a result, most of the EDI implementations that have been successful only apply to long term partnerships with high volume exchanges, and tend to involve only large companies.



```
UNB+UNOB:1+PARTNER
ID:ZZ+0038977332:01:MFGB+001230:
0000+00000000000001++
INVOIC+++
INVOICE-N
CUST_ORD
MANUFAC
JANE DOE
TE'NAD+ST
::92++COM
CORP.'CUX+2:USD:4'ALC+C++6++ABG'PCD+1:2.5'MOA+
204:200.00'LIN+1++240152:BP'QTY+47:
3.00:EA'PRI+AAA:1310.00:CT'UNS+
S'MOA+77:4378.28:USD'TAX+7+VAT+++:::
15+S'MOA+176:248.28:USD'
UNT+22+0001'UNZ+1+00000000000001‘...
```

```
UNA:+.? 'UNB+UNOC:1+GILDA+AP-HP+960830:1409+96083000003398++MFMVT/HIS_SJ'UNH+1+M
EDPID:0:1'BGM+++I01'DTM+9:30081996:AP1'DTM+9:1409:AP2'NAD+I01+00077'GIS+UNK'PNA+
I01+0596000726+I01:FOURNIER::DAVID'ADR+I01+I01:::127 RUE DES CHTIMIS:LILLE+++LIL
LE++59000'SEX+M'CSD+M'LOC+I01+59350:::LILLE'LOC+I02+59350'LOC+I03+100'LOC+I04+59
000'DTM+329:24031954'RFF+I01:779600687'RFF+I02:N'UNT+18+1'UNH+2+MEDDSA:0:1a'ATT+
VN++:::11'DTM+9:30081996:AP1'DTM+9:1409:AP2'PNA+D01+0596000726:D01'CTA+D01+oliv
ier'RFF+D01:779600687'GIS+D01'DTM+D02:01041996'DTM+D03:1001'CTA+D04+301'GIR+D01+
AD+L+1+1'UNT+13+2'UNH+3+MEDMVT:0:1a'DTM+9:30081996:AP1'DTM+9:1409:AP2'PNA+M01+05
96000726:M01'RFF+M01:779600687'RFF+M02:03'RFF+NEE:779600007063'GIS+M01:::EN'DTM+
M01:01041996'DTM+M02:1001'CTA+M01+301'UNT+12+3'UNZ+3+96083000003398'
```

Figure 2.1 – Example of EDIFACT invoice in use since '90

In order to provide a better comprehension of incoming difficulties when setting up business exchanges a new notion defined in 2004 by the Open-edi Reference Model [101] was been introduced. In their model a business collaboration is divided into two distinct phases: the *design time* phase[12] during which business requirements of the message exchanges are defined, and the *run time* phase[13]

[12] Design time covers all the necessary tasks for modeling and for setting up the execution of B2B collaborations. This phase involves the business process specification, the partner profile definition, the trading partner contract establishment, the business document conception and the message exchanges integration (or mapping) to the existing information system. Design time also includes the discovery and retrieval of existing business data.

[13] Run time covers the real execution of business exchanges from beginning to their termination. (i.e., business processes execution, messages exchange and dynamic services discovery).

which executes the business process through collaborating application systems. This distinction provides a key lecture of EDI implementations: they perform well during run-time phase at the cost of a much heavier design-time phase.

In the mid 1990s, the advent of Internet and its related technologies has lowered connection barriers between enterprise information systems (IS) by reducing the EDI set-up and operational costs, while adding greater accessibility. In the meantime, the eXtensible Markup Language (XML) [102] has provided a simpler and more flexible formal language that highly contributed to the reduction of development complexity at content integration and definition level, performed at design-time. Just as an example Figure 2.1 shows an excerpt of an EDI standard message that is in use since '90 (a more recent example based on XML is shown later in the document, see Figure 4.1). It clearly shows how this first business message format was meant for machines, and difficult to read for a human. The setup of common business data was therefore more difficult to handle before the introduction of XML. Finally these two elements provided a new way of doing business between companies that since 2000 is commonly referred to as business-to-business electronic commerce[14].

Nevertheless it is largely recognized that the complexity when setting up a new collaboration is still far from solved, and difficulties in defining the necessary business data still remains. One reason is that not only technologies evolve. It is also the case for needs and business collaborations. More messages arise and thus new requirements come up. As seen above, the *design time* phase needed to set up new business collaborations includes several tasks[12] that are at this time still performed manually or in an *ad hoc* manner, more often using UML tools or XML editors with a limited possibility to discover and reuse other business data. Therefore this process remains very long, complicated, and somewhat arbitrary. One consequence is that even if we are able to *physically* connect two enterprises information systems, the data integration problem still remains.

During the last few years more and more initiatives studying the integration of enterprises applications target the development and sharing of business data. This is the case for several governmental institutions, standardization organizations, large companies or consortia that look for efficient solutions to define and publish business exchange requirements. Such solutions are considered fundamental to increase visibility and availability of information exchanged among businesses.

However all these efforts fall within the *design time* phase. In order to give an idea and a measure showing the complexity of the task, we can cite the TIC-PME 2010[15] initiative. This initiative is a 3 years and 10M€ program promoted by the French government that aims to improve SME (Small and

[14] Even though in this document we tend to use B2B as term to describe the environment of our research, electronic message exchanges are not limited to businesses. Administrations are increasingly confronted with similar problems in their relationships with companies or other administration departments: they need to provide high quality services to a wide audience, targeting both private and public sectors, while improving their efficiency and reducing their costs. Even internally, companies need dynamic message exchange solutions.

[15] http://www.telecom.gouv.fr/tic-pme2010

Medium Enterprises) profitability and competitiveness regarding the market. The approach is almost sector strategy and involves particularly the harmonisation of the exchange model used by the actors of the sector (business area). The community leaders model (for instance Renault, Airbus, Carrefour,…) is connected to the other main companies' model, within a given service sector, subcontractors included. With this initiative the government provides substantial *design time* input to businesses to define requirements to electronic exchange execution. This is not the first and only initiative focusing the problem, we can also cite BoostAero [16] (International Associations for Aerospace & Defence), Etso[17] (Electricity sectors) and so on. We believe that all these initiatives are representative of the complexity of the problem. A lot of effort is spent on providing a common harmonized base of business data, but within an evolving, Web-enabled environment, producing static knowledge formalization could rapidly turn out to be obsolete.

For this reason we aim to analyse in this thesis new solutions that can improve dynamicity aspects of B2B domain and support some kind of automation.

2.1.1 B2B Overall Architecture

Without delving into Enterprise Applications Integration (EAI) [103] solutions, Figure 2.2 presents a high level view of the main pieces of software required by enterprise ISs[18] from the B2B business data [19] perspective. It provides the underlying IS architecture to operate a complete electronic transaction, where modules are specifically defined to group business data with a common target.

Firstly we divide modules between the *internal stack* and *external connection modules*. This division differentiates modules between the *closed world*, internal to the company thus normally more controllable, and the *open* one, open to others partners on which it is difficult to make an *a-priori* forecast concerning adopted solutions. The organization of business data for the internal stack of a company depends on several factors, mainly the size of the company, its organization and the IS software used (e.g. a complete SAP system or a lightweight ERP[20]). Since outside relationships are

[16] http://www.boostaero.com

[17] http://www.etso-net.org

[18] According to [114], we define an IS as an application or enterprise system that provides the information infrastructure for an enterprise. Typically, an IS consists of one or more applications deployed on an enterprise system. An IS provides a set of services to its users. Example of enterprise applications are Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), sales, accounting and messaging system.

[19] We use the general term business data meaning a formal description at concept level of a piece of information necessary to set up and operate an electronic business collaboration. Examples of information that must be defined are: business process, business document, message content, message protocol, electronic service description, catalogue, electronic signature, trading profile and trading agreement (an extensive description of the last two business data formalizations can be found in the OASIS CPP/CPA standard specification [115]).

[20] Enterprise Resource Planning is a company-wide computer software system used to manage and coordinate all the resources, information, and functions of a business from shared data stores.

open to every possible technical solution, the external connection system must be able to handle different ways of conducting electronic exchanges and managing new needs that might arise from a new business collaboration. The number of existing solutions covering B2B requirements is huge, therefore we organize the architecture into five main elements: message package, network protocol, security constraints, business process management and data format.

IS generally includes several applications, e.g. handling payroll processing, inventory management, manufacturing production control, and financial accounting. Even though the problem of data integration can subsist in large enterprises, for example when updating or adding a software element or when two enterprises merge, usually all these elements are integrated using ad-hoc layers for data flow or by sharing the same data table, e.g. by using a SAP[21] system. In the following we consider the simpler case where a company has a single business software solution that provides a unique user interface for all applications, and we will focus only on the external connection B2B elements.

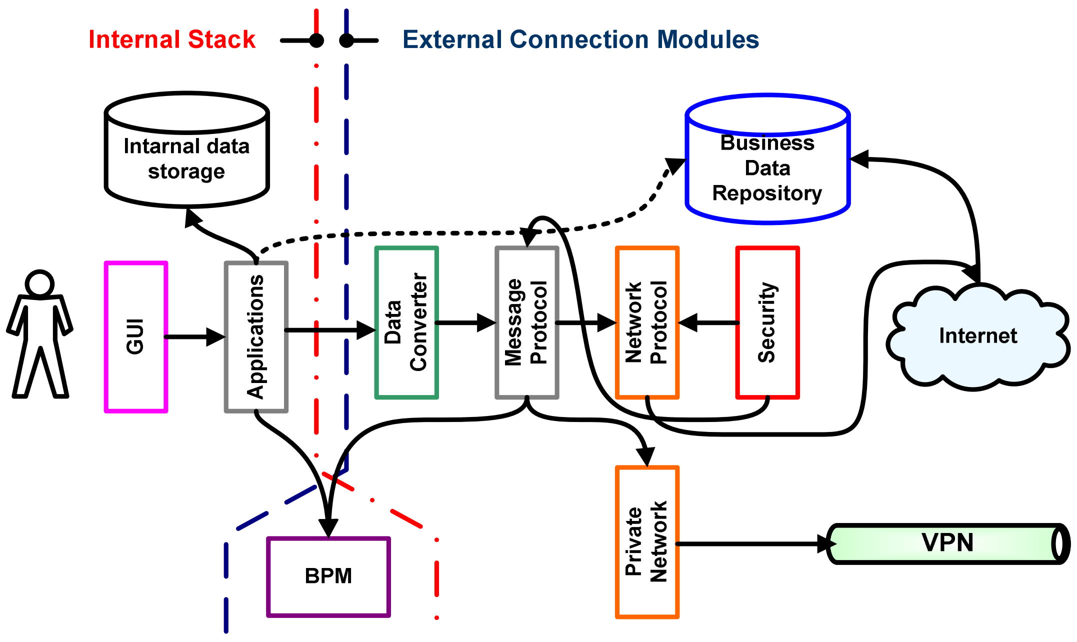


Figure 2.2 – Main elements of an electronic business exchange

2.1.1.1 Message Protocol

The **message protocol** module is needed to define the package, envelop, into which the information is enclosed and exchanged between partners. The list of acronyms and different standards defining this layer is large: we can cite *Applicability Statement* (like AS2), *ebXML Messaging System* (ebMS), *Web Services* based solutions (WS-*) and *Message Queue* (MQ) solutions. All these formats provide message exchange error handling and reliability over the IP network. Figure 2.3 illustrates an example

[21] http://www.sap.com

of the ebMS [104] package containing business data for the message protocol module. The real business message within this envelop, e.g. an invoice, is enclosed as a payload.

2.1.1.2 Network Protocol

The **network protocol** module defines what kind of protocol is used as communication layer, like HTTP/S, FTP, AFTP, P2P, with endpoint to access the physical address of partners' message-boxes and machine services and other specific data required by the adopted protocol.

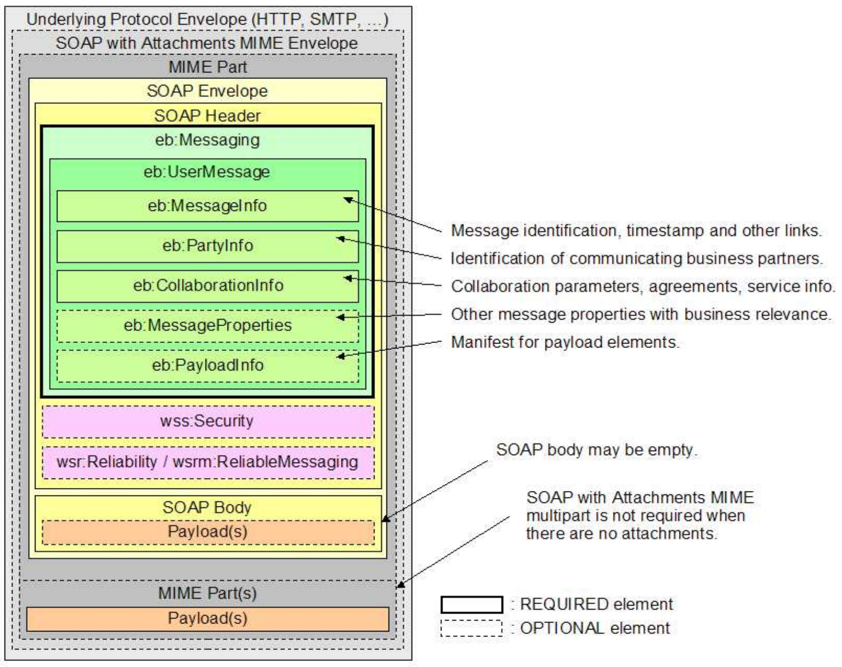

Underlying Protocol Envelope (HTTP, SMTP, ...)
SOAP with Attachments MIME Envelope
MIME Part
SOAP Envelope
SOAP Header
eb:Messaging
eb:UserMessage
eb:MessageInfo
eb:PartyInfo
eb:CollaborationInfo
eb:MessageProperties
eb:PayloadInfo
wss:Security
wsr:Reliability / wsrm:ReliableMessaging
SOAP Body
Payload(s)
MIME Part(s)
Payload(s)
Message identification, timestamp and other links.
Identification of communicating business partners.
Collaboration parameters, agreements, service info.
Other message properties with business relevance.
Manifest for payload elements.
SOAP body may be empty.
SOAP with Attachments MIME multipart is not required when there are no attachments.
: REQUIRED element
: OPTIONAL element

Figure 2.3 – General structure and composition of an ebMS User Message

2.1.1.3 Security Module

The **security** module provides all detailed information about the realization of security concerns. Business data defines encryption details, digital certificate and electronic signature to be used, and of course the signatures themselves.

2.1.1.4 Business Process Management Module

The **business process management** (BPM) module handles the execution of a business process, which is an ordered sequence of either human or machine tasks that perform a business objective, in practice they define the "who, what and where" tasks to be accomplished. Figure 2.4 illustrates an example of the description of the business process "drop ship multiparty collaboration". It represents a typical B2B business process where several partners are involved in a transaction. Furthermore each task can be linked to an internal process of the company. Different standards are available formalizing a business process language definition, e.g. XML Process Definition Language (XPDL) [105], Business Process Modelling Language (BPML) [106][107], Business Process Schema Specification (BPSS) [108], and Business Process Execution Language for Web Services (BPEL4WS) [109][110][22].

Each of them provides different standpoint of a business process definition, sometimes internal to the enterprise and sometime from the outside point of view. This differentiation is mainly due to the fact that B2B positions are often multiparty, like that one presented in Figure 2.4. This explains why this module is shared between internal and external world as illustrated in Figure 2.2.

Business data managed in this part represents an interesting use case where Semantic Web technology might add value and flexibility to business exchanges. Indeed we can have different formal representations of the same basic process. As such the definition of common business processes is often a strong task involving large amount of human works.

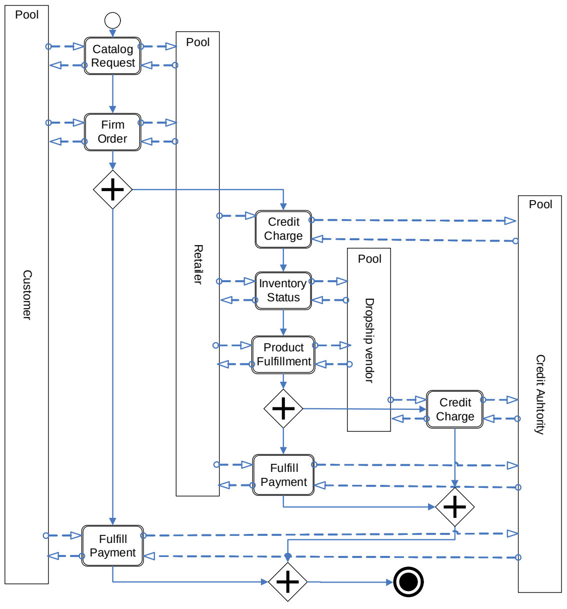


Figure 2.4 – Representation of the "DropShip" Multiparty Collaboration

2.1.1.5 Data Converter Module

The **data format converter** module is a specialized software layer that provides the transformation of external data to the internal format. In practice it maps the external messages format to an understandable internal format in order to be interpretable by enterprise applications. At this level business data consists of the formalization of the pieces of information that are involved in a business transaction. These can be business documents such as an invoice, medical record, contract or CV, or simpler messages like the response to a specific request such as the availability of a product, hotel reservation, employment history or flight details.

Figure 2.5 illustrates a typical B2B scenario where this kind of module finds all its usefulness. As we will see in Section 2.1.2 most B2B exchanges implement standard interfaces, which means that

[22] Some interesting comparisons about these different formalizations can be found in [111] and [112]

businesses often have a pre-defined mapping from their *internal interface* data format (provided by the IS application) to a B2B standard. Since the latter is more general and adaptable, it is used to provide the final data flow between the company and its partners. The picture shows the complete data flow performing messages exchanges between two generic companies, for instance one as a buyer and the second as its supplier, using two different standards.

Moreover a business message is often referred as **business document** that similarly to [116] we define as a formalized aggregation of more specific business components. Where a **business component,** or **core component,** is a building block for the creation of semantically correct and meaningful piece of information necessary to describe a specific concept. In UML notation a business component can be represented as a class, while in OWL can be a *NamedClass*.

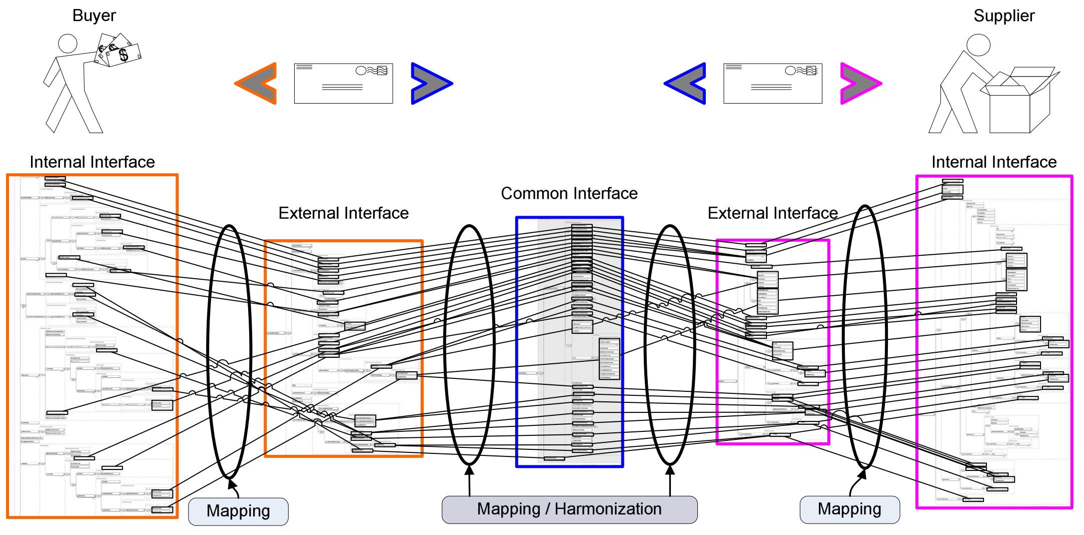


Figure 2.5 – Typical representation of B2B message transformation scenario

2.1.1.6 Business Data Presentation Module

Finally we introduce the **business data presentation** module as new element of the architecture for the external connection. This block is an advanced enterprise repository that contains all the defined business data and presents them publicly accessible on the Web in a formal and expressive representation language, making process information formally explicit and machine understandable. This is an essential prerequisite to facilitate business exchanges. In fact as argued previously one limitation to dynamic collaborations is the complexity of the *design time* phase and this is also due to the fact that these pieces of information are rarely accessible. When such information actually exists, it is mostly of informal nature and provides only text descriptions and graphics depicting some models. As a consequence, the query answering capabilities of such a repository are very limited and thus discovery and automation remain difficult to operate.

Few specific software exists to build this module. A first standard specification of this element can be found in the UDDI registry [117], but it is restrained to the publication of Web Services descriptions (WSDL files). Another is the ebXML Registry-Repository [118] [119] that provides a larger possibility for publication and sharing of business data, but as far as we know its' adoption is still limited. The former is already integrated in several production solutions, but it is limited to WSDL business data, thus it is not adequate to fulfil all this module's tasks. The latter is currently the subject of several initiatives aiming for the publication of business documents structure and semantics, such as the European SEMIC.EU Repository[23] [120], the UN/CEFACT Registry Implementation [121] and the experimental EDIFRANCE RepXML project [122]. These are mostly either governmental or Standardization initiatives and we are unaware of their adoption in an enterprise context.

2.1.1.7 Discussion

Business data from the message protocol, network protocol and security modules are required to create the physical connection between IS and from the information definition standpoint they are not the major cause of B2B complexity. In fact more and more available commercial systems are capable of handling several technical solutions at once and provide run time protocol transformation. Without any surprise modules managing business documents and business processes remains the hardest obstacle to flexible and dynamic B2B interoperability. It is caused by several factors, like the fact that often semantics follow cultural and usage constraints. In any case this has lead to a large heterogeneous design of business data that does not make viable the generation of an automatic data translator, but still we need a lot of human work for this. In such context business data needs of a more semantic framework. It could provide a key expressivity to machines and improve the automation.

Another element necessary to the automation is the business data presentation module. More data are available and formalized in a machine readable language and better is the discovery and reuse of existing practices. Indeed in origin it was our targeted research topic with a dedicated semantic Information Content Management for enterprise, a kind of semantic repository of business data. Finally we observed that still few real semantic information is available (as we will see in Section 2.3), consequently we opted to work on a system facilitating the set up of semantically annotated documents and knowledge representation improving messages matching.

2.1.2 Approaches to Business Document Design

As seen above setting up new business collaborations requires a lot of effort during design time phase in order to define requirements and business data. In this section we analyse current approaches to the design of business documents needed to implement messages exchanges between companies. For this we divide B2B exchanges into three main approaches: the *ad-hoc or point-to-point* approach, the *proprietary data model* approach, and *adoption of standards* approach.

In the **ad-hoc or point-to-point approach** business documents are defined multilaterally during the design time phase of the business collaboration. This system shows some kind of "flexibility", in the sense that it does not present specific constraints because every time a new design is made. This

[23] http://www.semic.eu/semic/view/snav/Repository.xhtml

flexibility on the other hand clearly shows a low degree of reusability and integration with new partners. Figure 2.6 depicts a simple example of such an approach applied to two businesses, while they each have their own internal data representation, messages exchanges are formalized in a common format that has been defined a priori, then each party develops a mapping layer on top of their internal application in order to integrate information and related actions. The right hand side picture in Figure 2.6 highlights what happens when a company has more business relationships to set up. The number of connections needed to have a fully meshed point-to-point connections between *n* companies is *n(n-1)/2*. I.E. for 10 applications to be fully integrated point-to-point, 45 point-to-point connections are needed.

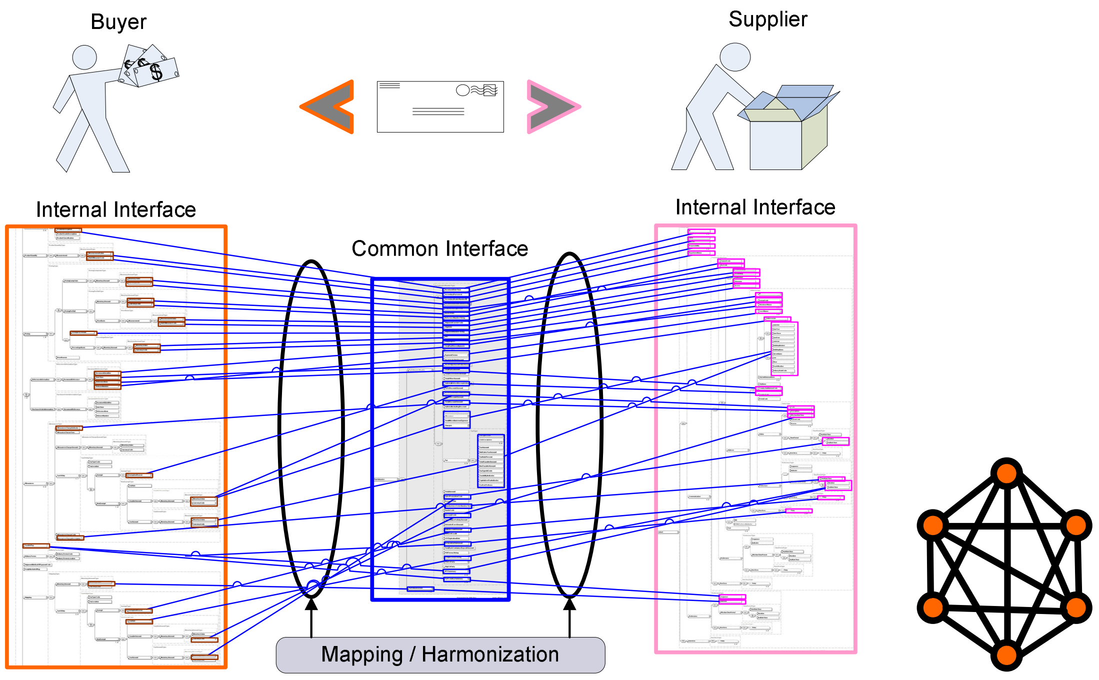


Figure 2.6 – Message content definition in ad hoc solution

The **Proprietary data model approach** business documents is decided unilaterally. Typically this approach covers business collaborations with a main contractor in cooperation with small businesses, such as a big retail group and its suppliers. In this case it is simpler for the big company to take entire charge of the business requirements design, trying to adopt the larger predictable requirement, because it often has the more complex system to manage and to make interoperable with internal processes, while a little company uses a smaller IS, thus more flexible. Setting up such a solution is faster and does not require the complex harmonization phase, but on the other hand partners who do not adopt the same solution are forced to develop a new application layer to join the business collaboration. Figure 2.7 depicts this business collaboration pattern, while the picture aside draws attention to the fact that there is a party that is forced to produce mappings and application layers for each new collaboration.

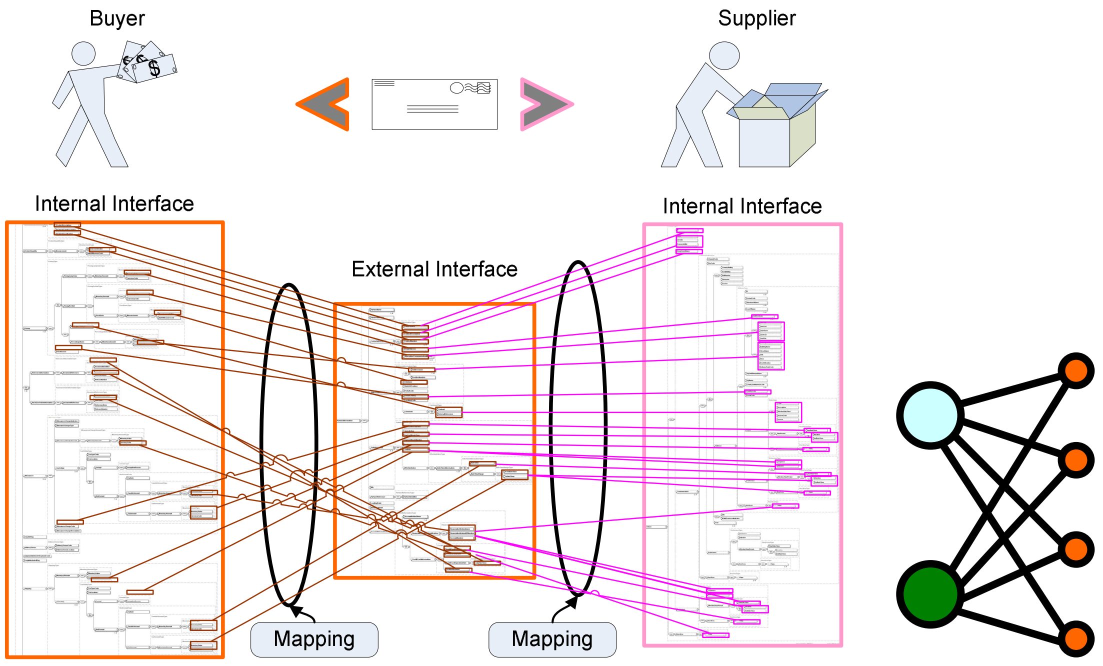


Figure 2.7 – Message content definition according a proprietary solution

In the **Adoption of standards approach** (mutualisation) business requirements are provided by a collegial work defined in a specific consortium. The realization is a common preliminary effort that involves several parties, mainly experts of the specific process and/or the whole domain. It has the advantage of being a standard and thus of guaranteeing a certain level of compatibility, durability and reuse of past experiences and knowledge. The resulting definition of business data is a static knowledge representation that can be changed only with further common effort. Negative points are that it is based on standards, so it requires a tremendous standardization effort and moreover, as shown in Section 2.1.4 quite often several standards coexist in the same sector, which implies the need to implement multiple standards. Figure 2.8 (a) illustrates how this business exchange pattern centralises efforts and makes this approach more profitable with respect to others, but it is so only in a theoretical perspective because as Figure 2.8 (b) shows, it can become complex when more standards come into the arena.

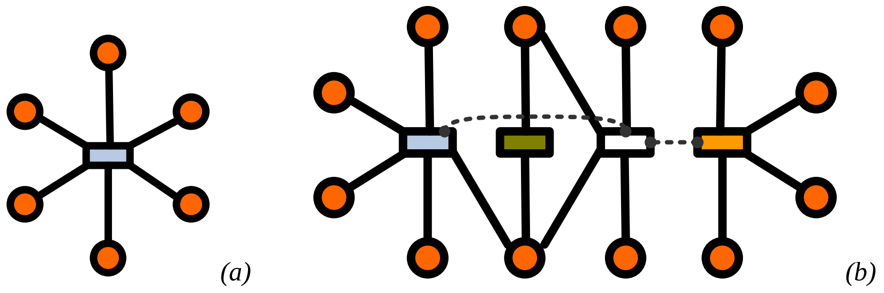


Figure 2.8 – Message content definition adopting standards

As shown in the European e-business report (E-Business W@tch, 2007 [123]) between these patterns at least three enterprises out of four that realize business exchanges with partners, declare implementing applications based on B2B standards solutions (in Europe). While this is surely a good way to reduce interoperability problems and to benefit from world wide experiences, it is **hopeless to standardize any possible business collaboration**. It also implies that business partners must first find already defined business data based on these standards that best match their needs. Only then can start the collaborative design using these models. We stress that the problem of finding, reusing, harmonizing and adapting the different standard components is not trivial: until now it has been common practice, including among standardization organizations, to simply publish business data on a web page in directories or even in flat files! However discovery and adaptation are tedious and take a lot of time, since browsing through search results must be performed manually. For all these reasons we conclude that these approaches are not able to cover and satisfy all the B2B requirements yet.

2.1.3 The Deterministic Method

Current methods of business collaborations and relative architectures exhibit a common characteristic of business data design: *they are always pre-formatted to strict and precise structures and semantics*. These methods have the advantage of allowing error-safe execution management but to the detriment of a strong initial effort. We define this approach as the ***deterministic method***. Although no module exists yet to resolve ambiguous situations due to similar, though different design.

Since the Semantic Web Vision [2] is all about machines being able to locate and process information on the World Wide Web without the need for human intervention, the next step to transform a deterministic method to a more semantic method, thus more dynamic and automated, should be the adoption of semantic related technologies. However the Semantic Web approach is still far from a complete automation of B2B message exchange directly at run time. In fact at present no automatic ontology matching/mapping system is able to guarantee a perfect and total error-safe result (and probably there never will exist such a system at all) and induced inferences could be wrong. Consequences of such errors can be evaluated on simple transactions, but it is difficult to evaluate real consequences in a complex process. So dealing with a complete automation of business exchanges, with dynamic set up at execution time should imply the integration of architectural modules able to provide more efficient matching/mapping algorithms, a more complex management of exceptions, agreement establishment and execution rollback in the case of such errors. This is the gap that still needs to be breached but as seen in Section 1.3 current research about ontology alignment focuses on quality results rather than computational time efficiency, and moreover as seen in Section 2.1.1 modules in the B2B architecture are not able to handle flexible data integration. We argue here that providing a complete system requires a lot of further research work. In this thesis we only consider ontology generation, with a special focus on matching features and efficiency when introducing this new approach to B2B exchanges based on semantic technologies.

2.1.4 B2B Standards…

Since we target B2B exchanges it was essential to find an appropriate data source in order to test our approach, but as argued above enterprises do not currently publish their formal messages and business data, which made it difficult for us to produce real use cases. However as argued in the section above most enterprises perform business exchanges implementing applications based on B2B standards solutions. We have therefore based our tests on B2B standards and for this we investigated and processed more than 40 of them.

Table 2.1 presents a list of 37 e-business standards, mainly targeting the B2B area. The data provided by this set of standards is a considerable corpus that gives us a broad view about current practices. The table lists: the name of the standard body or consortium; column three lists the business areas that the standard covers; the alliances column informs about declared compatibility coalitions, already active or expected to come; the fourth column summarizes what kind of business content is produced by each standard body; the following column details the formalization of published standards; the standards' downloads column provides the information of their availability and adoption (public, under a payment, or only for member of the consortium); the last column just provide a link. The table does not say if the consortium also provides a specific implementation framework.

We have not inserted in this list the standard bodies that have been a priori excluded because they are designed for too specific use case. Examples of the overly specific working groups are: EDItEUR (the international group for electronic commerce in the book and serials sectors), BISG (Book Industry Study Group) and EPISTLE (the European Process Industries STEP Technical Liaison Executive), PRODML (Production Mark-up Language and WITSML (Wellsite Information Transfer Standard Mark-up Language).

A growing number of standard bodies are currently adopting the ebXML design as basis for their own standards and are aligning their business components to the Core Components Library (CCL). Between them we can cite: OASIS Universal Business Language (UBL), Open Applications Group (OAG), EAN-UCC, SWIFT, ANSI ASC X12 and CIDX.

ebXML is a joint effort of OASIS and UN/CEFACT that aims to develop a complete framework for e-business. The library is prevalently developed by the UN/CEFACT standard body that counts 15 specific working groups, each one representing a business area such as Supply Chain, Transport Domain, Customs, Finance, Construction, Insurance, Healthcare, Agriculture and e-Gov. Another specialised group provides a synchronization of the documentation and specifications proposed by each group. It finalizes the work with a harmonized library of the so called CCL, which are the basic components to build B2B messages. Others groups also define standard business processes and technical implementations. The CCL is drawn on the UN/CEFACT Core Component Technical Specification [139] that provides a simple and powerful UML based data model, to define reusable structure and semantic content of business messages.

Another set of standards that have not been included are more e-Commerce focused standards like eCl@ss [124] and UNSPSC [125] simply because even if related to the B2B domain, they do not provide messages specifications but merely a product classification.

As we can see lots of business data is defined by standard bodies: core components, whole messages, business processes, web service descriptions, code lists and EDIFACT messages. In this work, only core components, often called *Data Dictionary*, and messages have had our attention and were used to compose the test corpus. Our study shows that XML Schema is the most widely supported formalism adopted by consortiums and at present it is the *de-facto* standard document format. It has overtaken other formats like the "old" EDIFACT and, at least for the moment, the "new" RDF/OWL format. Only cXML[24] still provides only a DTD based standard, and *not a single* RDF/OWL format is officially produced by any consortium.

Concerning data presentation, almost all organizations provide a package containing several documents. It includes specifications, graphics, examples, guidelines, implementation tutorial and, what we are most interested in, XSD files. Generally XSD files are numerous, at least one for each specific business message, one for grouping common core components, others for grouping common data type definitions and code lists. Only few of them provide a specific repository with a detailed view and discovery system of data components. Once processed, our final corpus source is composed of a collection of 26 B2B standards, with more than 3000 XSD files. We feel that this is largely enough in order to have significant information about B2B business message description practices and semantics, and our results show that, at semantic level, past a given point, adding more standards into the process does not change much (shown in Chapter 4).

As final consideration about inventoried groups, we stress that this is by no means an exhaustive and complete list of all existing standard bodies and industrial consortium; other standards exists or new ones will be created in the future, nonetheless we consider our list as the most comprehensive with respect to others we have met up to now concerning the B2B domain.

[24] http://www.cxml.org/

	Standard Body		Business Area	Alliances	What	Published Formats	Standards Downloads	Web Site
1	ACORD	Association for Cooperative Operations Research and Development	Insurance, reinsurance and related financial service	ASC-X12, XBRL, HR-XML, eEG7, CSIO	Dictionary, messages	EDIFACT, XML Schema, WSDL	registration	www.acord.org
2	AdsML	Advertising Standards	Advertising, Graphics communication		Dictionary, messages	XML Schema	free	www.adsml.org
3	AgXML	Agricolture XML	Agriculture supply chain	ebXML, CIDX, RAPID	Dictionary, messages	XML Schema	membership fees	www.agxml.org
4	AIAG	Automotive Industry Action Group	Automotive industry				membership fees	www.aiag.org
5	ARTS	Association for Retail Technology Standards	Retail		Dictionary, Relational Data Model	XML Schema	payment (exept for schemas)	www.nrf-arts.org
6	ASC X12	The Accredited Standards Committee	Cross industry		Dictionary, messages, EDIfact messages, BP	EDI X12, XML Schema	registration	www.x12.org/
7	BMECat	Federal Association for Material Management, Purchasing and Logistics	Electronic		Dictionary, Classification schemas, Product Configuration, price formulas	XML Schema and DTD	registration	www.bmecat.org
8	ChemITC	American Chemistry Council's Chemical Information Technology Center	Chemical					www.americanchemistry.com/s_chemITC/
9	CIDX	Chemical Industry Data Exchange	Chemical	ebXML, RAPID, OAGi, ChemITC	Dictionary, Business Processes, WSDL, RFID codes, messages	XML Schema	free	www.cidx.org
10	CSIO	Centre for Studies in Insurance Operations	Insurance, reinsurance and related financial service					www.csio.com/
11	ebInterface		Invoice		Invoice Document	XML Schema	free	www.ebinterface.at/
12	EbIX	European forum for energy Business Information eXchange	Energy				free	www.ebix.org

13	ebXML	e-business XML	Multi area. 15 business area represented. One WG with harmonisation purposes and one for BP definition	ISO	Dictionary, Messages, code lists, EDIFACT, methodologies	XML Schema and UML, EDIFACT, Spreadsheet	free	www.unece.org/cefact
14	eEg7	E-business Standards for the European Insurance Industry	Insurance, reinsurance and related financial service					www.eeg7.org
15	Energistics		Energy		Dictionary		registration	www.energistics.org
16	ETSO	European Transmission System Operators	Specific electric transaction	ebXML	Dictionary	XML Schema	free	www.etso-net.org
17	FIX	Financial Information eXchange	Banks, broker-dealers, exchanges and institutional investors	SWIFT (ISO 20022), FpML	Framework with message protocol, message definition, codes and Dictionary	XML Schema	registration	fixprotocol.org
18	FpML	Financial Product Markup Language	Financial	FIX, FIXML	Dictionary, Business Processes, architecture	XML Based	registration	www.fpml.org
19	GS1	Global Standards	Supply chain for Healthcare, Defence, Transport & Logistics	ebXML	Dictionary, Business Processes, Messages, SOAP Messages…	XML Based	free	www.gs1.org
20	HL7	Health Level 7	Health				free	www.hl7.org
21	HR-XML	Human Resources XML	Human Resource	ACORD	Dictionary	XML Schema	free	www.hr-xml.org
22	IFX	Interactive Financial eXchange (IFX) Forum	Financial		Dictionary, Messages, Web Services	XML Schema, WSDL	registration	www.ifxforum.org
23	ISO 20022	ISO 20022 Universal financial industry message scheme	Financial	IFX, OAGi, TWIST	Dictionary	XML Schema, UML	payment	www.iso20022.org
24	MDDL	Market Data Definition Language	Financial		Specific XML framework		registration	www.mddl.org
25	MISMO	Mortgage Industry Standards Maintenance Organization	Residential, commercial, eMortgage	IFX, ACORD, ASC X12	Dictionary	XML Schema	free	www.mismo.org
26	NAESB	North American Energy Standards Board	Energy (Gas, electric)				membership fees	www.naesb.org

27	OAGi	Open Application Group integration Standard	Cross industry	ebXML	Dictionary, Web Services, Messages	XML Schema, WSDL	registration	oagi.org
28	Odette		Automotive industry				membership fees	www.odette.org
29	OTA	Open Travel Alliance	Turist		Dictionary, codes, messages	XML, Spreadsheet	registration	www.opentravel.org
30	PapiNet	Paper Industry Network	Paper Industry		Dictionary, messages	XML Schema	free	www.papinet.org
31	PIDX	Petroleum Industry Data Exchange	Energy (petroleum industry)	ebXML	Dictionary, Web Services, Bar codes, EDI messages, Business Process	XML, WSDL, EDIFACT	free	www.pidx.org
32	RAPID		Agricolture	CIDX	Dictionary, Messages, Code lists, Bar codes	XML Schema, EDIFACT	free	www.rapidnet.org
33	RosettaNet		Supply Chain Management, IT, Telecommunication	GS1 US, ebXML	Dictionary, Business Processes	DTD, EDIFACT, XML Schema	registration	www.rosettanet.org
34	STAR	Standards for Technology in Automotive Retail	Automotive industry	OAGi, ebXML	Dictionary, messages, Web Services	XML Schema, UML, WSDL	free	www.starstandard.org
35	TWIST	Transaction Workflow Innovation Standards Team	Supply chain, payment	FpML, FIX, SWIFT	Dictionary, Business Process	XML Schema	free	www.twiststandards.org/
36	UBL	Universal Business Language	Invoicing, ordering	ebXML	Dictionary, messages, Business Processes	XML Schema, UML, ebBP	free	www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl
37	XBRL	eXtensible Business Reporting Language	Reporting, accounting	UN/CEFACT, CIDX	Dictionary, messages, formulas	XML	free	www.xbrl.org

Table 2.1 – B2B Standards

2.2 Why create a B2B ontology?

It is known that adding new tools adds new complexities and new learning curves, so there needs to be a concrete business benefit to justify the cost of implementation. Throughout this section we argue why ontologies should be introduced in the B2B domain.

Firstly we observe that B2B provides an interesting use case for semantic applications because by its nature it illustrates the problem of different designs and ways of structuring the same set of concepts producing data heterogeneity problems. The deterministic approach (see Section 2.1.3) prevents from any possible automation of data interpretation because machines are only called to execute code and no data description is available for handling reasoning and inferences at run time, even for simple mismatches. This is the consequence of an approach completely designed for human understanding. Reasoning on this kind of data is impossible because of the intrinsic limits of its definition.

B2B applications are implemented by interfaces based on standard messages defined by several consortiums and it appears that standardization organizations are often organized by business area. To create electronic connections with different industry partners, a new application layer and a new design are needed every time a new partner joins the collaboration on the fly, with the objective of integrating information describing the same set of concepts. Moreover even when solutions are based on the same upper standard, direct compatibility is un-guaranteed as shown in [38] and [126].

How can we conjugate dissimilarities of semantics, information details, structure and also cultural approaches in a comprehensive model? How can machines communicate between themselves reducing human effort?

As we already mentioned above the Semantic Web, and particularly ontologies, seem to achieve good results within the last years. Several people have addressed the specific adoption of such technologies for the e-business domain. Dieter Fensel in his book, *Ontologies: Silver bullet for knowledge management and electronic commerce* [23], outlines the key differences between ontologies and databases schemas which are more close to a “physical data model”. Moreover he argues that the language for defining ontologies is syntactically and semantically richer, by its own nature the ontology requires a consensus among several parties and as such it is more similar to a domain theory rather than a data container.

The document *Best Practices and Guidelines* [127] focuses on applications of Semantic Web for electronic commerce on the Internet, and defines a specific list of potential benefits from its adoption. Like the development of efficient and profitable Internet solutions, a meaningfully share of information, that provide a good base to argue the benefit of the integration of semantic technologies. At the same time, the authors identify critical issues and research priorities to transform these potentials into real benefits.

In the paper *Potential Advantages of Semantic Web for Internet Commerce* [130], and in [131], Zhao Yuxiao *et al.* provide a comprehensive list of twelve points on the potential benefits of adopting

Semantic Web e-commerce. Among these twelve categories we can see a possible improvement in the integration of applications, information management, filtering of information, the composition of complex systems, a more flexible standard vocabulary, and what he defines Serendipity (unexpected benefits).

Antony B. Coates in his talk [132] is more pessimistic and argues that the Semantic Web Vision [2] still remains a long term goal, and this is the reason why businesses and standard bodies still hesitate to introduce it. However he adds some factual reasons linked to the limitations of current data models and how ontologies can already improve them in the short term. For instance the UML (Unified Modelling Language) is the most widely used modelling technique in the domain. Indeed UML is intended as general modelling approach because it not only proposes data modelling, but also use cases, process flows, state diagrams and also has an XML interchange format (XMI). However the interchange format has numerous versions and different tools either use different versions, or use the same version in different ways (too much flexibility in the format?) so real interoperability is poor. Another relevant limitation of UML is that for object-oriented reasons in some cases it requires adding extra classes, which is fine for technical users but it is irrelevant and unnecessary in a model designed to be used by business experts. This makes diagrams more complex and confusing than they need to be. Take as an example, illustrated in Figure 2.9, an intended business model like "vendor sells to company or government", where UML forces the creation of common "purchaser" parent class.

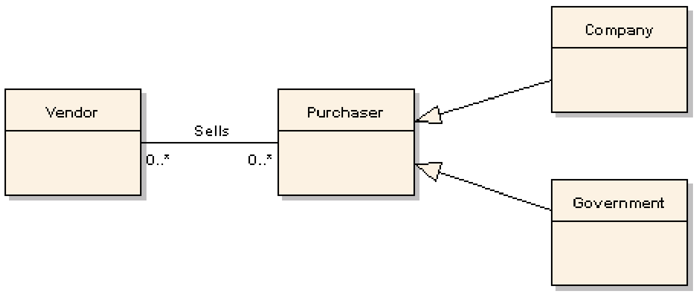


Figure 2.9 – Example of UML class diagram

OWL adds simplicity, when representing the same model, and allows us to say that a Vendor sells to a "Company or Government", without introducing a named parent class, as illustrated in Figure 2.10.

Also the UML tools' support for objects/instances (e.g. "a particular car, a particular person") is much weaker than RDF/OWL tools, and not really usable for constructing business context models referencing particular countries, business areas, etc. Moreover when merging models, RDF/OWL assertions are preserved and also enable detection of inconsistencies, while the UML merging operation is completely a human task.

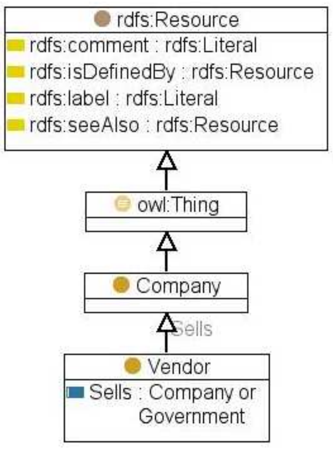

rdfs:Resource
rdfs:comment : rdfs:Literal
rdfs:isDefinedBy : rdfs:Resource
rdfs:label : rdfs:Literal
rdfs:seeAlso : rdfs:Resource
owl:Thing
Company
Sells
Vendor
Sells : Company or Government

Figure 2.10 – OWL modellisation example (the same that Figure 2.9)

In [38] Anicic *et al.* define an architecture (see Figure 2.11) based on Semantic Web technologies to investigate the enterprise application integration (EAI). As an example both enterprise applications implement two correlated but independent standards for messages exchanges. One is Standards in Automotive Retail (STAR) and the second is the Automotive Industry Action Group (AIAG) and both base their interface on a more "horizontal" standard defined by the Open Application Group (OAG).

Their study shows that ontologies and reasoners improve the integration of message exchanges between companies. Conversely, in their implementation the integration still requires human intervention, since identification and resolution of semantic and syntactic similarities, is done by hand.

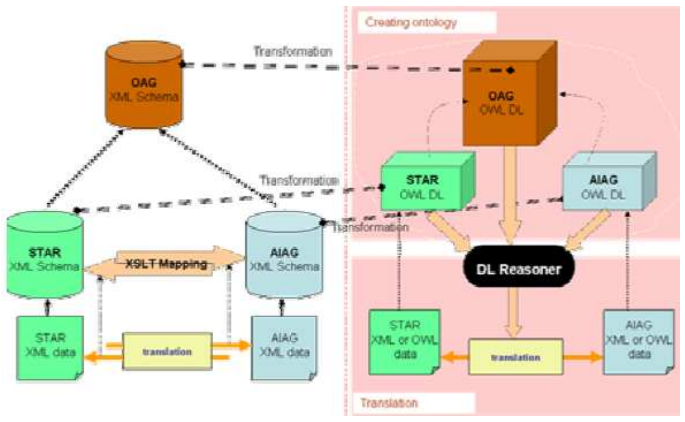


Figure 2.11 – Traditional and Semantic Web-based EAI Standards Architectures

This experience improves the data converter module presented in the B2B architecture (see Section 2.1.1.5) and by doing so, interoperability problems between worldwide enterprise applications is strongly related to the ontology matching/alignment problem, which becomes the new core question.

2.2.1 The Canonical Data Model

The book *Enterprise Integration Patterns* by Gregor Hohpe [1] clearly formalizes problems with application integration. He provides an exhaustive list composed of 65 enterprise integration patterns to be considered when building a system able to manage the whole process of electronic business exchange. Its approach is based on a messaging system simply depicted in Figure 2.12. Focusing on those patterns for data integration, Gregor Hohpe suggests different approaches to resolve the problem. One is to share the same base of data like using a shared database or adopting the same base of documents between applications, but these patterns can be at most adopted within a single company. A second approach is to build a messaging system that translates business documents, called *message translator*, which is similar to the **point-to-point** approach presented in Section 2.1.2. Yet in the same approach a complementary pattern suggest using a *message mapper* which tries to conceptualize messages as business objects and thus more independent of application data. By doing so, he adds a pattern including a *Canonical Data Model* in order to minimize dependencies from different data formats. In this approach the Canonical Data Model provides an additional level of indirection between applications' individual format, similar to a pivot format, like a "lingua franca" for information systems. This approach is somewhat a mix of the **proprietary approach** with the **adoption of standard** approach seen above. In fact this approach is used by many industry specific consortia (like PIDX for the petroleum industry, or XBIT for the book industry) that produce a formal model specific to their use that must be adopted by all partners of a collaboration.

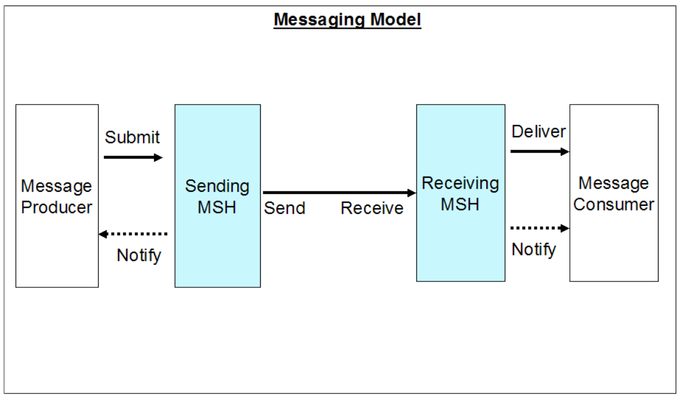


Figure 2.12 – A model of B2B exchanges based on messaging system (where MSH stands for Messaging System Handler)

In our approach we suggest adopting an ontology when building the specific B2B messages canonical data model. More than a pivotal format, we want to construct a *reference background knowledge* to improve application integration on the basis of a *message mapper* pattern. This approach is quite different from other experiences in the e-business domain, such as those provided by Corcho *et al.* [4], by Hepp [5] and by Fensel [23], because it targets message definition rather than a thesaurus like the eCl@ss ontology, since a message is not a well defined hierarchical set of products. This means that matching messages is a more complex operation because each message meets a specific action, which is not always the same for different standards. In other words, in a heterogeneous environment we are not able to say beforehand if the sending application has messages that correspond exactly to the receiver application messages, in a one-to-one association, but we can make the hypothesis that the sender application manages some "concepts" that are similar to those of the receiver application. In this context we consider a new pattern based on a canonical data model developed as ontology that aims to correlate these messages with common concepts. A procedure that performs such pattern is shown in Figure 2.13 and is as follows: 1) detect what concepts the message conveys; 2) match them with the canonical model; 3) find corresponding concepts in the target application data model; 4) chose the messages that fit the requirement best and finally; 5) translate.

However one main problem we meet here is the Canonical Data Model generation, which corresponds to the development of a domain ontology, or at least a reference ontology common to the whole B2B domain. The difficulty is that the classical development of this ontology is typically entirely based on strong human participation, which is a long task, really similar to the realization of a big standard and delves into a static knowledge representation. In the B2B context, where business partners can join a collaboration on the fly, the Canonical Data Model should be able to integrate new knowledge on the fly as well. In the following section we trace those requirements that such knowledge representation should have to fit into the B2B domain well and fills the assigned task in the pattern defined above.

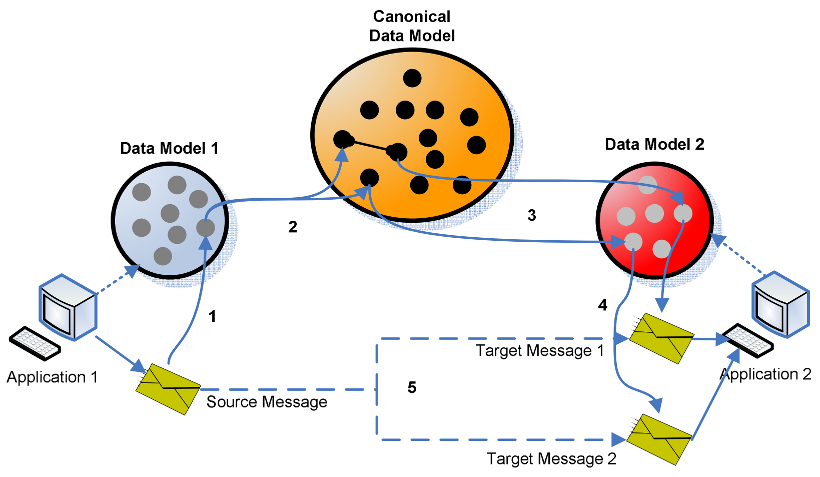


Figure 2.13 – Messages translation procedure

2.2.2 Ontology B2B Requirements

There are some general features that have to be respected when building an ontology, independently of the application domain. For example Barry Smith in his paper [133] examines the ISO 15926 upper ontology [134] and furnishes a series of principles to follow when developing reference ontology, of which we can mention: the principles of **intelligibility**; **openness**; **simplicity** and **re-use** of available resources; **coherence**; **compositional**, if two concepts are used to express a third concept, the formers must be included into the ontology; **singular** nouns, the terms of an ontology should be formulated in the singular. In his analysis he concludes that ISO 15926 is not an ontology because it does not follow any of these principles and the result is just a coding scheme rather than an ontology.

In a general way we can state that ontologies glue together three important requirements to consider when developing one:

- Ontologies aim at consensual knowledge, their development requires a cooperative process and normally, for pragmatics reasons (e.g. limiting complexity and dimension) they are restrained to a specific domain or application.
- Ontologies formalize semantics for information, consequently allowing information processing by a computer.
- Ontologies implicitly use real-world semantics, which makes it possible to link machine tractable content with meaning for humans.

We next detail some requirements that we have added specifically for the B2B use case, but they can fit others use cases as well.

Firstly the concept of **dynamicity** of an ontology for the e-business domain has been already introduced by Dieter Fensel in [23] which affirms that "Ontologies must have a network architecture and Ontologies must be dynamic". Also Martin Hepp in [135] sustains that otology must be able to grow dynamically without "bustling" existing applications. From the NeOn project we also find the concept of **networked ontologies** [136] [137] where ontologies can be distributed in a dynamic environment, like a peer to peer network, and applied to a B2B integration use case. At the same time computational time for discovering the best matches between several ontologies is expensive, therefore the technique applied to match elements should maintain previous discovered alignments and common uses in order to quickly recognize similarities between concepts and to compute only new information. We capture these characteristics in the following attributes for an ontology: memory, dynamism and polysemy.

Dynamism – An ontology is a static knowledge representation thus saying that it must be dynamic can be controversial in itself. In current literature the ontology dynamic is strictly associated to ontology evolution/versioning and has been investigated in several papers, like Noy *et al.* in [138] that traces all possible changes that can take place in ontologies. However when dealing with dynamic ontology we closely refer to the generation process of the ontology, like the life-cycle defined in Section 1.2.2, and with its capacity to introduce new knowledge interactively. For this the process should follow an iterative approach, i.e., conceptual knowledge may be integrated in turn. One condition that the ontology must respect in this case is the **completeness** criterion, which means that all matched concepts must be represented in the ontology and in the simpler case where a concept has no conflict with other concepts it is simply added to the ontology. Consequently an ontology is a dynamic characteristic of the domain, thus evolution should not be equivalent to a classical versioning system, but more to a learning system, including a merge operation without loss of information and backward compatibility. We call this feature the *dynamism* of an ontology.

From this viewpoint, and also from [30], ontologies and also applications using ontological background knowledge should not refer to the concepts as in a static model.

Memory – This feature of an ontology is strictly related to the previous one. Memory of an ontology provides a complete view of domain concepts and can be used as an anchor to identify quickly and accurately similarities between concepts, even if they are not identical, in order to conduct consistent alignments. For example concepts like *Postal Address* or *Delivery Location* can be referred to *Address* as upper concept, because even though the information they convey in a specific context can be different, they always represents the same basic concept of the ontology, the *Address*. The memory feature assures that even during a merging operation where two concepts are merged into an upper one, both basic concepts are still maintained as sub or related concepts in the resulting ontology. Moreover an ontology is not only a classification, or taxonomy of general concepts of a domain, it includes and maintains the most common properties of concepts, their relationships existing alignments and known semantics in an inclusive manner.

Polysemy – A third characteristic an ontology must have is the ability to provide the polysemeous forms that a term associated with a concept can have. Targeting dynamic ontology, where new knowledge can be added over the time, a term can have different uses depending on the context. For example, in English the term *Individual* can be used to define *Person* and in another context it can be synonymous with *Alone*. This difference can be detected by making a grammatical analysis of the text to see whether it appears an adjective or a noun, but if the corpus source is not a text, but as in our use case an XML Schema, its meaning must be drawn from its properties only. Thus the concepts must maintain the various groups of common properties and their type, what we call polysemy of a concept.

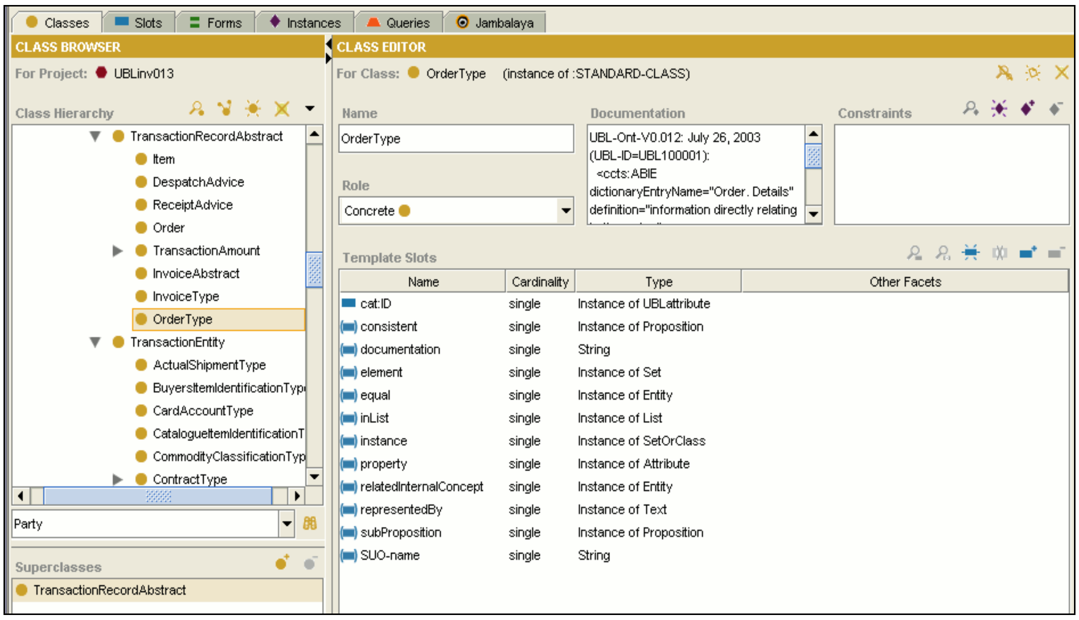
On top of these requirements, we want to be able to generate and enrich the ontology as automatically as possible. Indeed, even in a specific field, the concepts handled by the applications can be numerous and the quantity of information which we wish to maintain for each concept is vast. Solely relying on human management could quickly become impossible: recall that our example corpus size is thousands of XSD files and all the more concepts.

2.3 Existing B2B Ontologies

In this section we present some the most representative works on B2B ontologies. Among them, we can find some common points like: i) the fact that all of them are developed starting from existing B2B standards; ii) except the Ontolog Community with the UBL Ontology Project, all others develop a direct transformation from the input XSD format to an ontology language, mainly OWL, following the direct transformation generation process depicted in Section 1.2.3; iii) that all of them use B2B ontologies to improve matching and discovery of heterogeneous definition of similar concepts, but none of them continue to use ontologies as a message exchange formalism directly; iv) all these B2B ontologies are in a proof of concept phase or ongoing works, but as far as we know, no real business transactions are formalised with the help of ontology adoption yet; v) the generated ontologies are applicable to only a specific set of input sources, strictly related to the selected standard. Only the SET ontology tries to develop a more generic transformation, but still too close to the standards related to the CCTS model [139].

2.3.1 UBL Ontologies

The Ontolog Community UBL Ontology Project[25] started the design of the UBL ontology in March 2003. The aim of the project was to develop a formal ontology of the UBL Business Information Entities as defined by the UBL OASIS technical committee. The ontology is mainly hand made following the Ontology 101 method [140] and conceived as extensions of the Suggested Upper Merged Ontology (SUMO) [141]. They started formalizing UBL terms in SUO-KIF [142] extracting nouns and verbs from a UBL specification source text, then looked for classes in SUMO for the nouns and verbs extracted and finally mapped related terms as being either equal, subsuming or instance of. Figure 2.14 shows a view of the UBL ontology using Protégé editor.



For Project: UBLinv013

- TransactionRecordAbstract
  - Item
  - DespatchAdvice
  - ReceiptAdvice
  - Order
  - TransactionAmount
  - InvoiceAbstract
  - InvoiceType
  - OrderType
- TransactionEntity
  - ActualShipmentType
  - BuyersItemIdentificationTyp
  - CardAccountType
  - CatalogueItemIdentificationT
  - CommodityClassificationTyp
  - ContractType

Party

Superclasses: TransactionRecordAbstract

For Class: OrderType (instance of :STANDARD-CLASS)

Name: OrderType

Role: Concrete

Documentation: UBL-Ont-V0.012: July 26, 2003 (UBL-ID=UBL100001): <ccts:ABIE dictionaryEntryName="Order. Details" definition="information directly relating

Name	Cardinality	Type	Other Facets
cat:ID	single	Instance of UBLattribute	
consistent	single	Instance of Proposition	
documentation	single	String	
element	single	Instance of Set	
equal	single	Instance of Entity	
inList	single	Instance of List	
instance	single	Instance of SetOrClass	
property	single	Instance of Attribute	
relatedInternalConcept	single	Instance of Entity	
representedBy	single	Instance of Text	
subProposition	single	Instance of Proposition	
SUO-name	single	String	

Figure 2.14 – Ontolog Community UBL Ontology view

Another experience targeting UBL Ontology has been developed by Yarimagan and Dogac [143] from the Middle East Technical University. The so called UBL Component Ontology[26] is generated automatically by a conversion tool that reads UBL schemas and creates corresponding class, object properties and existential restriction definitions in OWL.

The Component Ontology template, shown in Figure 2.15, represents relationships between entities, types and business concepts. Each *xsd:ComplexType* and *xsd:element* declaration is a corresponding subclass under *DataType*, *TypeDefinition*, *ElementDeclaration* and Concept root

[25] http://ontolog.cim3.net/cgi-bin/wiki.pl?UblOntology

[26] http://www.srdc.metu.edu.tr/ubl/UBL_Component_Ontology.owl

classes of the Component Ontology. Every UBL element represents a unique business concept or an entity. This allows the definition of multiple elements representing the same business concept/entity and their correspondence is expressed through their relation to the same Concept class.

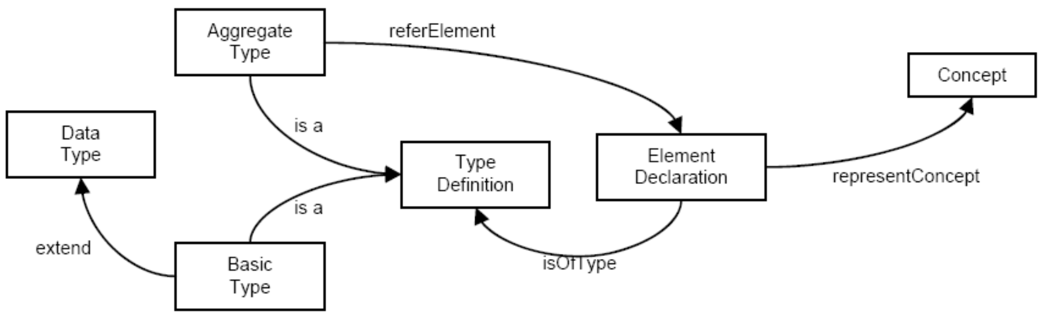


Figure 2.15 – Proposed UBL Component Ontology

Classes are related to each other through object properties where: Basic UBL types are defined through extending simple data types such as text, integer, date; the *referElement* object property represents the relationship between classes representing UBL aggregate types that refer to a similar set of elements; the *isOfType* object property represents the relationship between classes representing type definitions and element declarations; finally, the *representConcept* object property allows the definition of multiple elements that represent identical business concepts and relate element declaration classes to corresponding business concept classes. Listing 2.1 shows an example of the *ContactParty* concept expressed in OWL following the UBL Component Ontology representation.

```
<owl:Class rdf:about=" urn:ubl:CAC-2#ContactParty">
 <owl:equivalentClass>
  <owl:Class>
   <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
     <owl:someValuesFrom rdf:resource="#ContactPartyConcept"/>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#representConcept"/>
     </owl:onProperty>
    </owl:Restriction>
    <owl:Restriction>
     <owl:someValuesFrom rdf:resource=" urn:ubl:CAC-2#PartyType"/>
     <owl:onProperty>
      <owl:ObjectProperty rdf:ID="isOfType"/>
     </owl:onProperty>
    </owl:Restriction>
    <owl:Class rdf:about="#ElementDeclaration"/>
   </owl:intersectionOf>
  </owl:Class>
 </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:about="urn:ubl:CAC-2#PartyType">
 <owl:equivalentClass>
  <owl:Class>
   <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
     <owl:someValuesFrom>
      <owl:Class>
       <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="urn:ubl:CBC-2#WebsiteURI"/>
        <owl:Class rdf:about="urn:ubl:CAC-2#PartyIdentification"/>
        <owl:Class rdf:about="urn:ubl:CAC-2#PartyName"/>
        <owl:Class rdf:about="urn:ubl:CAC-2#Language"/>
        <owl:Class rdf:about="urn:ubl:CAC-2#PostalAddress"/>
        <owl:Class rdf:about="urn:ubl:CAC-2#PhysicalLocation"/>
        <owl:Class rdf:about="urn:ubl:CAC-2#Contact"/>
        <owl:Class rdf:about="urn:ubl:CAC-2#Person"/>
        <owl:Class rdf:about="urn:ubl:CAC-2#AgentParty"/>
       </owl:intersectionOf>
      </owl:Class>
     </owl:someValuesFrom>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#referElement"/>
     </owl:onProperty>
    </owl:Restriction>
    <owl:Class rdf:about="#TypeDefinition"/>
   </owl:intersectionOf>
  </owl:Class>
 </owl:equivalentClass>
</owl:Class>
```

Listing 2.1 – Excerpt of the UBL Component Ontology

2.3.2 XBRL Ontology Initiative

XBRL is a standard that formalizes financial reports. XBRL is used to define the so called XBRL taxonomies, which provide the elements that are used to describe information, instances, and give the real content of the elements defined.

Ruben Lara *et al.* in [144] advocated the use of OWL as an alternative to XBRL and produced a set of OWL files able to describe DGI[27], ES-BE-FS[28] and IPP[29] taxonomies. For this they have developed

[27] DGI stands for General Data Identification of economic agents Spanish taxonomy de agentes económicos (DGI as Spanish acronym)

[28] DGI is the Financial information report taxonomy for the *Estados Públicos Individuales y Consolidados*

a generic translation process of XBRL taxonomies into OWL ontologies[30] so that existing and future taxonomies can be easily converted into OWL ontologies following the transformation rules defined in *Table 2.2*.

The conclusion was that extensions to OWL are required in order to fulfil all the requirements of financial information reporting, to incorporate mathematical relations and that while its semantics can be appropriate (e.g. for investment funds classification), they could sometimes be problematic (e.g. for validation purposes). Finally they validate the adoption of such an ontology to automate and improve the classification and discovery of funds but do not use them as a formal format for data exchange.

Parsed taxonomy element	Root OWL class	Direct OWL subclasses
XML complex types	DGI ComplexType	A subclass for each complex type
XBRL Tuples XBRL items	DGI Element	DGI Tuple DGI Item
XLink links	DGI Link	DGI LabelLink DGI PresentationLink DGI CalculationLink
XBRL Contexts	Context (range of properties is subclass of ContextElement)	Subclasses of ContextElement: ContextEntity ContextEntityElement (Identifier) ContextPeriod ContextScenario
XBRL units	Unit (range of properties is subclass of UnitElement)	Subclass of UnitElement: UnitMeasure

Table 2.2 – Summary of parsed taxonomy element translations

2.3.3 RosettaNet Ontology

Armin Haller *et al.* [145] and [146] developed a WSMO [147] core ontology expressed in the WSML [148] formal language for the Supply Chain Management based on the RosettaNet standard. The process of developing a complete Supply Chain ontology from RosettaNet schemas is carried out in two steps: i) the core ontology is obtained by a direct translation from XSD to WSML including a reconciliation phase to hierarchically structure the ontology and to add a proper subsumption hierarchy; ii) RosettaNet specifications are analysed to identify remaining sources of heterogeneity in order to model and reference richly axiomatised ontologies, forming the outer layer in our ontological framework. As the previous experience they defined a set of rules from the XML representation to the selected ontology language, Listing 2.2 shows an example of such mapping from the XML extension element to its corresponding WSML formalism.

[29] ES-BE-FS is the Taxonomy of the Stock Quote Exchange National Commission

[30] The resultant OWL ontologies can be found here:
http://www.tifbrewery.com/tifBrewery/resources/XBRLTaxonomies.zip

```
<xs:complexContent>
 <xs:extension base="uat:IdentifierType">
  <xs:sequence>
   <xs:element name="ProductName" type="xs:string" minOccurs="0">
   <xs:element name="Revision" type="xs:string" minOccurs="0">
  </xs:sequence>
 </xs:extension>
</xs:complexContent>

hasIdentifierType ofType extIdentifierType

concept extIdentifierType subConceptOf uat#IdentifierType
  ProductName ofType (0 1) _string
  Revision ofType (0 1) _string
```

Listing 2.2 – Example of Complex extension type mapping to WSML

Authors argued that their ontology is able to resolve most of the heterogeneity problems between different RosettaNet implementations that are not structurally and semantically covered by the RosettaNet specification.

2.3.4 The SET Harmonized Ontology

The SET Harmonized Ontology is an initiative of the OASIS Semantic Support for Electronic Business Document Interoperability (SET) Technical Committee[31]. The purpose of this SET TC deliverable [113] is to provide standard semantic representations of electronic document artifacts based on UN/CEFACT Core Component Technical Specification (CCTS) [139] and hence to facilitate the development of tools to support semantic interoperability. The basic idea is to explicit the semantic information that is already given both in the CCTS and the CCTS based document standards in a standard way to make this information available for automated document interoperability tool support.

The resulting ontology[32] provided by Asuman and Kabak is currently the most valuable effort in describing an upper ontology for the real B2B domain. The SET Harmonized Ontology contains about 4758 Named OWL Classes and 16122 Restriction Definitions. Their approach is a semi-automatic derivation of an ontology from the business data components defined by OAGIS, GS1, UBL and UN/CEFACT CCL, which are all B2B standards based on the CCTS specification. Another point of interest is that it is one of the rare experiences applying a strong adoption of Semantic technologies, like DL reasoners, SPARQL, OWL and OWL queries to derive a harmonized ontology. This can be viewed as similar to a merging operation.

Without delving into details Figure 2.16 shows an overview of the SET upper ontology. The overall process to get the harmonized ontology is as follows: i) first specify an upper ontology, which is an OWL description of the CCTS specification; ii) transform input source documents into schema ontologies, which are afterwards mapped manually to the defined upper ontology format and thus

[31] http://www.oasis-open.org/committees/set/

[32] The SET Harmonized Ontology is publicly available from http://www.srdc.metu.edu.tr/iSURF/OASIS-SET-TC/ontology/HarmonizedOntology.owl

automatically transformed to OWL compliants files; iii) define four normative upper ontologies, one for each of the UBL, GS1 and OAGIS® 9.1 standards separately, while the UN/CEFACT CCL is considered as upper ontology of reference. While creating these ontologies, the relations with the CCTS upper ontology classes are also established. Finally, with the help of additional heuristics, using a Description Logics (DL) reasoner, a Harmonized Ontology is computed.

The resulting ontology and heuristics enable the discovery of equivalences and subsumptions of structurally similar document artifacts between two document schemas. When translating such document artifacts, automatically generated XSLT rules are used, that produce query templates (SPARQL and Reasoner based queries) to facilitate the discovery and reuse of document components.

The advantage of this approach is twofold. Firstly it shows the powerful benefits of semantic technologies. Even with a more complex syntax description, a reasoner is able to autonomously discover several useful subsumptions and equivalences. It also shows that it is possible to provide a first real B2B normative upper ontology formalization that could lead into a new era of B2B standard ontologies development.

However a strong and somewhat limitative hypothesis is that input sources must be compliant with the CCTS specification. This is not applicable to the whole domain and thus prevents a larger adoption of this solution. It is also unclear how the different semantics of input elements are matched. For example, as presented in Figure 2.17, it is not clear how the *NameAndAddress* class has been associated to the owl *Address* class. For instance an automatic matcher should have to choose between the classes *Name* and *Address*, which is not the case in the resulting ontology. Another example is the detection of the semantic equivalence between *Postal_zone* and *Postcode*, which is not explained.

To conclude, this approach also lacks the definition of a semantic matcher and we argue that the integration of such a module could improve resulting correspondences and help in possible ambiguities.

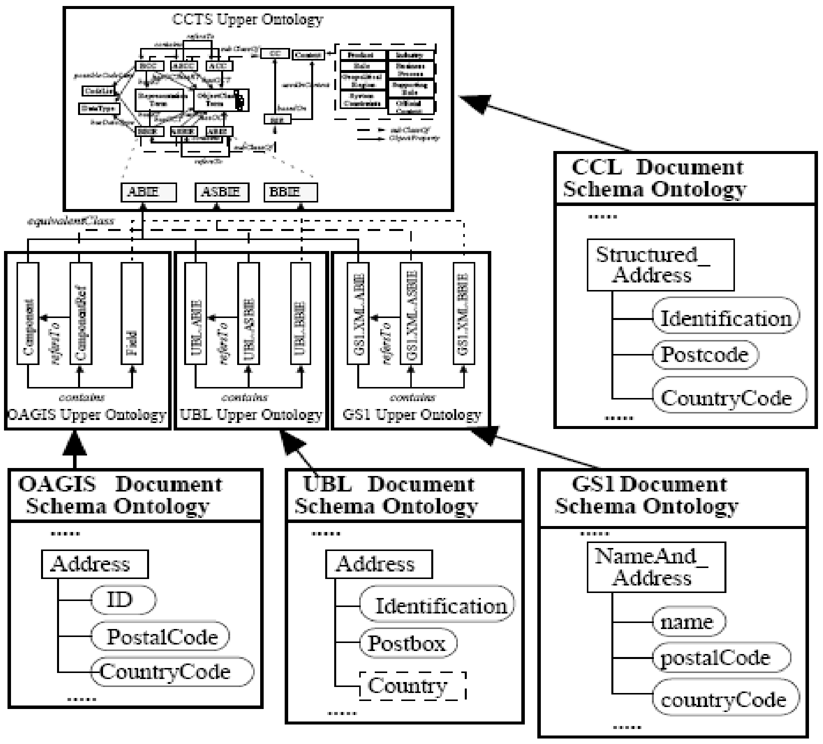


Figure 2.16 – An Overview of SET Upper Ontologies and Document Schema Ontologies

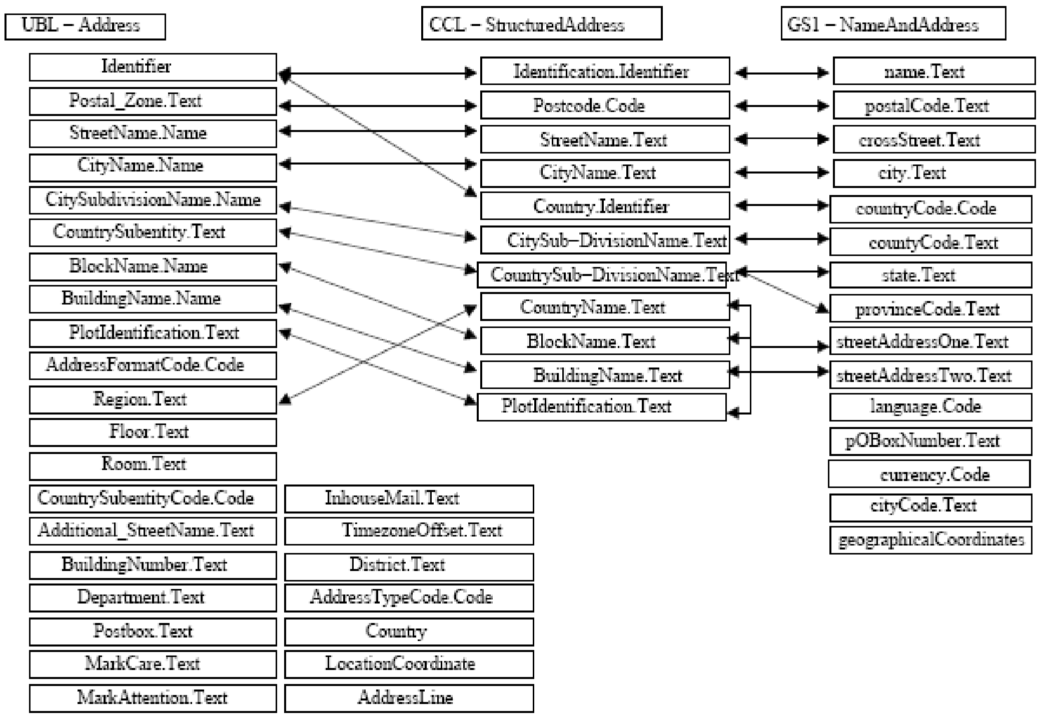


Figure 2.17 – The Semantic Equivalences among the BBIEs of UBL-Address, CCL-Structured Address and GS1-NameAndAddress Discovered through the Harmonized Ontology

2.4 Conclusion

In this chapter we presented the B2B domain, the requirements that it currently imposes on companies and their IS in order to support business messages exchanges. Through this analysis we pointed out the current architecture limitations and explained why ontologies are the best approach to follow to gain in flexibility and dynamicity.

Nevertheless facts show that it is still not the case and B2B standards, which are the most adopted solutions for B2B, yet do not define standards as ontologies but still as XML Schemas. Although it is already a respectable improvement with respect to older systems like EDIFACT, it still requires relevant human effort to be operational.

In this sense we have provided an analysis on B2B ontology requirements and grouped them into three main elements which are dynamism, memory and polysemy. Afterward we have presented the most known ontologies for B2B. Despite the interest of these works, still real businesses seem to hesitate in their implementation. So initially we identified two main topics to develop, one on the definition of an enterprise semantic repository, and the other one a way to facilitate the automation of business document mapping. Finally we have been interested by a system that facilitates, by automation, the transformation from the current model to the "next one", from XML to OWL, believing that the existing gap could be shortened improving this direction.

So after large introduction (yet somewhat shallow with respect to the complex B2B domain), we now leave this specific theme to develop our thesis that delves into a more general solution to the ontology generation automation. The adoption of Web Semantic technologies to business messages exchanges has an essential requirement, which is that messages must be semantically well defined using ontologies.

Chapter 3.

Semantic Data Model for Ontology

The two previous chapters introduced some problems we have with the matching process for automatic ontology generation. We showed how concepts matching and ontology generation are strictly correlated. As shown in Section 1.2.5, several matching systems adopt intermediary conceptual or semantic models to reduce the complexity of the integration process, but even if conceptual models are largely used, few works still provide a complete description of such model for ontology generation. Moreover several intermediary data models follow an approach that belongs to the deterministic method defined in Section 2.1.3. They often do not permit to store probable relationships between concepts, but only exact concepts and relationships. For example in current approaches a relationship like synonymy can outcome as an equivalent concept in a specific matching but not in another depending on the context. In the first case the correspondence can be mapped in the model as a relation of equivalence and the information kept in the resulting model, but in the second case it is ignored. Consequently if ignored, through this approach we delve in a new onerous re-calculation the next time the labels are met again.

For this reason we have defined yet another conceptual data model. However this is not a completely new formalism for conceptual modelling but an adaptation of the approach followed by the CCTS [139] model for business components description. It is fundamentally an object oriented model deriving from UML to which we added two main features in order to provide a solution to the limitations described above.

The first feature we added is a set of predefined, but extensible, meta-association defining different level of similarity relationships that can subsist between concepts of a model instance. The second feature is linked to the Dynamic Object Model [149]. It permits to resolve inconsistency between heterogeneous design representations of similar concepts, like granularity, and provides more flexibility. For this the nature of an object (e.g., attribute or class) is not frozen at the design time, but derived dynamically according to its characteristics at the moment the model instance is queried. Furthermore the targeted model should be generated by machine computation by adding sources incrementally, which means that the model can automatically evolve in due course as new sources are added. In addition we add rank and frequency measures to provide statistical evaluation to try to resolve ambiguous situations that can arise from multi-sources representations.

We decided not to provide a serialisation format for our dynamic conceptual model but to use the OWL format representation and transformation as basis for the serialization. This choice permits to maintain a better relationship with the final targeted formal ontology generation and at the same time a more sharable and directly integrable format, with the possibility to use other tools to improve our work.

So this chapter is devoted to SDMO (Semantic Data Model for Ontologies), the model we propose as intermediary conceptual model to maintain a list of concepts and relations to reuse for building automatically an ontology and to provide useful information to matcher systems. It is outlined as follows: Section 3.1 delves into the drawing of our model, trying to provide a formal description and main features; Section 3.2 defines the OWL representation, motivating our choices, and the representation of SDMO using OWL; Section 3.3 provides a deeper overview of related works and shows their limitations with respect to our requirements. Section 3.4 concludes this Chapter.

3.1 SDMO Description

As mentioned above, several models have been defined either to provide a conceptual representation of an input source or to maintain final alignments and exact correspondences. However the most part of these models can be used as representation for a single input at a time, as a final version of integrated sources or as a bridge between two inputs sources.

In this section, we describe the **Semantic Data Model for Ontologies** (SDMO) defined to provide an organized model to record as much knowledge as possible for matching systems. **The goal is improving the concept correspondences similarity detection**. The improvement that we target with this model is the machine capability to faster recognise similar concepts on the basis of their relationships and from this the ability to adopt more efficient algorithms to refine mappings. This Section is organized as follows. In the first subsection we provide the main requirements to which the model must answer. In section 3.1.2 we provide an informal description of SDMO. In Section 3.1.3 we formalize the relationships. Section 3.1.4 provides a more formal definition of a concept and its nature. In section 3.1.5 we furnish some elements for measuring the frequency and rank concepts and relations and finally in Section 3.1.6 we depict all graphical elements we use for the model.

3.1.1 Model Requirements

Before providing the description of the model let us show an example of what we want to modelise. For this Figure 3.1 illustrates three different representations of a similar concept. The first one is called *Coordinate* which has three attributes, *Latitude*, *Longitude* and *AltitudeMeasure*, and two of them are defined as a *Position* which is further detailed with other attributes. A second representation defines a concept named *GeographicalCoordinate* with only two attributes, again *latitude*, *longitude*, that are likely defined as to be strings. The last one named *SimpleCoordinate* has also two attributes but these are *SystemID*, and *CoordinateReference*. The question we have is: if exists, what is the right conceptual representation for these three definitions?

Indeed it represents a classical example of what we find in our use case.

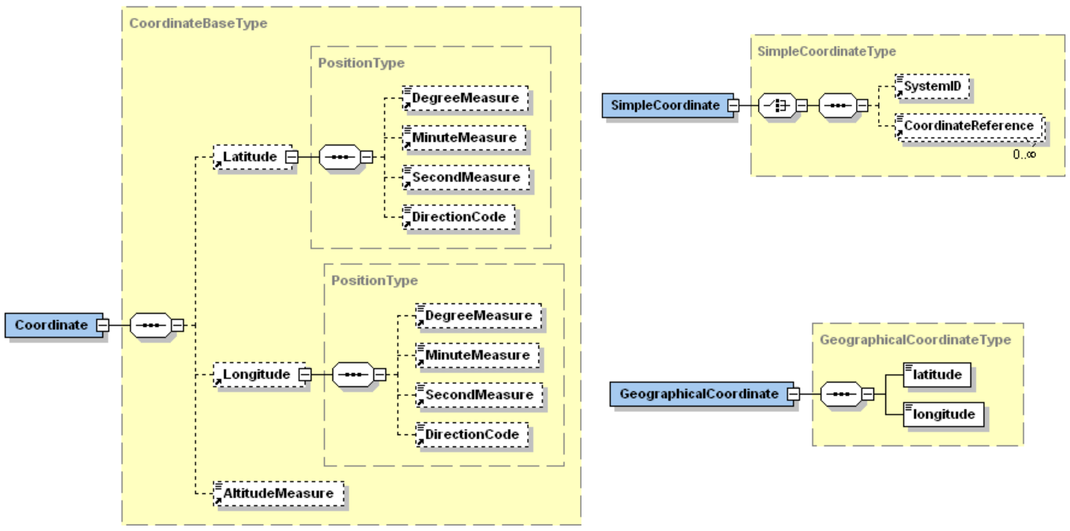


Figure 3.1 – Examples of XML Schemas representations of the concept Coordinate

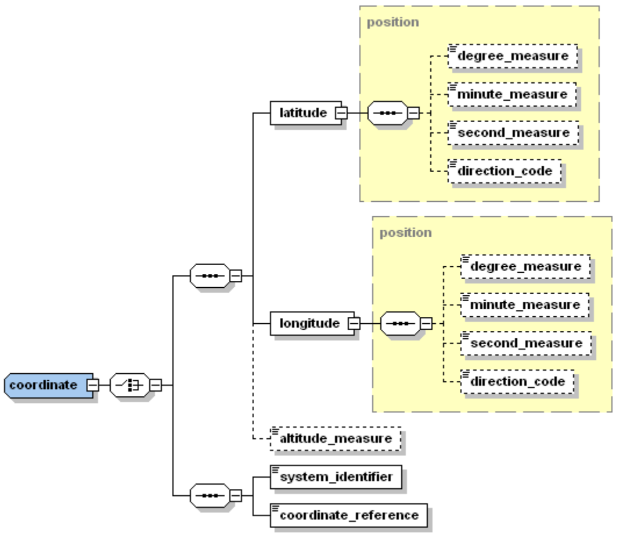


Figure 3.2 – An integrated view for Coordinate concept

This is the challenging task that we propose to our system and in this Chapter to our model.

So given this set of XML Schemas as input it is humanly simple to imagine a possible consensual result. For example like the one illustrated in Figure 3.2 with a sole main concept named *coordinate* and having all attributes with a possible choice among the two different aggregations of attributes. At the same time it seems logic to maintain *longitude* and *latitude* with the deeper granularity using *position* as sub-type. But now what happens to our modelization if another representation for the same set of concepts comes up again? How to maintain choices made for the future? And also, how to find it automatically?

3.1.2 SDMO Informal Description

The basic representation of SDMO is data about concepts and relationships. Such **object-based** modelling allows a high level of data definition independent from the different representations. A second basic precept of our model is that many relationships are **functional** like they are in nature. Such kind of functional relationships are often called *has attribute* in models like the SDM [150], IFO [151] and the more known Relational Model [152] and Entity-Relationships [153], or *functional property* in OWL. In our model these relations are part of the set of what we call **structural** relationships which also provides hierarchical mechanisms for building object types out of other object types. For example, *address* and *postal address* that might be the aggregation of *street*, *city*, and *country*.

A third basic percept is the **semantic** relationship, which specifies the fact that some concepts share a common meaning, like synonyms.

A fourth basic element of the model is the set of **syntax** or **linguistic** relationships. The aim of this kind of concepts relations is to maintain the link among concepts having a similar name, like *postcode* and *postal code* attributes, or names sharing the same stem. This kind of relations brings us more inside the characteristics that we want to give to the model. These are not a natural human percept that we find in other models for the real-world representation, but a more natural feature for matchers, which need to compute an operation.

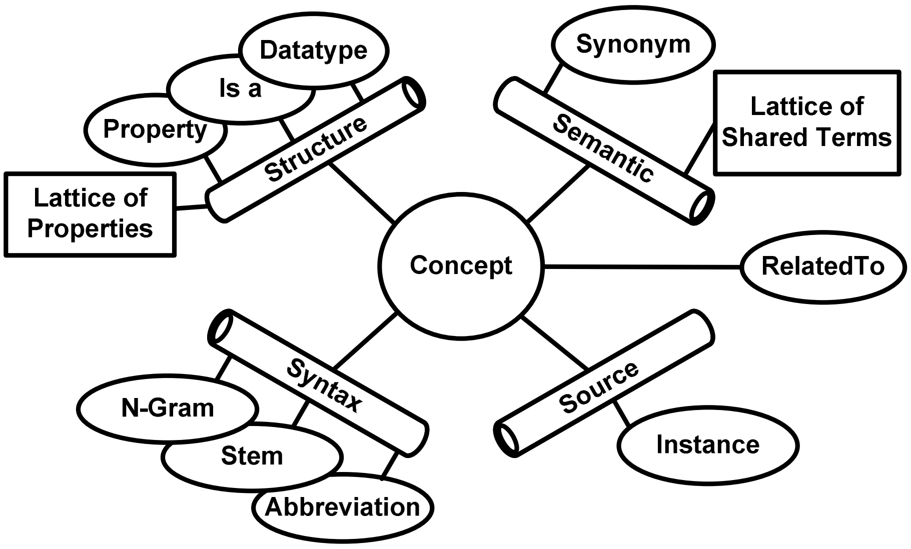


Figure 3.3 – SDMO Concept relationships overview

The fifth and final basic element is a link to the original input. Normally a matcher compute a normalization of initial labels and during this operation some little details can be lost and at the same time it is important to maintain the link with the source in order to be able to regain the original context or to produce a mapping. In our model these relations are part of the set called **source** relationships. Figure 3.3 shows the overall view of SDMO concept relationships.

Moreover our model incorporates these principles within a coherent, graph-based representational framework. So that we can also define a SDMO schema as (formally speaking) a directed graph with various types of vertices and edges.

3.1.3 A More Formal Definition of SDMO Relationships

The representation of relationships subsisting among concepts is the first SDMO component. Differently from other intermediary conceptual models (shown in Section 3.3), our model focuses on the storage of discovered links. Links are modelled as valid relationships of different kinds between concepts. Thus, using the final model instance, it should be possible to return from a given concept all similar concepts already contained in the model. The aim is not to return all exactly equivalent concepts of a given one, but rather those who are correlated with it. The final choice of the best correspondence is done subsequently with more specialized matching algorithms that refine the query. This feature allows the use of simple but efficient algorithms to compare disparate concepts, like *umbrella* and *washing machine*, as discussed in Section 1.3. The gain estimated in both quality and efficiency should be notable. For instance we can apply a matching algorithm with exponential computational complexity order if we have few elements to analyse at once, but we cannot use the same algorithm over a large set of input concepts. We will come back on this aspect later in the implementation chapter (Chapter 5). We now introduce the different types of relationships of the model.

We distinguish the following categories, natures, of relationships: (i) **Semantics**, (denoted with ***S***) including shared terms (also known as tokenization) and synonyms; (ii) **Structural**, (denoted with ***H***) including properties of, data types, *is-a* and equivalence; (iii) **Syntactic**, (denoted with ***L***) including close string match value and abbreviations; iv) **Source**, (denoted with ***I***) which maintain links with sources and original elements from which concepts have been derived. Figure 3.3 already depicts these groups of relationships.

More formally SDMO relationships can be either symmetric or directed binary relations that subsist between two concepts of the model. A relation is defined as a quadruple*:*

$$r = < c,\ d,\ type,\ f >$$

Where, c and d are concepts of a model instance, $type$ defines the nature of the relationship that ties together the two concepts and f is a frequency/rank measure. Sub-sections below detail relationship types.

3.1.3.1 Semantic relationships

Semantic relationships (***S***) aim at building a graph of neighbourhoods of concepts having a common meaning. In reality very few matching algorithms are capable of making meaning-based similarity choices [154] and even less tools and algorithms are available to this scope yet. We currently describe a semantic relationship on the basis of WordNet [59] associations and **shared term** relationships. At present only a general WordNet **synonymy** is considered, but a finer model definition could integrate more specific WordNet relations, like those between WordNet concepts, called synsets, (hyponymy, entailment, similarity, member meronymy, substance meronymy, part meronymy, classification, cause, verb grouping, attribute) or also between word senses (derivational relatedness, antonymy, see also, participle, pertains to).

Shared term relationships target a fast way to match compound words with common terms, like *PostalAddress* and *ShippingAddress* having *Address* as common term. These kinds of associations are relevant when we consider XML tag names as input candidate for concept names: they reflect the common practice when building tag names with sequence of terms. This practice is usually adopted for data definition [15].

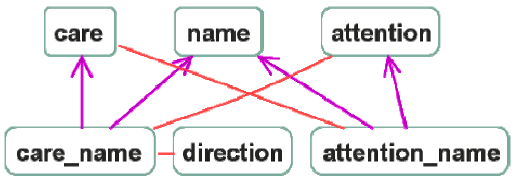


Figure 3.4 – Semantic Relationships

This XML tag annotation has the advantage of providing a human readable format but cannot be exploited by a machine as is. Indeed it is known that classic string matchers algorithms, like N-Gram, or Levinstein distance based matchers, fails when trying to match compound words labels, and the only way to match labels is to reduce compound words in sequence of terms and only then apply such matchers. The construction of a lattice of Shared Terms provides a fast machine interpretable format that can provide good relationships between concepts with similar names like *care_name* and *attention_name* of Figure 3.4. It is of simple understanding that tokenizing compound words with their lemma[33] and build a lattice over them provides a direct machine exploitable form to look for sub-string similarities between labels having common terms. Section 3.1.3.2 delves into details of the Shared Terms lattice.

Synonym relationships rely on a common dictionary based synonymy between terms, like *care* and *attention_name*, *attention* and *care_name* (thus between *care* and *attention*), *care_name* and *direction*, represented by a simple line in the SDMO graph of Figure 3.4. The peculiarity of the synonymy relation between words is that it is really useful to find out similar concepts from different sources, nevertheless it can be misleading in the most cases, like *care_name* and *direction* of Figure 3.4 identified by WordNet, and in addition the detection of synonyms can be onerous in time. For this

[33] A lemma in morphology is the canonical form or citation form of a set of forms (headword); e.g. in English, run, runs, ran and running are forms of the same lexeme, with run as the lemma.

reason we decided to maintain this information in the model even if in a certain context the two concepts are not equivalent.

3.1.3.2 Shared Terms Lattice

The Shared Term relationship is particularly useful when concept names are compound words, because names correspondence recognition is the base of matching algorithms. Common algorithms adopt string matching or distance measure and it is obvious that in the most cases these kinds of algorithms can fail when applied directly to compound names. At the same time the tokenization is an operation that can be forgathered because we observed that even in different sources, compound words are often similar. With the construction of a lattice[34] graph based on shared terms, terminological similarities can be quickly discovered and/or discarded for those concepts with/without naming equivalences. In this section we define the lattice, that we also call **Lattice of Words** (denoted with **WL**), built over the Shared Term relationship.

Firstly we formalize the lattice as formed by only compound words elements, other relations between elements of the lattice can be added subsequently without modifying basic properties of the lattice, just realizing a graph of concepts. This approach is similar to FCA depicted in Section 3.3.2, just we rely on the lattice only more specific relations between concepts which delve into a more flexible and complete graph, rather a limited relation representation to extents and intents. In this specific case our extent are just terms used to define concept names, while intent are concept names (labels) themselves.

Let w a short sequence of terms t_i that for simplicity we formalise as: $\{w\} = \{t_1, ..., t_n\}$, *for* $1 \leq n \leq 6$. We define w as a **compound word**. Moreover we define the absolute value of $|w|$ the compound word where the terms can appear in w with any order constraint, e.g. if $\{w_1\} = \{t_1, t_2, t_3\}$ and $\{w_2\} = \{ t_2, t_1, t_3\}$ than $\{w_1\} \neq \{w_2\}$ but $|w_1| = |w_2|$.

We limit to 6 the upper bound value of n, the number of terms of a compound word, because heuristically more than 6 terms loose sense and the compound word could be considered like a pseudo-sentence itself and other grouping techniques should be adopted.

Definition 2: A Shared term relationship ***St*** is a directed association that subsists between two compound words every time they have at least one common term, marked as $\boldsymbol{w_1}$ **St** $\boldsymbol{w_2}$**.**

Let D be a set of all compound words, $D = \{w_1, w_2, \ldots, w_n\}$.

Let be $w_x, w_y \in D$ such that $|w_x| \cap |w_y| \neq \varnothing \Rightarrow w_x$ ***St*** w_y.

Where we define the intersection operation ($\cap$) between two compound words as the matching common terms they have (i.e. the number of shared terms), while the union ($\cup$) is the set of all terms composing the two compound words.

Shared term relationship derived properties:

[34] In mathematics the lattice is a partially ordered set in which any two elements have a *supremum* (also called extent) and an *infimum* (also called intent).

i) **Reflexive property:** w_x ***St*** w_x.

ii) **Simmetric property:** if w_x ***St*** $w_y \Rightarrow w_y$ ***St*** w_x.

iii) $|w_x| \cap \varnothing = \varnothing$ and $|w_x| \cup \varnothing = w_x$

iv) $|w_x| \cap |w_x| = w_x$ and $|w_x| \cup |w_x| = w_x$

Shared term relationship definitions:

i) if $w_x = \{t_1, \ldots, t_m\}$, $w_y = \{t_1, \ldots, t_m, t_{m+1}, \ldots, t_n\}$ with $m < n$, thus w_x ***St*** w_y, in this case we say that w_x is a **direct subsequence**, like a **sibling node** of the lattice, of w_y, which is the **master node** $\Rightarrow |w_x| \cap |w_y| = w_x$ and $|w_x| \cup |w_y| = w_y$ and of course $\{w_y\} \supset \{w_x\}$

ii) if $w_x = \{t_1, \ldots, t_m\}$, $w_y = \{t_1, \ldots, t_m, t_{m+1}, \ldots, t_n\}$, $w_z = \{t_1, \ldots, t_m, t_{m+1}, \ldots, t_h\}$ with $m < n,h$ and $m \geq l \Rightarrow |w_y| \cap |w_z| = w_x$ and $|w_y| \cup |w_z| = w_x$, in this case we say that w_x is the **root node** (or **root word**) for w_y and w_z $\{w_y\} \supset \{w_x\}, \{w_z\} \supset \{w_x\}$, and $\{w_y\} \cap \{w_z\} = \{w_x\}$

iii) if $\exists w_x, w_y \in D, \exists w_z \notin D \mid |w_x| \cap |w_y| = w_z$, In his case we say that the root word is an extension of D and that D is a **non complete set of compound nodes** for the domain.

iv) if $\forall w_x, w_y \in D$ such that $|w_x| \cap |w_y| = w_z \Rightarrow w_z \in D$, in this case we say that D is a **complete set of compound nodes** for the domain.

Let be *Dc* the completed set of words extracted from a domain, $Dc = \{w1, \ldots, wn\}$, than the Lattice of Word (**WL**) is defined as a tuple of words and *St* relationships: $WL = <w, St>$, where $w \in Dc$ and *St* is the set of binary associations between words. While we define **root nodes** of the WL those words belonging to *Dc*, the completed set of words *D*, such that for each w_x belonging to the set, $w_t \cap w_x = w_t$ for each not empty intersection. $\forall w_i \in D$, $R = \{w_r \in Dc \mid w_i \cap w_r = w_r\}$

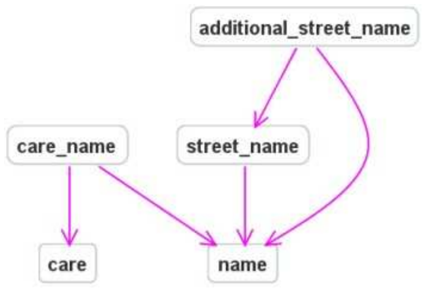


Figure 3.5 – Words Lattice example

Figure 3.5 illustrates an example of a word lattice where, given a set of compound words derived after the normalization and tokenization phase from the following four tags: *CareOfName*, *AdditionalStreetName*, *StreetName* and *CareOf*, we obtain the set of words *D = {care_name, street_name, additional_street_name, care}*. The completed set of *D*, *Dc* is the following *Dc = {care_name, street_name, additional_street_name, street,* ***name****}*, where *name* has been added in order to fill the intersection between *care_name* and *street_name*. While *R*, the set of root words is composed by *care* and *name* (they are not necessarily simple words).

3.1.3.3 Structural relationships

Structural relationships (***H***) provide hierarchical and properties relations between concepts of the model. These are: **has property** (inverse relations of **property of**), **has printable-types, is a** (inverse relations of **super-class**) and **equivalence** relationships. Except done for the equivalence relation all other are not symmetric relations.

Is a relationships define if a concept is considered as specialization or inversely a generalization of another. Intuitively such an association can be used to qualify possible roles of a concept in a specific context, or in a specific usage. For example a *student* can be a *person*, or a *delivery location* that is an *address*. This relation is denoted with ***IS*** and we say that given a non empty set of concepts O and $a,b \in O$, $a\ IS\ b$ if b is more generic than a.

Has printable type (denoted with ***Pt***) structural relationship defines if a concept of the model can be expressed directly by a basic printable element like string or integer. In our case we defined the list of basic printable elements as correspondent to the XSD list of basic data types [155].

Property of (denoted with ***Po***) relationship defines if a concept is an attribute of another concept. Inversely **has property** (***Ph***) defines if a concept has an attribute. More formally defined as:

Definition 3: Given a non empty set of concepts O and $a,b \in O$ than we say that $a\ Ph\ b$ if b is an attribute of a.

Definition 4: We also define the set of attributes for a given concept a (or also group of properties), the aggregation $P(a)=\{b\},\ \forall a \in O$ as $\{\forall b \in O \mid a\ Ph\ b\}$

The final structural association of the SDMO is the representation of the **equivalence** (***EQ***) relationship. This kind of association naturally relates concepts having the same meaning. Figure 3.6 shows an example of equivalence relationships between *geographical_coordinate*, *coordinate* and *coordinate_base*.

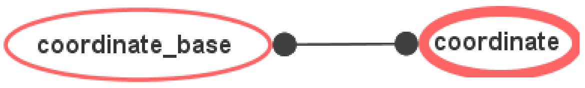


Figure 3.6 – Graphical representation of an equivalence relationship

In the aim of merge several input sources thus set of concepts, we introduce the **principle of maximum inclusive**, that we can define the equivalence relationship in terms of shared attributes.

Definition 5: (Maximum inclusive) – Let be O, O' two sets of concepts and M the resulting merged set of concepts, $M = O \cup O'$, $a \in O$, $b \in O'$ and $c \in M$ with respectively $P(a), P(b), P(c) \neq \varnothing$. Let $P_i \neq \varnothing$ a generic non empty group of properties. Moreover we define the intersection operation between two property groups as the number of common elements they have (i.e. the number of equivalent common concepts properties).

i) if $P(a) = P(b) \Rightarrow a \equiv b$; (**equivalent** concepts)

ii) if $P_1(a) \cap P_2(a) = \varnothing \Rightarrow$ if *a* and *b* have the same concept name but have no common attributes than we say that *a* and *b* are **polysemic** concepts

iii) if $P(a) \cap P(b) \neq \varnothing$ and $P(a) \subset P(b) \Rightarrow b$ is the master concept and *a* represents a specialization/restriction of *b*

iv) if $P(a) \cap P(b) = P(c) \Rightarrow$ we say that *c* is property of *a* and *b*

if $P(a) \cap P(b) = P(x) \Rightarrow$ we introduce heuristics to determine if *a* and *b* can be structurally related with the following formulae:

$- \sigma_a = (\# P(x)) / (\# P(a));$ Variance of 'a'

$- \sigma_b = (\# P(x)) / (\# P(b));$ Variance of 'b'

$- \rho = \max(\sigma_a, \sigma_b);$

Finally $0 < \alpha < 1$ a threshold parameter, if $\rho \geq \alpha \Rightarrow a \approx b$

Thus concepts classes respecting this formula can be related and *P(x)*, that we call *common causality of properties*, is a distinctive set of properties.
In the formulae *#P(x)* is the cardinality of *P* expressed in terms of number of contained attributes and the *variance* is calculated as the ratio between cardinalities of the common causality and the concepts. This because we simply assume that common elements, concept properties, are exactly the same or not at all (thus expressed with a similarity value equal to 1 or 0). The formulae could be improved by adding a different value of similarity using an appropriate distance between identified common elements.

These rules introduce some important choices that will be considered during the implementation of the prototype in order to be able to characterize relationships between concept classes.

3.1.3.4 Lattice of Properties

Similarly to semantics relationships also in structured relations we add a lattice to group common concepts' attributes of the model. In this case the extents of the lattice are concept classes, while intents are concept properties, normally *common causality of properties* (seen above in Definition 5). Definitions and properties defined for the Shared Terms Lattice still stand for the Lattice of Properties (denoted also with **PL**), with the difference that intersections and unions are done over groups of properties and not on sequence of terms.

3.1.3.5 Syntax correlations

Let us also stress **syntax** groups of relationships (***L***), which maintain associations between retrieved concepts having common abbreviations, stem or a close value using a relevant syntax distance measure, for example up to a specified threshold measured with algorithms like N-Gram or Levenstein distance.

However, aiming automation for concepts similarities detection, these kinds of relations often lead to misleading relation because of their "mechanical" nature rather semantics and structural, thus these relations need further refinement before to be maintained in a model instance, and consequently they can be used no more than anchor links between concepts.

3.1.3.6 Other relationships

Always in Figure 3.3 we can see **source** relations which aim to maintain links with the context of extracted concepts and their original labels, cardinality and also instances of the concepts.

Finally the **related to** relation aims to store all kinds of relations that until now have not been explicitly designed in the model. For instance, merged concepts are not removed in model instances (for the completeness rule of model instances defined above), thus in this case we use a specialization of the *related to* relation to maintain the information that two concepts have been merged.

3.1.4 SDMO Concept Definition

A SDMO concept is the constituent entity of the model and is defined as a quadruple:

$$c = < \mathcal{L}, \mathcal{R}, \mathcal{S}, f >$$

Where:

- $\mathcal{L}$ is a set of words, simple or compounds, that best represents the name of the concept. Among them we also define a *preferred label* as the best representative label as concept name (e.g.: having extracted concepts named *geographical_coordinate* and *coordinate,* they can be merged to form the same concept and the final name can be one of them)
- $\mathcal{R}$ is the set of relations between concepts (all seen above)
- $\mathcal{S}$ for Source, is the set of originating instances of a concept (not to be not confused with instances as individuals in the ontology representations)
- f is a frequency and/or rank measure

Similarly to UML and many other models, in SDMO we defined three basic kinds of concepts, also called nature of the concepts, but a concept can be of more kinds at the same time or change all over its "life in the model". No mandatory relationships are required beforehand for a concept, but depending on them, we can determine dynamically its nature. These three types are: *class*, *property* (or *attribute*) and *printable-type*. Figure 3.7 shows an example of a simple graphical representation of the three basic SDMO concepts, where *string* is a **printable** concept, graphically represented by a rectangle, *latitude* and *longitude* are concept **properties** (or attributes of a concept class), graphically represented by a rounded rectangle, and *geographical_coordinate* is a concept **class**, graphically represented by an ellipse.

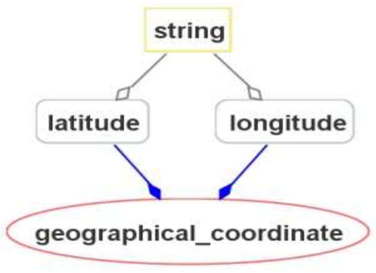


Figure 3.7 – SDMO basic concept structures

The main concept type is called **class** and corresponds intuitively to non atomic concepts, thus to concepts characterised by a finite set of attributes. The second basic nature of a concept is the **property** (or attribute). It represents either a specific and atomic characteristic of a class or also a role that semantically redefines another concept class, like an UML association (e.g. *address* that becomes a *residence for a* person or a *delivery address* in another context). The foster typically corresponds to concepts in the world (of data exchange) that have no underlying structure. Simple examples are *first name* and *last name* of a *person,* or *city name, etc.* The last one and most basic concept type in the SDMO structure is the **printable type**. This kind of concept can be also considered as the type that serves as the basis for application inputs and outputs. It can be a conventional basic type, such as *string* or *integer* or a more complex representation of a printable data type like *measure*, *amount*, or *text* that in turn are directly linked to basic types.

We stress out the fact that a concept can be of different types at the same time, they are not strictly closed to be of only one nature at once, but depending on their behaviours they can be seen for example as a class or a property. For instance a **class property** SDMO concept is allowed and is a non atomic concept, thus a class, which is also property for another concept class.

More formally:

Definition 6: Let O be a set of concepts, c a concept of O and $P(c)$ a non empty set of properties for c. A concept c is a class if $P(c) \neq \varnothing$. Consequently we define the set concepts classes C as
$\{ \forall c \in O \mid \exists p \in O$ and $p \in P(c)\} \Rightarrow c \in C$ and $C \subseteq O$

Definition 7: Let O be a set of concepts, $p \in O$. A concept p is a **property** if $\exists c \in C : p \in P(c)$. Consequently we define the set concepts properties C as
$\{ \forall p \in O \mid \exists c \in C$ and $p \in P(c)\} \Rightarrow p \in P$ and $P \subseteq O$

Definition 8: Let O be a set of concepts, B a predefined list of basic type elements and $b \in B$. A concept $d \in O$ is a **printable type** if $d \in B$ or $d\ Pt\ b$ and $d \notin P \cup$C. Consequently we define the set printable concept D as $\{ \forall d \in O \mid d \in B$ or $d\ Pt\ b$ and $d \notin P \cup C\} \Rightarrow d \in D, D \subseteq O$ and
$D \cap (P \cup C) = \varnothing$

As defined above a class is a non atomic concept, which implies that a class must have more than one property. Thus if a class has only one property we assume that the property is just a key differentiator of a class (like a name or an ID) because it does not provide further information.

3.1.5 Ranking Concepts and Relationships

Aiming at building dynamic ontologies by a model instance generated automatically from incremental addition of input sources, it is not uncommon to find incoherent or conflicting data descriptions. For example sometimes we can find cycles in non symmetric relations like data structures e.g. let be *a*,*b* concepts for *O*, and *Po* the property relation, then it can happen that *a Po b* and *b Po a*. What is the right representation to translate in the ontology to generate? How to prioritise in non symmetric incoherent relations?

Also we can have two different aggregations of attributes for the same concept, so what is the best characteristic group of properties for each concept derivable from a model instance?

Furthermore semantic and syntactic relations can link disjoint concepts, how to be sure to find the right correspondence? Once again, given a set of concepts of the model what are the most representative concepts among them?

These are just some of the possible problems and questions we incur when trying to generate automatically an ontology or even simply when providing information to matching/merging systems. Most of known solutions rely on hypothetic external reference knowledge that is rare to have and often inadequate for the domain. So common approaches based on only two input sources can not take advantage from information that can arise from multiple sources. Conversely, our approach permits to point out one solution that statistically arises among others.

This is the reason why we decided to introduce some information capable of providing hints for selecting the better decision to undertake in situations of ambiguity. Delving into this area, the choice of a key measure that well fit the problem is absolutely not a trivial choice, above all if we target large scale and evolving environment. At present we maintain two pieces of information based on the occurrence and attendance for both concepts and relations. Albeit few, these basic elements supply useful information, resolve some of the most popular measures like TF (Term Frequency) [157], provide an absolute/relative weight measure for each element of the model and thus for the ontology to build.

For this we define the **attendance** for an element of the model as the number of input sources where the element appears. Moreover, we define the **occurrences** as the number of times it appears in all input sources. Initially we stored an occurrence value for each input source element from which an element has been extracted, but in large scale this can require a huge quantity of data to maintain without a real proved benefit from the detail of the information. About the measure to compute with these information, the most classical measure is TF-IDF (Term Frequency –Inverse Document Frequency) [157] measure that requires the frequencies per document globally to be calculated. But this measure tends to promote the less common and thus distinctive concept from a corpus rather then the most representative. So we have discarded its adoption and calculated the frequency value for a concept a belonging to the set of concepts O of the Model as follows:

$$F_a = (\#a) * Att(a) / Max(\#O) * Max(Att(O))$$

- Where *Max(#O)* and *Max(Att(O))* stand respectively for the higher value of occurrence and attendance of the model instance.

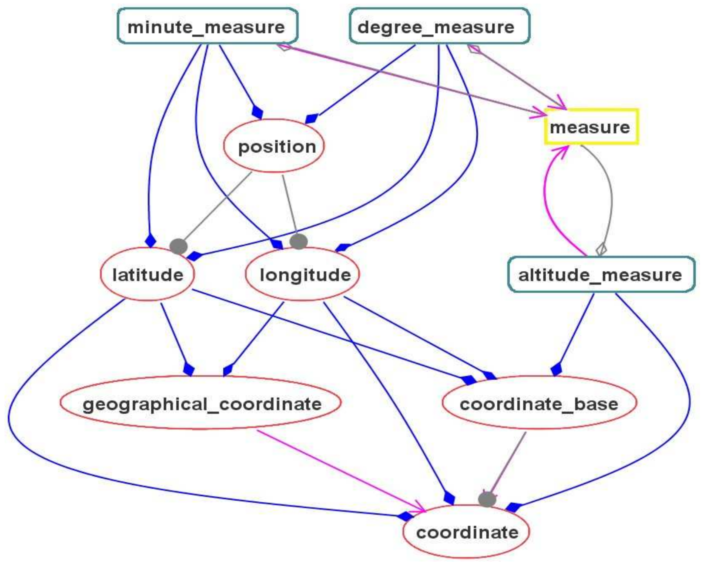
With these elements we are still far from the optimal solution of the problem, but our model still represents an improvement w.r.t. other approaches proposed. Further improvements of our approach could be the adoption of rank measures like PageRank [158] and symRank [159], because they lend themselves well to the relatively high number of relationships we define. These relationships can be seen as in-links and out-links for a concept and thus make ranking formulas applicable.

3.1.6 Graphical Notation

We have defined a SDMO graphical representation that provides a global view of concepts organization with their relationships. Figure 3.8 illustrates the graphical syntax we use to describe a SDMO schema.

In first instance, by supporting subtype and property relationships, the model achieves a *structurally object-oriented* model, i.e., one which is able to represent relationships, data types and attributes that are found in Object-Oriented languages like UML. Secondly by supporting rich semantic and syntax relationships the model fulfils the requirements of a *semantic model*, i.e., one which is able to represent relationships that are found in ontological language like OWL. Thus the model has sufficient expressiveness to model information extracted from UML and OWL.

As we can see from Figure 3.8, a graphical representation of the model could become complex because of the number of relationships. However, as we will show in the implementation section, it is possible to provide simple selection criteria for concepts and associations to visualize the model at different scales. In other words, the system permits a scalable view even very appropriate for big models.



Symbol	Name
label	Class Concept
label	Property Concept
label	Data type Concept
—◇	Data type relationship
—◆	Property relationship
—●	Is a relationship
●—●	Equivalence relationship
→	Shared term relationship
↔	Syntax relationship
—	Synonymy relationship

Figure 3.8 – SDM Graphical Representation

3.2 SDMO to OWL

As a first step, it is important to point out the difference between our utilization of SDMO models and OWL ontology. SDMO serves as a transition conceptual model between input sources and a global ontology. SDMO is useful to represent application data to handle in order to optimize and automate as much as possible similarity discovery and primary concept definitions. OWL meanwhile serves to represent the generated final ontology with consistent axioms; it does not help us to manipulate the data. In this context it is clear that all represented data descriptions with SDMO are not necessarily preserved when exported to OWL. And of course if we target the inverse transformation, OWL to SDMO there is information that could be lost. However SDMO contains all necessary information to produce a first basic ontology. For developing the OWL compatibility, we have firstly defined an OWL generic model that corresponds as much as possible to SDMO.

The following subsections present the derived OWL model and the resulting mapping technique from SDMO constructs to OWL ones.

3.2.1 OWL Model Definition

Before detailing the interpretation of SDMO as OWL ontology, the subsections below present some differences and consequent problems that appear with a direct mapping to OWL constructs. Next, we motivate our decisions and present the OWL representation of SDMO.

3.2.1.1 Different Abstraction Level Problem

One SDMO feature is its own skill to represent metadata thus providing meta-model information. Indeed, what it proposes is a way of representing concepts in order to obtain a model of the similarity of data in a domain. In this context it allows to create classes, properties, types of data and relationships. However, the model represents relationships at different levels of abstraction. To understand this point, let us give an example. Considering the concept class *Enterprise* simply defined with SDMO structural relations referring to the following concepts *Person, Address*, *Activity*, as follows:

$$\text{Enterprise} \equiv \exists\text{hasDirector.Person} \sqcap \exists\text{hasOffice\_location.Address} \sqcap \exists\text{hasActivity.Activity}$$

Through this axiom we assume that the concept *Enterprise* has, for example, the *hasDirector* object property which is an instance of the concept class *Person*. This is still true with concept instances:

$$\text{Enterprise(ORANGE\_LABS)} \equiv \{\text{hasDirector(THIERRY\_BONHOMME)} \sqcap \text{hasOffice\_location(42\_RUE\_DES\_COUTURES\_CAEN\_FRANCE)} \sqcap \text{hasActivity(RESEARCH)}\}$$

This means that *Thierry Bonhomme,* instance of *Person* is the director of *OrangeLabs*. Now if we consider the concept *Company* as synonym for *Enterprise*, the semantic relationship is valid between these two concept names but not between instances of the two concepts. *OrangeLabs* and *Ford Motor* are not synonyms, while in another case we could say that instances are synonyms for a concept, like *car* and *vehicle* for *Manufacture_product.* What we mean here is that some SDMO relations are designed for metadata concepts relations and not for their individuals, that can be considered as meta-metadata for individuals of an ontology. Thus, while some relationships are maintained between concepts others do not stand for the latter because of a different level of abstraction, thus:

$$\text{Rsyns(Enterprise, Company)} \nvDash \text{Rsyns(ORANGE\_LABS, FORD\_MOTOR)}.$$

A way to resolve this problem with OWL is to interpret the synonym relationship using the *owl:equivalentClass* relation. Indeed OWL permits to handle this situation and a reasoner can deduce that any individual that is an instance of *Enterprise* is also an instance of *Company* and vice versa, without entailing the same identical relation between individuals, just they belong to a same subset of individuals. Nevertheless it can induce to other possible errors. If we add into the ontology the concepts *Institution* or again *Fellowship* which are synonyms for *Company,* being equivalence a transitive function a problem arise. In this case a reasoner will deduce that being:

$$(\text{Enterprise} \equiv \text{Company}) \wedge (\text{Company} \equiv \text{Institution}) \Rightarrow (\text{Enterprise} \equiv \text{Institution})$$

Which is false in several cases, because if the *University of Versailles* can be considered as a kind of (educational) *Company* it is surely not an *Enterprise.*

The same kind of problem also subsists with SDMO syntax relationships. For example in our use case we found *PO* as abbreviation for *Purchase Order* and *Post Office* concept names, therefore the same substring can link two different concepts and thus incur into the similar abstraction level problem and error described above. What we mean here is that we would like to maintain the discovered information that two concepts with a different name just share the same abbreviation to put in the ontology. Because, even if it is irrelevant for the real meaning of the ontology, it can be a relevant information for a semantic matcher system. But we have not a correct ontology relation that is able to maintain such information without incurring into wrong interpretations on concepts individuals. Moreover it is of our advice that the adoption of word contractions or substrings as concept names is self-contradictory with the definition of ontology that requires well formed structures and semantics. That means that as general rule we try to use only real terms as concept names, thus *PO* should find another accommodation rather be the concept name.

3.2.1.2 One vs. Two Ontologies

There are different applicable solutions to resolve the translation of SDMO to OWL w.r.t. the problems pointed out above. One of them can be to generate more ontologies in order to separate the different abstraction level. For example, with two ontologies, the first one can constitute the meta-model whose contents are concepts and high-level relations (syntactic, semantic, lexical and sources). For those familiar with WordNet it could be similar to the WordNet RDF/OWL representation [160].

The structure of the second ontology is generated as an instance of the first, transforming each instance class, property or data type, by keeping, this time, structural relationships.

Figure 3.9 illustrates an example of using two ontologies to represent an SDMO model. Figure 3.9 a) shows the metadata ontology with an example of individuals, while Figure 3.9 b) shows the second ontology generated from elements of the first ontology with an example of its own individuals.

3.2.1.3 Enforcing OWL Annotations

The second solution we considered has been to create only one ontology and manage uncertain and meta relations as OWL annotations. Indeed OWL allows advanced annotations on classes, properties, individuals and ontology headers, with the *owl:AnnotationProperty* construct. For example it is possible to define some axioms capable of inferring precise domain and range for the annotation.

Listing 3.1 illustrates an example of an owl annotation property that can be used to represent the synonymy relationship. Without delving into details of the OWL syntax, it shows that it is a symmetric function, used as annotation, and that it is a property relation between two concepts belonging to the *Concept* class or its subclasses.

Although this solution well fit our needs, it enforces domain and range within annotations and this kind of definition is not allowed for OWL DL ontologies. Consequently, we are obliged to reduce the annotation property statement without defining sub-properties and domain/range constraints. Thus, the control of declaration correctness shall be left directly to the ontology construction algorithm. Moreover the object of an annotation property must be either a data literal, or a URI reference, or an individual.

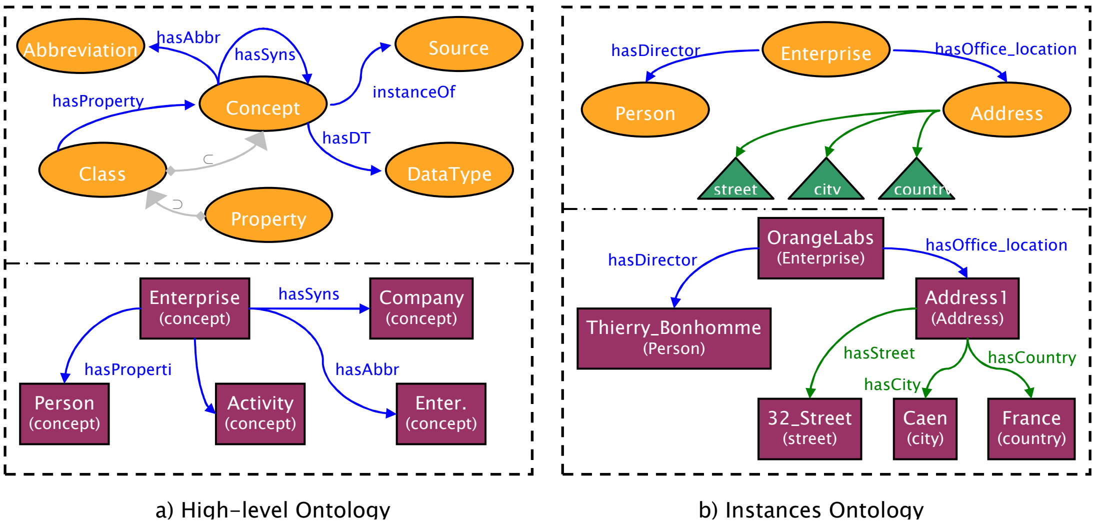


Figure 3.9 – Double ontology representation of SDMO

```
<owl:SymmetricProperty rdf:about="#synonymOf">
  <rdf:type rdf:resource="#AnnotationProperty"/>
  <rdf:type rdf:resource="#ObjectProperty"/>
  <rdfs:subPropertyOf rdf:resource="#semantic"/>
  <rdfs:domain rdf:resource="#Concept"/>
  <rdfs:range rdf:resource="#Concept"/>
</owl:SymmetricProperty>
```

Listing 3.1 – Example of advanced OWL annotation property

3.2.1.4 Extending OWL

Another approach to obtain a full representation for SDMO could be to extend OWL expressivity. The extensions should be able to manage uncertain relations between meta-concepts, conditional properties, and significant groups of properties. A clearer separation between meta information and real instances should provide more way to reasoning with ontology entities and more flexible knowledge.

For the latter in literature already exist some proposal to extend OWL to more probabilistic approach, like Ding and Peng that propose a probabilistic extension to OWL that models uncertainty of class memberships [161]. Using Bayesian Networks, they are able to model conditional class membership probabilities. However uncertainty of other relationships is not supported. PR-OWL is a more general probabilistic extension to OWL with which uncertainty of relationships between concepts can be expressed [162]. A similar approach is proposed by Pool et al. who argue for extending OWL due its widespread use and tool support and the simplicity to implement probabilistic extensions [163].

Nevertheless, as also argued in [163], these changes could delve into OWL decidability capacity and a high representational complexity. This discussion is not the topics of this thesis. Thus, for the moment, we limit our SDMO representation to existing OWL constructs. In particular, current results are largely theoretic and development tools are hardly available.

3.2.2 An OWL Representation for SDMO

Among the representations of SDMO OWL models presented above, we adopted the single file one with specific annotations for SDMO relations that cannot be directly represented with OWL constructs. This choice has the advantages of: (i) Making possible a direct representation of real concept instances as individuals for the ontology. (ii) Providing a complete representation of SDMO in a sole file, that can simplify maintenance. We know that reasoners do not consider annotation properties, but we plan to use them in a custom reasoner, or at least via custom inference rules.

The first element to describe in order to understand how the mapping works, is the base file we use to build the ontology (provided in Appendix A) that represents the basis of our model representation. The top level concepts element in the ontology is an OWL class named `sdm:Concept`, detailed in Table 3.1; all SDM classes and properties are subclasses of this one.

Relationships between concepts are split into four main categories (according to the categories of the SDM model): semantic, syntactic, structural and source. We create an OWL Object Property for each of these categories; they represent the top-level object properties in the ontology.

For the semantic relation *synonym of* we define an annotation property named `sdm:synonymOf`. This property allows the linkage between two classes or two properties to specify that they are synonyms. More details about semantic relations are provided in Table 3.2.

Table 3.3 depicts syntax and syntactic relations. More in detail for the syntax relation *has abbreviations* we define an enumerated class named `sdm:Abbreviation` (see Table 3.1 for this class) which will be the root of all abbreviations in the ontology. It is an enumerated class as it is defined by the set of all its individuals (which will be created during the export process from SDMO instances to OWL ontologies) that could be assimilated to be a group element. To link a concept to its abbreviations, we define another annotation property `sdm:hasAbbreviations` that let us say that a sub-class of concept has a set of abbreviations. This is similar for syntactic relations.

Concepts	
SDMO	**OWL**
General Concept	Meta Class: *sdmo:Concept* Name: *Concept* Sub class of: *owl:Thing*
Abbreviations	Enumerated class: *sdmo:Abbreviation* Name: *Abbreviation* Sub class of: *owl:Thing*
Instances	Enumerated class: *sdmo:Instance* Name: *Instance* Sub class of: *owl:Thing*
Classes	Class: *owl:class* Name: *SDMO_class_name* Sub class of: *sdmo:Concept*
Properties	Class: *owl:class* Name: *SDMO_class_name* Sub class of: *sdmo:Concept*
Datatypes	Class: *owl:DatatypeProperty* Name: *SDMO_concept_name* Sub property of: *sdmo:hasDatatype*

Table 3.1 – OWL representation of basic SDMO concepts

Semantic relations	
SDMO	**OWL**
Semantic (meta-property)	Meta Object property: *sdmo:Semantic* Name: *Semantic*
Synonym Of (synonym)	Symmetric property: *sdmo:synonymOf* Name: *synonymOf* Sub property of(*): *sdmo:semantic* Domain(*): *sdmo:Concept* Range(*): *sdmo:Concept*
Shared Term	Symmetric property: *sdmo:sharedTermWith* Name: *sharedTermWith* Sub property of(*): *sdmo:semantic* Domain(*): *sdmo:Concept* Range(*): *sdmo:Concept*

Elements marked with (*) are not allowed in OWL DL

Table 3.2 – OWL representation of basic SDMO semantic relations

Syntax relations	
SDMO	**OWL**
Syntax (meta-property)	Meta Object property: *sdmo:Syntax* Name: *syntax*
Abbreviations	Annotation Property: *sdmo:hasAbbreviation* name: *hasAbbreviation* Sub property of(*): *sdmo:syntax* Domain(*): *sdmo:Concept* Range(*): *sdmo:Abbreviation* Inverse property(*): *sdmo:isAbbreviationOf*
	Annotation Property: *sdmo:isAbbreviationOf* name: *isAbbreviation* Sub property of(*): *sdmo:syntax* Domain(*): *sdmo:Concept* Range(*): *sdmo:Abbreviation* Inverse property(*): *sdmo:hasAbbreviation*
Syntactic	Simmetric Property: *sdmo:linguisticSimilarity* name: *linguisticSimilarity* Sub property of(*): *sdmo:syntax* Domain(*): *sdmo:Concept* Range(*): *sdmo:Concept*
	Simmetric Property: *sdmo:nGramWith* name: *nGramWith* Sub property of(*): *sdmo:linguisticSimilarity* Algorithm details: rdfs:isDefinedBy, rdfs:comment, rdfs:label
	Simmetric Property: *sdmo:hasCommonStem* name: h*asCommonStem* Sub property of(*): *sdmo:linguisticSimilarity* Algorithm details: rdfs:isDefinedBy, rdfs:comment, rdfs:label

Elements marked with (*) are not allowed in OWL DL

Table 3.3 – OWL representation of basic SDMO syntax relations

For the structural relation *has property*, we define an Object Property named `sdm:hasProperty`. This OWL object property is the top level node for all properties relations. The structural relation *property of* is defined by the object property `sdm:isPropertyOf` and as inverse property of the `sdm:hasProperty`. The relation "has datatype" if defined by the datatype property `sdm:hasDatatype`. It is used as the root of all datatype properties of our model. All these elements are depicted in Table 3.4.

Structural relations	
SDMO	**OWL**
Structural (meta-property)	Meta Object property: *sdmo:structural* Name: *Structural*
Properties	Object property: *sdmo:hasProperty* Name: *hasProperty* Sub property of: *sdmo:structural* Domain: *sdmo:Concept* Range: *sdmo:Concept* Inverse of: *sdmo:isPropertyOf*
PropertyOf	Object property: *sdmo:isPropertyOf* Name: *isPropertyOf* Sub property of: *sdmo:structural* Domain: *sdmo:Concept* Range: *sdmo:Concept* Inverse of: *sdmo:hasProperty*
hasDatatypes	Data Type: *sdmo:hasDatatypes* Name: *hasDatatypes* Sub property of: *sdmo:structural*

Elements marked with (*) are not allowed in OWL DL

Table 3.4 – OWL representation of basic SDMO structural relations

Finally, for source relations illustrated in Table 3.5, we define the annotation property `sdm:instanceOf`. This property allows us to link a concept class to a sub-class of the enumerated class `sdm:Instance`. Those classes represent the different instance names and details on source elements (specification, file, database, ...) in which we found the original form of the concept. As for abbreviations, instances are declared as Individuals defining their class. Moreover three annotation properties are provided to maintain statistical information and one to maintain a link with the source document.

Source relations	
SDMO	**OWL**
Source	Object property: *sdmo:Source* Name: Source
InstanceOf	Annotation Property: *sdmo:instanceOf* name: *InstanceOf* Sub property of(*): *sdmo:source* Domain(*): *sdmo:Concept* Range(*): *sdmo:Instance*
Source document	Annotation property: *rdfs:seeAlso*
Attendance	Annotation property: *sdm:trustAttendance*
Counter	Annotation property: *sdm:trustCounter*
Number of Input sources	Annotation property: *sdm:numberOfSources*

Elements marked with (*) are not allowed in OWL DL

Table 3.5 – OWL representation of basic SDMO source relations

Appendix A provides a more detailed table with all defined SDMO representations with some example and the complete OWL model file.

3.2.3 Some Concerns about Expressivity of SDMO-OWL

We recall the fact that our model has been initially designed to represent and maintain information automatically extracted from T-Box elements (see Section 1.1.3) only. This is motivated by the fact that in the domain we target it is not always possible to get a consistent set of A-Box, instances, from which extract information and deduce more powerful expressive knowledge. Let us take a simple example that motivates this choice. If we want to extract knowledge to produce an ontology from a B2B exchange between a bank with its clients, it is not realistic believe that the partners agree that we look at and mine their personal data. So our efforts are focused only on meta-data freely available without any privacy violation.

Nevertheless the model we defined is able to produce relatively expressive ontologies. Indeed with some attention to limit the annotation properties and to produce only tree like hierarchy relations, our model belongs to the OWL-DL family. Without care, we could generate OWL-Full ontologies that risks to become complex for decidability reasons. For that we generate annotations without range, domain and sub property information, and we also reduce the ontology to a tree-like structure removing less probable heritages between classes and object properties.

More properties could be added with cardinalities that currently have not been considered yet, like `owl:functionalProperty`, `owl:cardinality`, `owl:maxCardinality` and `owl:minCardinality`.

More precisely, following DL naming convention presented in Table 1.1, our ontology corresponds to a *S*HOI*N*Q*F*(D) expressivity, where italic elements refer some limitations we have, like concept negation that is difficult to discover with only T-Box basic information, and NF are dependent to the integration of cardinality information.

3.3 Related Works

As mentioned in Section 1.3 our goal is to produce a pre-alignment representation of "convincing" matchings, which we can also define as candidate alignments. Throughout our study we observed that the most part of matching systems use an intermediary data representation before producing final alignments or mappings. However, even if more than 50 systems exist right now, descriptions of such a model are still rare and we did not find any complete and reusable model conformant to our goal. Among those that already exist, the concept theory has been vector of inspiration for the SDMO definition. The first one is the conceptual graph theory which is a notation for logic based on the existential graphs and the semantic networks of artificial intelligence. In the first paper published on Conceptual Graphs [164] the author applied them to a wide range of topics in artificial intelligence, computer science, and cognitive science.

In the following subsections, we analyze works that in some extents treat of semantic data models, conceptualization of a domain and ontology matching. We finish with the most recent Linked Open Data community presenting some common problems and how our approach could improve the linking of data in an open environment.

3.3.1 Existing Data Model Databases Oriented

Several **semantic data model** were proposed in the 80's for modelling databases. They were basically extensions of the Entity/Relationship data models or abstraction of the object model for databases. We can consider in this category examples as SDM [150], IFO [151] and Morse [165]. The model that we propose is targeted towards modelling and memorizing efficiently dynamic ontologies. It has a finer meaning granularity (e.g., various types of relationships) than classical SDM. In addition, we believe it is important to show that SDMs of the 80's can be enriched to support ontologies.

3.3.2 Formal Concept Analysis

To our extent an interesting application of conceptual graphs applied as model for ontology element is **Formal Concept Analysis** (FCA). We recall the basics of FCA as far as they are needed for this document. A more extensive overview is given in [166] and its applications in [167] and [166].

To allow a mathematical description of concepts as being composed of *extensions* and *intensions*, FCA starts with a formal context defined as a triple $\mathcal{K} := (\mathcal{G},\mathcal{M},I)$, where $\mathcal{G}$ is a set of objects, $\mathcal{M}$ is a set of attributes, and $\mathcal{I}$ is a binary relation between $\mathcal{G}$ and $\mathcal{M}$ (i. e. $\mathcal{I} \subset \mathcal{G} \times \mathcal{M}$). $(g,m) \in \mathcal{I}$ is read "*object g has attribute m*". A *formal concept* of a *formal context* $(\mathcal{G},\mathcal{M},\mathcal{I})$ is defined as a pair $(\mathcal{A},\mathcal{B})$ with $\mathcal{A} \subset \mathcal{G}$, $B \subset M$. The sets $\mathcal{A}$ and $\mathcal{B}$ are called the *extent* and the *intent* of the formal concept $(\mathcal{A},\mathcal{B})$. The *subconcept-superconcept* relation is formalized by $(\mathcal{A}_1,\mathcal{B}_1) \leq (\mathcal{A}_2,\mathcal{B}_2) \Leftrightarrow \mathcal{A}_1 \subset \mathcal{A}_2$ $(\Leftrightarrow \mathcal{B}_2 \subset \mathcal{B}_1)$. The set of all formal concepts of a context $\mathcal{K}$ together with the partial order $\leq$ is always a complete lattice (i.e. for each set of formal concepts, there is always a greatest common *subconcept* and at least common *superconcept*) called the concept lattice of $\mathcal{K}$ and denoted by $\mathcal{B}(\mathcal{K})$.

The nodes in Figure 3.10 represent formal concepts. It summarizes the relationship between Concept $\mathcal{A}$ and Concept $\mathcal{B}$. Concept $\mathcal{B}$ is a subconcept of Concept $\mathcal{A}$ because the extension of Concept $\mathcal{B}$ is a subset of the extension of Concept $\mathcal{A}$ and the intension of Concept $\mathcal{B}$ is a superset of the intension of Concept $\mathcal{A}$. All edges in the line diagram of a concept lattice represent this subconcept-superconcept relation. The top and bottom concepts in a concept lattice are special, the top concept has all formal objects in its extension, while its intension is often empty but not necessarily.

An example of FCA approach applied to ontology merging is **FCA Merge** [52]. In their approach lattice nodes are formal concept consisting of all attributes, called the *intent* of the lattice and corresponding to ontology concepts, while the *extent* of the lattice are given by the so called *Keys* which are likewise super concepts derived by the description of attributes of the node.

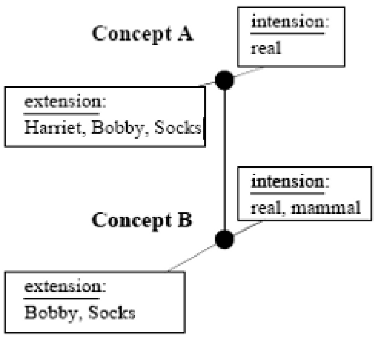


Figure 3.10 – A subconcept-superconcept relation in FCA

To our scope the implementation of this model is not complete because some retrieved concepts are pruned in order to maintain only the main common concepts. This means that this model can not be used to explore sources incrementally. Indeed it maintains only a sub-set of retrieved input concept at a given instant. Moreover concept naming, which is a large problem when merging different sources, is mainly deferred to users in the sense that, if more than one attribute is associated to the same formal concept, a user is needed to choose between the different names.

This approach as well the **Fuzzy FCA** extension proposed by Quan *et al.* [168] could be adopted in our work but it needs some extensions aiming at our environment with multiple inputs and algorithms reducing human intervention. Furthermore, because it impacts the construction algorithm, the model needs also a sort of reengineering to maintain overlapping concepts and limit the loss of important information to be reused.

3.3.3 The Canonical Conceptual Model

A **Canonical Conceptual Model** (CCM) is proposed by [74] to represent XML data. The model is an adapted mix of two others models: The conceptual basis of the canonical model comes from ORM/NIAM (Object with Role Model / Natural language Information Analysis Method) [76]. Most of the graphical notation comes from Extended Entity-Relationships (ERR) [77] model to support semi-structured data representation. Figure 3.11 presents a corresponding schema in the CCM which contains *non-lexical* and *lexical* concepts. Where *non-lexical* concepts (solid rectangle) model information that is composed by other information, like *Authors*, while *lexical concepts* (dotted rectangle) model information that has a direct associated value, like *Year.* Furthermore lexical concepts can be specialized to *enumerated lexical concepts*, that additionally include a *value constraint*, like *Type*. A value constraint denotes an enumeration of permitted values. A *root concept* (thick rectangle) is provided as a type of non-lexical concept to represent the root object of a semi-structured object hierarchy, like *Conference*.

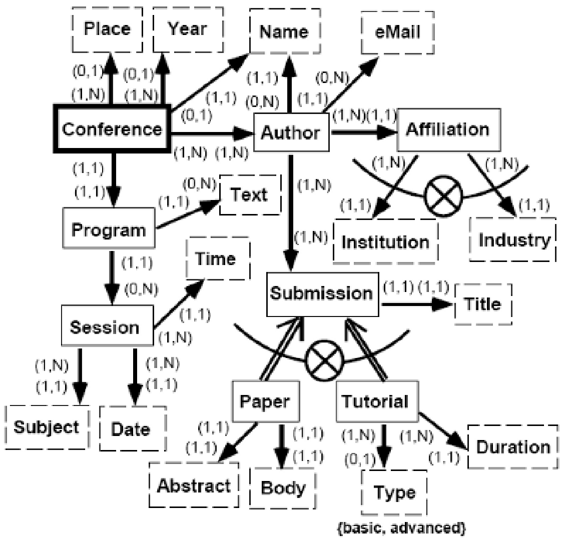


Figure 3.11 – Example of graphical representation of a CCM Schema

Exclusion constraints, borrowed from ORM/NIAM and represented by an "X" circled graphical notation, define disjoint relationships (suitable to support heterogeneous relationships of semi-structured objects). An example is *Affiliation* that may be related to an *Institution* or an *Industry*.

Formally the conceptual schema is a 4-uple $s = <\mathcal{NL}, \mathcal{L}, \mathcal{R}, \mathcal{EC}>$ where $\mathcal{NL}$ is the set of non-lexical concepts, $\mathcal{L}$ is the set of lexical concepts, $\mathcal{R}$ is the set of binary relationships between concepts and $\mathcal{EC}$ is the set of exclusion constraints among relationships.

The approach proposed here encounters our interest for the proved conceptualisation of XML schemata, the completeness and the incremental generation process of this model. However, the model is limited in relationships expressivity, only composition and inheritance are defined. Furthermore, as far as we know, the problem of multiple input sources is not addressed in practice and still remains theoretical.

3.3.4 Conceptual representation with Extended X-Formalism

The *Extended X-Formalism* (EXF) presented in [96] [97] is a conceptual model that maps features of different XML schemas to highlight classes of concepts and their relationships. The most important features are extracted from proposed XML schemas as input corpora.

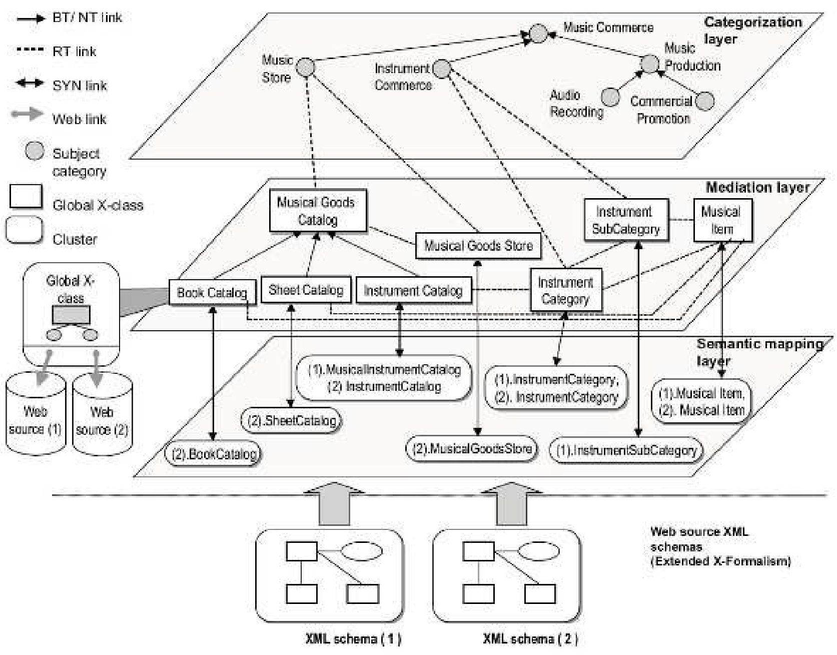


Figure 3.12 – An example of three-layer ontology derived from the EXF Frame model

In the EXF model, a set of concepts is provided, namely *XClass*, *XType*, and *EXF frame*, capable of describing at a high level several source features. Intuitively, an *XClass* represents a set of entities that have a common structure and correspond to an element declaration whose type is complex. Each *XClass* is characterized by a name, a type, a set of properties, and a set of attributes. An *XType* is a user-defined type and corresponds to a type declaration. Each *XType* is characterized by a name, a set of properties, and a set of attributes. An EXF frame represents the content of an XML schema document and is composed of a set of *XClasses* and *XTypes*. Figure 3.12 shows an example of the resulting so called *three-layer ontology* derived from XML sources with EXF model adoption. The semantic links defined in the ontology are SYN (synonymy), BT (hyperonymy) and its inverse NT (hyponymy), and RT (positive association). According to their possible use in the ontology, semantic links are classified in *intra-layer links* and *inter-layer links*. The *X-classes* modelling the data sources are grouped into clusters at the semantic mapping layer. The *global X-classes* at the mediation layer are constructed in a specific integration step and provide reconciled representations for each cluster.

3.3.5 The Logical Data Model Ontology

The Logical Data Model (LDM) Ontology [169] is used in the STASIS project as a Neutral representational Format (SNF) to represent incoming information into their mapping environment from an external schema. Currently the system already provides a transformation of Relational Database, XML-schema, EDIFACT, and Flat File formalisms. The underling LDM formalism is itself a conceptual model to obtain a unified representation of several data models, needed to abstract from syntactical aspects of a specific data model.

In this way, LDM Ontology corresponds to *a graph with directed labelled edges.* It proposes the following types of concepts:

- The nodes of the graph, which are partitioned in *SimpleNodes* and *ComplexNodes*.
- The edges of the graph, which represent Relationships between Nodes.

The following types of Relationships can exist:

- Reference: A Reference is a directed labeled edge between *ComplexNodes*.
- Identification: A *ComplexNode* can be identified by a *SimpleNode* or a set of *SimpleNodes*.
- Containment: A *ComplexNode* can contain other Nodes, *SimpleNodes* and/or *ComplexNodes*.
- Qualification: A Node can be qualified by a *SimpleNode*.
- Inheritance: Inheritance can exist between *ComplexNodes*.

The LDM allows the representation of classes/concepts (sets of individuals), relationships (binary predicates relating individuals), and attributes (binary predicates relating individuals with values such as integers and strings). Relationships are subject to constraints such as specification of domain and range, plus cardinality constraints.

The formalization of this model is based on a transformation in a LDM_OWL ontology that finally is used as basis to map two different schemas. An overview of the concepts and their relations in the ontology is shown in Figure 3.13. A detailed description of the LDM Ontology is provided in [170].

Similarly to our approach, the authors use OWL as serialization format, but their model does not provide dynamic integration of sources. Moreover relationships are limited to either hierarchical or structural relationships.

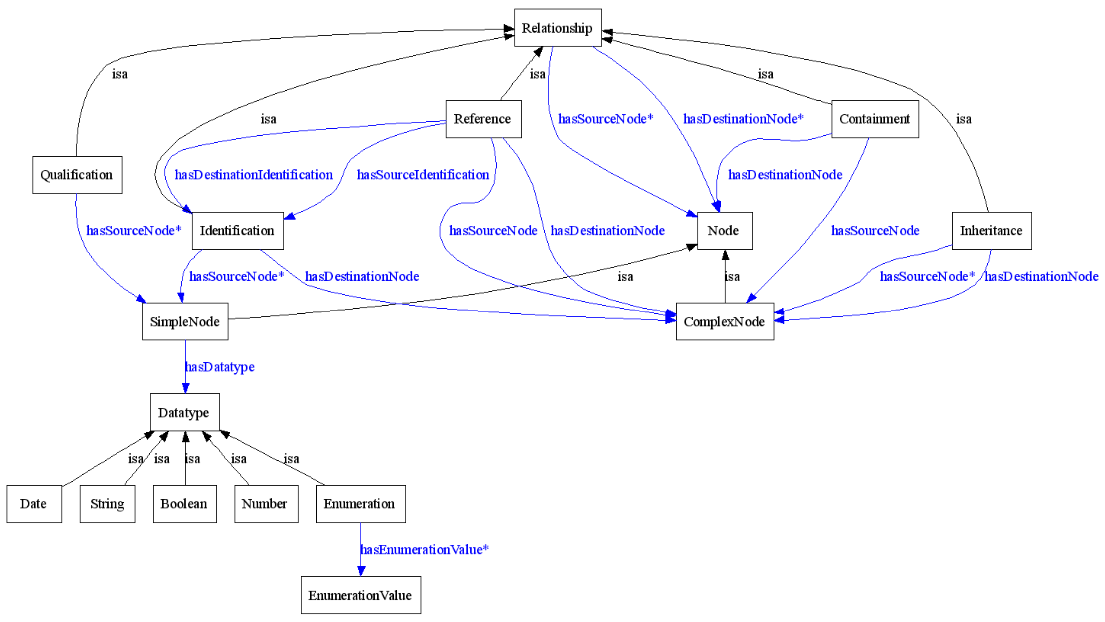


Figure 3.13 – The LDM_OWL ontology

3.3.6 Linked Open Data

Linked Data [171] assumes that with the growing of Semantic Web technology stack, and by the publication of large datasets according to W3C RDF/OWL formalisms, other than documents, the Web can be explored by a person or machine. It implies that by the adoption of these links between data items from different data sources, Web site can be enriched automatically. But as exposed in [172] matching Web resources based on simple URIs, similarly to string matching, can cause disambiguation problems.

In this sense the UMBEL project [173] aims to provide a lightweight structure of subject concepts as a reference to what Web content or data "is about"; and to define a variety of binding protocols for different Web data formats to map to this backbone. The model we target can fill this need, because it naturally provides this backbone with structured concepts and scalable relationships.

3.3.7 Synthesis

We provide here a short evaluation of models seen above. This analysis does not aims a complete evaluation of each model but simply furnishes elements that we have considered necessary to our use case and to our scope. For this Table 3.6 summarizes our considerations on existing models. Elements used for the general evaluation (lines of the table) are as follows:

- *Adapted to XML* – aims to evaluate if the model is adapted to XML input sources and used to swap their content information into ontologies;
- *Concepts nature* – expresses the possibility to choose the nature of a concept depending on its behaviour (like class or attribute);

- *Structural relations* – says if it is possible to define a hierarchy among concepts (like subclasses or even property);
- *Semantic relations* – says if it is possible to define semantic relations among concepts;
- *Generic relations* – says if it is possible to define other types than structural or semantics of relations among concepts;
- *Complete* – aims to evaluate the possibility of a model to maintain and store information coming from different sources even when some information has been merged. Also if sources can be added incrementally or if the model requires a complete regeneration each time a source is added;
- *Dynamic* – tries to evaluate the possibility to change the nature of a concept (e.g. following the insertion of new sources with different granularity);
- *OWL serialization* – states if the model provides an OWL serialization formalism;
- *Specification* – says if the model has already been specified and available;
- *Implementation* – says if the model has already been implemented;
- *Automation* – simply says if somewhere in the model generation process, human intervention is required.

As the table shows all models have characteristics that meet our scope. However few of them provide a dynamic behaviour and tend to manage only static information and exact concepts, relations and correspondences. Moreover even when it is adaptable there is not available implementation or clear specifications.

	Databases Oriented	Formal Concept Analysis	Canonical Conceptual Model	Extended X-Formalism	Logical Data Model	UMBEL
Adapted to XML		✓	✓	✓	✓	✓
Concepts nature	✓	✓		✓	✓	✓
Structural relations	✓	✓		✓	✓	✓
Semantic relations				✓		✓
Generic relations						✓
Complete			✓	✓		✓
Dynamic			?			✓
OWL serialization					✓	✓
Implementation	✓	✓			✓	
Specification	✓	✓	✓	✓	✓	
Automation	✓					✓

Table 3.6 – Overall evaluation of Conceptual Models

We have not presented here two other models which are the Core Component Model [139], which is probably one of the widely adopted model in B2B, and the Dynamic Object Model [149]. This because they are generic models which are not linked to ontology construction. However these have also influenced our vision and needs of the model to build because, the first one is really close to the domain that we target and thus really close to several sources we consider. The second has an interesting feature, which is the fact that the concept's nature is not frozen but derived dynamically directly from behaviours of the element in the model. This solution attracted our interest and has been adopted in our model.

3.4 Conclusion

In this Chapter we have defined the intermediary conceptual model as an important part of the architecture defined in our approach for the automatic generation of ontology derived by XML Schemas.

We have described and defined SDMO, the Semantic Data Model for Ontology, and showed that it improves existing solutions. Indeed it can furnish valid background knowledge for the automatic construction of ontology and for semantic matcher systems. This thanks to the rich expressivity of the supported relationships among elements of the model. It provides not only natural percept of real world modelization (like "is a" relations), but also specific relations for matching concept names similarities at different levels, meaning and linguistics. In addition the model also provides a way to maintain the frequency for concepts and relations, which permits to unveil and resolve some incoherencies and ambiguities that often arise from the merging of heterogeneous sources.

Moreover we provide a complete mapping of our model to OWL in order to be able to derive automatically an ontology from an SDMO instance.

Limitation of our model can be found in still limited expressivity of other aspects that other models can have, like cardinality and well defined disjoint set of elements, however it still remains open and flexible enough to be extended if needed is.

In the next Chapter we already use our model in a real context, which is the conceptualization of information extracted from XML Schemas. We will show its applicability for maintaining large quantity of information into an aggregated and organized view to be reused …in Chapter 5.

Chapter 4.

Mining XML Schemas to Extract Conceptual Knowledge

As seen in Chapter 2, XML [102] is the formalism that in the last decade has reached the largest consensus among all standard bodies, until to become the de facto standard format for B2B messages definition. Several reasons can motivate this choice, the first of them being that it provides both human readable and machine interpretable format at the same time. Another reason is its simplicity and suppleness of usage that well fit the great part of application information exchange requirements. Furthermore the introduction of DTD and XSD formalisms permits a good separation between meta-data information and instances containing the real data to be exchanged.

Nevertheless the XML formalism still remains in a certain sense too much open and provides a lot of dialects that tend to overload its basic usage and meanings. This is the reason why we can have interoperability problems even when two applications adopt XML as formalization of input and output messages.

Without delving into philosophical dissertations about how these differences arose, throughout this Chapter we provide a pragmatic view and analysis of XML B2B specifications and practices. Our unique goal is finding and demonstrating how XML Schemas can be exploited to extract DL assertions. Therefore, they can be used to produce automatically a first skeleton of ontology. We show that it is not a simple transformation, but that this operation requires precise attention on design practices.

From these considerations we describe how XML Schema sources can be exploited to generate automatically a specific vocabulary of terms representing the sources and a basic taxonomy of unstructured concepts. After this first step, we also provide a complete formalization of the transformation of XML Schemas metadata into our SDMO model and we compare achieved transformations with those of other systems.

This Chapter is organized as follows. Firstly, we highlight some benefits we have by using XML tree structure with respect to other formats. Then, we recall the main XML components focusing on XML Schema. In Section 4.2, we provide some figures on the B2B specifications seen as XML sources and we develop an analysis of B2B XML design practices. In Section 4.3, we present results we get from the automatic generation of a taxonomy from the collected sources. Then, we focus on the ability of our system to provide correct semantics. In Section 4.3, we detail the XML Schema conceptualisation using SDMO, we define rules for the mapping of schemas to SDMO, and we provide some elements to evaluate our transformations. Section 4.5 presents some measures we adopted to decide if a source XSD document is compatible with our system in order to be able to extract useful information for the ontology to build. Finally, Section 4.6 concludes this Chapter with a recall about the more relevant contributions and results we got with the conceptualization of B2B XML Schema sources.

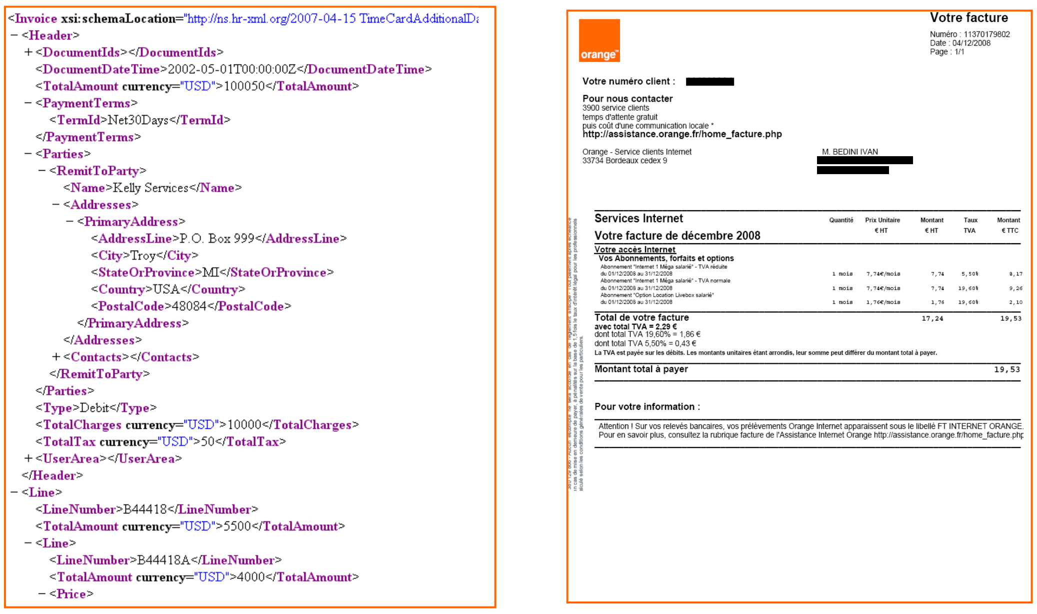
4.1 XML Documents and XML Schemas

An XML Schema [20] formally describes what a given XML document [174] contains, in the same way a database schema describes the data that can be contained in a database (tree structures, data types, integrity constraints, etc.). It describes the coarse shape of the XML document. It can be used to express a set of rules to which an XML document must conform to be considered as 'valid' according to that schema. Rules can define what fields or sub-elemnt an element can contain. It can also describe the values that can be placed into any element or attribute. At present, there exist several XML languages to describe XML documents. Among them, Document Type Definition (DTD) makes part of the XML basic standard. It was the first formalized standard to describe XML data structures, but it is rarely used anymore. The eXternal Data Representation (XDR), an IETF standard from 1995, was an early attempt to provide a more comprehensive standard than DTD. This standard has pretty much been abandoned now in favour of XML Schema Document (XSD) that is currently the most used standard for describing XML documents. Currently two versions are proposed, 1.0 and 1.1, with very few remarkable differences.

4.1.1 Benefits of using XML Documents and XML Schemas

The B2B scenario highlights benefits of choosing XML with XSD as messages formalization with respect to other formats, like the EDIFACT formalization seen in Section 0. Figure 4.1 shows an example of two representations of an invoice business document. The first one is shown as a plain format (i.e., PDF, HTML, or simple text), which has the advantage to be directly human readable. The second one is an XML description that, even if less human friendly, is more scalable and valuable. Arguably the greatest benefit of using XSD is that it provides a formal description useful at every step of the development of an end-to-end solution.

We can add that in a typical program, a great part of the generated code is spent checking the data (someone argues up to 60%). If data is structured as XML, and there is a schema, then you can hand the data-checking task off to a schema validator. Indeed an XML document is **well-formed** if it conforms the XML syntax rules. When compliant to an XML Schema, a document is also **valid**. Figure 4.2 depicts this way of validating an XML document instance with a given XML Schema.



```
<Invoice xsi:schemaLocation="http://ns.hr-xml.org/2007-04-15 TimeCardAdditionalD
- <Header>
  + <DocumentIds></DocumentIds>
    <DocumentDateTime>2002-05-01T00:00:00Z</DocumentDateTime>
    <TotalAmount currency="USD">100050</TotalAmount>
  - <PaymentTerms>
      <TermId>Net30Days</TermId>
    </PaymentTerms>
  - <Parties>
    - <RemitToParty>
        <Name>Kelly Services</Name>
      - <Addresses>
        - <PrimaryAddress>
            <AddressLine>P.O. Box 999</AddressLine>
            <City>Troy</City>
            <StateOrProvince>MI</StateOrProvince>
            <Country>USA</Country>
            <PostalCode>48084</PostalCode>
          </PrimaryAddress>
        </Addresses>
      + <Contacts></Contacts>
      </RemitToParty>
    </Parties>
    <Type>Debit</Type>
    <TotalCharges currency="USD">10000</TotalCharges>
    <TotalTax currency="USD">50</TotalTax>
  + <UserArea></UserArea>
  </Header>
- <Line>
    <LineNumber>B44418</LineNumber>
    <TotalAmount currency="USD">5500</TotalAmount>
  - <Line>
      <LineNumber>B44418A</LineNumber>
      <TotalAmount currency="USD">4000</TotalAmount>
    - <Price>
```

Votre facture
Numéro : 11370179802
Date : 04/12/2008
Page : 1/1

Votre numéro client : [redacted]

Pour nous contacter
3900 service clients
temps d'attente gratuit
puis coût d'une communication locale *
http://assistance.orange.fr/home_facture.php

Orange - Service clients Internet
33734 Bordeaux cedex 9

M. BEDINI IVAN
[redacted]

Services Internet / **Votre facture de décembre 2008**	Quantité	Prix Unitaire € HT	Montant € HT	Taux TVA	Montant € TTC
<u>**Votre accès Internet**</u>					
Vos Abonnements, forfaits et options					
Abonnement "Internet 1 Méga salarié" - TVA réduite du 01/12/2008 au 31/12/2008	1 mois	7,74€/mois	7,74	5,50%	8,17
Abonnement "Internet 1 Méga salarié" - TVA normale du 01/12/2008 au 31/12/2008	1 mois	7,74€/mois	7,74	19,60%	9,26
Abonnement "Option Location Livebox salarié" du 01/12/2008 au 31/12/2008	1 mois	1,76€/mois	1,76	19,60%	2,10
Total de votre facture			17,24		19,53

avec total TVA = 2,29 €
dont total TVA 19,60% = 1,86 €
dont total TVA 5,50% = 0,43 €
La TVA est payée sur les débits. Les montants unitaires étant arrondis, leur somme peut différer du montant total à payer.

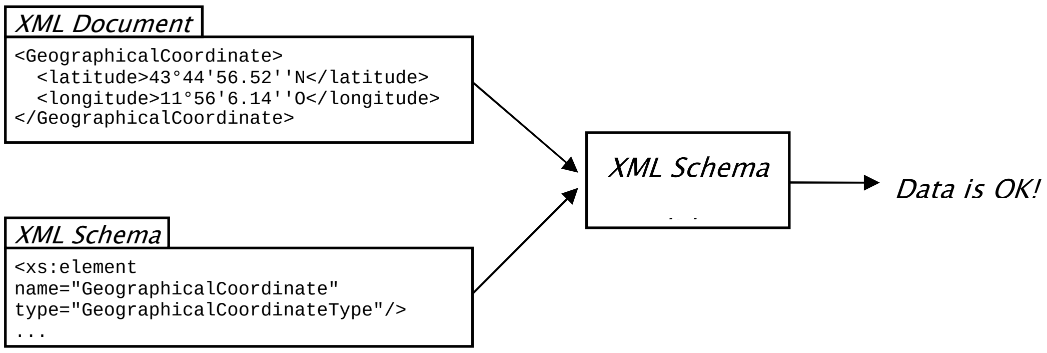
Montant total à payer 19,53

Pour votre information :

Attention ! Sur vos relevés bancaires, vos prélèvements Orange Internet apparaissent sous le libellé FT INTERNET ORANGE.
Pour en savoir plus, consultez la rubrique facture de l'Assistance Internet Orange http://assistance.orange.fr/home_facture.php

Figure 4.1 – Example of invoice as simple document and XML instance

The primary reason for defining an XML schema is to formally describe an XML document; however it can be also useful to others tasks that go beyond simple validation. Indeed the schema can be used to generate human readable documentation; this is especially useful where the authors have made use of annotation elements and to generate code (this is referred to as XML Data Binding). From the automatic ontology generation standpoint, the XML Schema data model already includes: the vocabulary (element and attribute names), the content model (relationships and structure) and the data types. This feature makes it profitable for the information extraction step and simplifies the concept structure recognition.



```
<GeographicalCoordinate>
  <latitude>43°44'56.52''N</latitude>
  <longitude>11°56'6.14''O</longitude>
</GeographicalCoordinate>
```

```
<xs:element
name="GeographicalCoordinate"
type="GeographicalCoordinateType"/>
...
```

Figure 4.2 – Validating XML data

4.1.2 XML Schema Components

Technically, a XML schema is an abstract collection of metadata. This collection is usually created by processing a collection of XML schema documents, which contains the source language definitions (also known as XML Schemata) of the metadata components. In popular usage, however, a schema document is often referred to as a schema. Thus for the sake of simplicity, throughout this document XSD, the schema definition language format, will be often the name used to refer a schema itself.
The W3C XML Schema recommendation [20] defines an XML Schema as a set of building blocks, also referred to as schema components, that comprises the abstract data model of the schema. As depicted in Figure 4.3, there are 13 different components, falling into three groups:

Primary components which may (in case of type definitions) or must (in case of element and attribute declarations) have names: simple type definitions, complex type definitions, attribute declarations and element declarations.

Secondary components, which must have names: attribute group definitions, identity-constraint definitions, model group definitions and notation declarations.

Helper components, which provide small parts of other components and are dependent on their context: annotations, model groups, particles, wildcards and attribute uses.

XML Schema proposes several ways to declare and compose components in a schema declaration. For example we can find at least 17 ways to declare elements (e.g. global/local element, ref's to a global element, a global/local element which defines a simple/complex type inline declarations) and 20 different ways to declare attributes. This makes a real challenge to provide a quick view of XML Schema design and how components can be composed among them. So what follow is just a brief introduction to some XSD components and their combination, at least for those that are used in our system to extract information for the ontology generation. A more detailed explanation of XML Schema can be found in [175] [155].

4.1.2.1 Elements

XSD *element* is the most used component (see Section 4.2.1 for more details) . With the *attribute*, it defines the tag syntax for XML documents. More than attributes, the element component allows the description of simple and complex entities to define different kind of concepts for the ontology to build, like classes or properties. Elements can be declared in several different methods. Listing 4.1 shows three examples of possible declarations for an element. The first one is a local element with a declared basic built-in XSD datatype. It also defines the expected occurrence number that in this case is 0, i.e., the element is optional (because the *minOccurs* is set to 0), while *maxOccurs set to 1* means that at most, it can appear 1 time. In the second example, the element is declared with an inline *simpleType* that refines an XSD built-in data type. The latter is declared with inline *complexType* and inline sub-elements.

Global elements and global types are element declarations/definitions that are immediate children of the root <*schema*> element. Local elements, local types, and inline types are declarations/definitions that are nested within other elements or types. Although inline and local declarations, like those presented in the listing above, result in a much more compact schema, they have the disadvantage of being not reusable by other elements. Listing 4.2 shows a formally equivalent global declaration for the previous `GeographicalCoordinate` element referencing a named complex type.

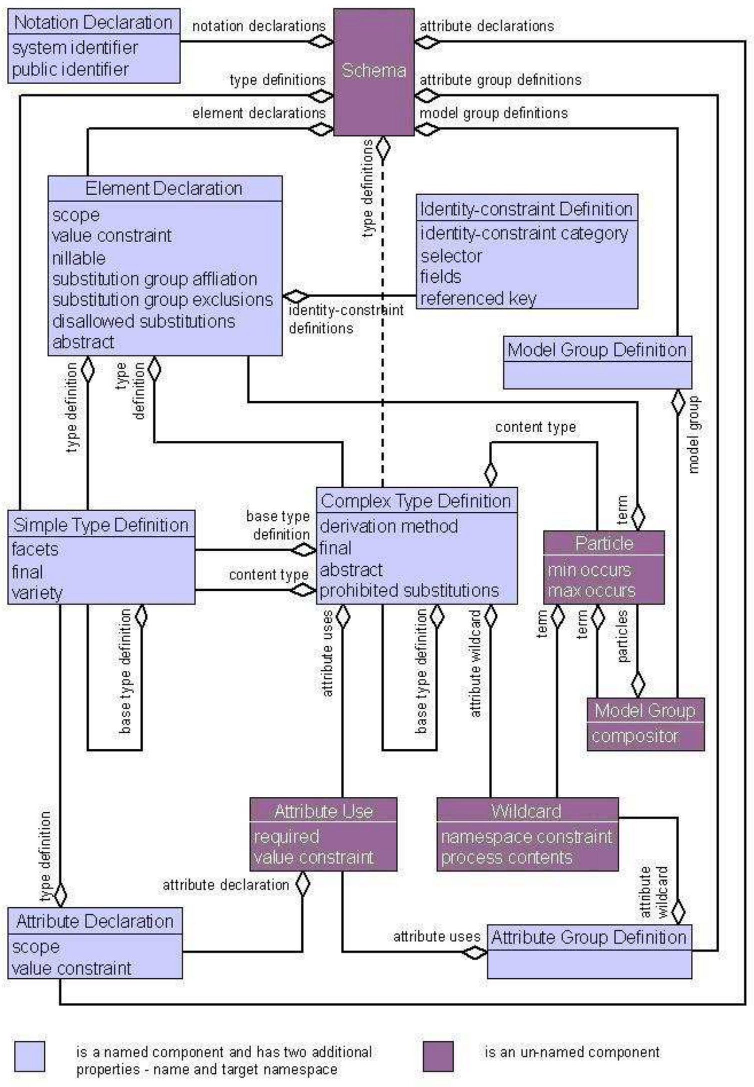


Figure 4.3 – XML Schema component data model

XML Schema specifications do not outline preferences to follow, but as general rule the global declaration should be preferred to local and inline declarations. As illustrated in Listing 4.3, a global element can be reused by other component definition simply using the element *ref* declaration. This makes definition of elements and their usage clearly separated, which is generally simpler to understand and reuse.

```
<xsd:element name="Amount" type="xsd:integer" minOccours="0" maxOccours="1"/>

<xsd:element name="Amount">                                        Local declaration
 <xsd:simpleType>
  <xsd:restriction base="udt:amountType">
   ...
  </xsd:restriction>
 </xsd:simpleType>                                     Anonymous types (no name)
</xsd:element>

<xsd:element name="GeographicalCoordinate">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="longitude" type="xsd:string" minOccurs="1"/>
      <xsd:element name="latitude" type="xsd:string" minOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>                                          Inline declarations
</xsd:element>
```

Listing 4.1 – Elements declarations

```
<xsd:element name="GeographicalCoordinate" type="GeographicalCoordinateType"/>
<xsd:complexType name="GeographicalCoordinateType">
 <xsd:sequence>                                  Named type
  <xsd:element name="longitude" type="xs:string"/>
  <xsd:element name="latitude" type="xs:string"/>
 </xsd:sequence>
</xsd:complexType>
```

Listing 4.2 – Examples of Geographical Coordinate element declaration

```
<xsd:complexType name="Someone">
  <xsd:sequence>
    <xsd:choice>
      <xsd:element ref="Person" minOccurs="0"/>
      <xsd:element ref="Contact" minOccurs="0"/>
      <xsd:element ref="Employee" minOccurs="0"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="Person" type="Person"/>
<xsd:element name="Employee" type="Employee"/>
<xsd:element name="Contact" type="Contact"/>
```

Listing 4.3 – Example of element ref usage (from HR-XML)

4.1.2.2 Attributes

XML Schema *attribute* component is used to declare simple values for a given complex element (attributes cannot have child elements). Attribute declarations can appear at the top level of a schema document, or within complex type definitions, either as complete (local) declarations, or by reference to top-level declarations, and also within attribute group definitions. For complete declarations, top-level or local, the type attribute is used when the declaration can use a built-in or pre-declared simple type definition. Otherwise an anonymous simple type is provided inline.

Listing 4.4 shows an example of inline attributes declaration for a complex type component. We can observe that at data content level, this definition of `GeographicalCoordinateType` and that one provided in Listing 4.2 are equivalent. Here again XML Schema specifications do not provide any recommendation about the usage of one declaration rather than another. Generally attributes are indicated for transmitting metadata information, like an internal identifier or a specific detail on the value. For the geographical coordinate it could be the specific coordinate system (e.g. cartesian or polar). Sub-elements may be more appropriate for carrying out the real data.

```
<xsd:complexType name="GeographicalCoordinateType">
  <xsd:attribute name="longitude" type="xsd:string"/>
  <xsd:attribute name="latitude" type="xsd:string"/>
</xsd:complexType>
```

Listing 4.4 – Example of usage of attributes

4.1.2.3 Simple and Complex Types

As mentioned above, elements and attributes are declared in a schema. They have a representation in an XML instance document, while complex and simple type components are defined and used only within the schema document(s) and thus have no representation in an XML instance.

The XSD *complexType* component is normally used to define components with child elements and/or attributes. The *simpleType command* is used to create a new datatype that is a refinement of a built-in XSD type (e.g., string, date, gYear, etc). In particular, we can derive a new simple type by restricting an existing simple type; in other words, the legal range of values for the new type are a subset of the existing type range of values. We use the *simpleType* element to define and name the new simple type.

Type components can be anonymous (without name) when used locally for an element, but they must be named for a global definition. Listings above already provide examples of declaration for complex types. Listing 4.5(1) provides the definition of a global simple type `CountryCodeType` as a restriction of the built-in string datatype. For instance it has a specific pattern that allows string instances with only two characters defined by the regular expression "`[A-Z][A-Z]`". In addition to the so-called *atomic* types XML Schema simple types have also the concept of *list* and *union* types. Atomic types and list types enable an element or an attribute value to be one or more instances of one atomic type. In contrast, a union type enables an element or an attribute value to be one or more instances of one type drawn from the union of multiple atomic and list types. Listing 4.5(3) illustrates an example of a simple type with *union* definition, where the `DispositionType` union type is built from one atomic type, `xsd:string` in this case, and one simple type, `CriminalDispositionTypes` which is a closed list of allowed string values, called *enumeration*, shown in Listing 4.5(2).

4.1.2.4 Derived Types

XSD provides two forms of sub-classing type components, also called **derived types**. A first way derives by extension a parent complex type with more elements, while a second derivation can be obtained by restriction of the base type, creating a type as a subset. The restriction for simple types operates with the application of constraints on predefined simple types or with the help of regular expressions, as already seen above. Restriction of complex types is conceptually the same as restriction of simple types, except that the restriction of complex types involves a type's declarations rather than the acceptable range of a simple type values. A complex type derived by restriction is very similar to its base type, except that its instances are more limited than the corresponding declarations in the base type.

```
<xsd:simpleType name="CountryCodeType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[A-Z][A-Z]"/>
  </xsd:restriction>                                      1
</xsd:simpleType>
<xsd:element name="CountryCode" type="CountryCodeType"/>

<xsd:simpleType name="CriminalDispositionTypes">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Acquitted"/>                              2
    <xsd:enumeration value="AdjournedToX"/>
   ...
    <xsd:enumeration value="Waiver"/>
  </xsd:restriction>
<:xsd:simpleType>

<xsd:simpleType name="DispositionType">
 <xsd:union memberTypes="CriminalDispositionTypes xsd:string"/>     3
</xsd:simpleType>
```

Listing 4.5 – Example of simple type component definitions

XML Schema provides two components to derive types. The *complexContent* component signals that we intend to restrict or extend the content of a complex type. A *simpleContent* component indicates that the content of the new complex type contains only simple data and no element. In other words, *simpleContent* provides a solution for adding attributes to simple types.

Listing 4.6 illustrates two extensions for a complex type and precisely in (1) with the simple content component we provides more attributes to `DescriptionType`, which is defined as a string (not shown in the example).While in (2) with complex content component we extend `PostalAddressType` base complex type with more sub-elements and attributes at the same time.

4.1.2.5 Grouping XML entities

XML Schema enables groups of elements to be defined and named, so that the elements can be used to build up the content models of complex types. Thus to provide more information about an element, XML Schema permits to create a named *group* global component that permits to assembly together more elements that can be simply referenced in complex elements. The same is done with the *attributeGroup* containing all the desired attributes of an item element that can be referenced by name in more elements declarations.Moreover the definitions of complex types are declared using sequences of elements that can appear in the document instance. XML Schema provides three different constructors to allow the definition of sub-elements sequences:

- *sequence* corresponds to an order collection of typed sub-elements;
- *choice* groups element using an exclusive-or, i.e., only one of its children can appear in an instance;
- *all* contains at most one of each element specified as sub-elements. It means that all the elements in the group may appear once or not at all (i.e. the permissible values of minOccurs and maxOccurs are 0 and 1) and they may appear in any order.

Listing 4.7 illustrates the definition of `TelecomNumberType` complex type, where sub-elements can be either `FormattedNumber` or the ordered sequence of elements grouped by `TelecomNumberGroup`.

```
<xsd:complexType name="TelcomNumberType">
  <xsd:choice>
    <xsd:element ref="FormattedNumber"/>
    <xsd:group ref="TelcomNumberGroup"/>
  </xsd:choice>
</xsd:complexType>
<xsd:group name="TelcomNumberGroup">
  <xsd:sequence>
    <xsd:element ref="InternationalCountryCode" minOccurs="0"/>
    <xsd:element ref="NationalNumber" minOccurs="0"/>
    <xsd:element ref="AreaCityCode" minOccurs="0"/>
    <xsd:element ref="SubscriberNumber"/>
    <xsd:element ref="Extension" minOccurs="0"/>
  </xsd:sequence>
</xsd:group>
```

Listing 4.6 – Examples of extension with simple and complex content (excerpts from GS1 (1) and from HR-XML (2))

```
<xsd:complexType name="NoteType">
  <xsd:simpleContent>                                    1
    <xsd:extension base="DescriptionType">
      <xsd:attribute name="author" type="StringType" use="optional"/>
      <xsd:attribute name="entryDateTime" type="DateTimeType" use="optional"/>
      <xsd:attribute name="status" type="StringType" use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="CreditPostalAddressType">        2
  <xsd:complexContent>
    <xsd:extension base="PostalAddressType">
      <xsd:sequence>
        <xsd:element name="ReportedDate" type="ReportedDateType" minOccurs="0"/>
        <xsd:element name="LastReportedBy" type="xsd:string" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="current" type="xsd:boolean" use="optional"/>
      <xsd:attribute name="enteredOnInquiry" type="xsd:boolean" use="optional"/>
      <xsd:attribute name="timesReported" type="xsd:string" use="optional"/>
      <xsd:attribute name="validFrom" type="AnyDateTimeNaType" use="optional"/>
      <xsd:attribute name="validTo" type="AnyDateTimeNaType" use="optional"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Listing 4.7 – Example of components to group entities (from OAGIS 9.0)

4.1.2.6 Annotations

XML Schema provides three elements for annotating schemas for the benefit of both human readers and applications. One is a basic schema description information, the *documentation* component, which is the recommended location for human readable material. The second is *appinfo* component that can be used to provide information for tools, style-sheets and other applications. Both *documentation* and *appinfo* appear as sub-elements of *annotation*, which may itself appear at the beginning of most schema constructions. To illustrate, Listing 4.8 shows a documentation annotation element appearing at the beginning of a complex type definition.

```
<xsd:complexType name="AmountType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      <ccts:UniqueID>UDT000001</ccts:UniqueID>
      <ccts:CategoryCode>UDT</ccts:CategoryCode>
      <ccts:DictionaryEntryName>Amount. Type</ccts:DictionaryEntryName>
      <ccts:VersionID>1.0</ccts:VersionID>
      <ccts:Definition>A number of monetary units specified in a currency where
the unit of the currency is explicit or implied.</ccts:Definition>
      <ccts:RepresentationTermName>Amount</ccts:RepresentationTermName>
      <ccts:PrimitiveType>decimal</ccts:PrimitiveType>
      <xsd:BuiltinType>decimal</xsd:BuiltinType>
    </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleContent>
    ...
  </xsd:simpleContent>
</xsd:complexType>
```

Listing 4.8 – Example of UBL annotations following CCTS format for annotations

4.2 B2B Specifications

After a brief introduction to XML Schema in this section we present the analysis of the source corpus we collected for the B2B domain. As already told in Chapter 2, in our research of B2B specifications we found the most part of them formalized using XML Schemas. Thus before starting the extraction of conceptual knowledge from them, we provide elements to quantify the information we collected and secondly an analysis of some design practices to profile the conceptual knowledge extraction from this kind of source. The result is a tailoring for the extraction operation to XML sources for the B2B domain. However even though it has not been proved yet we estimate that our choices can be applied to a more generic set of XML Schema sources.

4.2.1 Some Figures of B2B XML Schemas

With a corpus of 25 B2B standard specifications we collected a base of 3432 XSD files containing more than 586.000 XML Schema components (that hereafter we will also call 'tags') and among these tags at least 170.000 are named. For information Figure 4.4 illustrates the repartition of extracted information, measured as total number of XML components, among the considered B2B standard bodies.

From the *camembert* graph we observe that Mismo is the more prolific standard body, few others provide between 5 and 10 % each and around 30 % is shared between the remaining standards. Of course, this picture does not say if the extracted information provides relevant knowledge, for this we need further investigation.

Figure 4.5 provides a global view of the usage of XML Schema components we have considered. It clearly shows that standard bodies include a considerable amount of documentation. Moreover XSD *element* and XSD *attribute* are the most used components, while others like *union*, *all*, *any* and *substitutionGroup* are very few adopted. Here again, the figure only provides a statistical measure of the component adoption and simply gives us a list of those components that should be included in the extraction of information from XML Schemas.

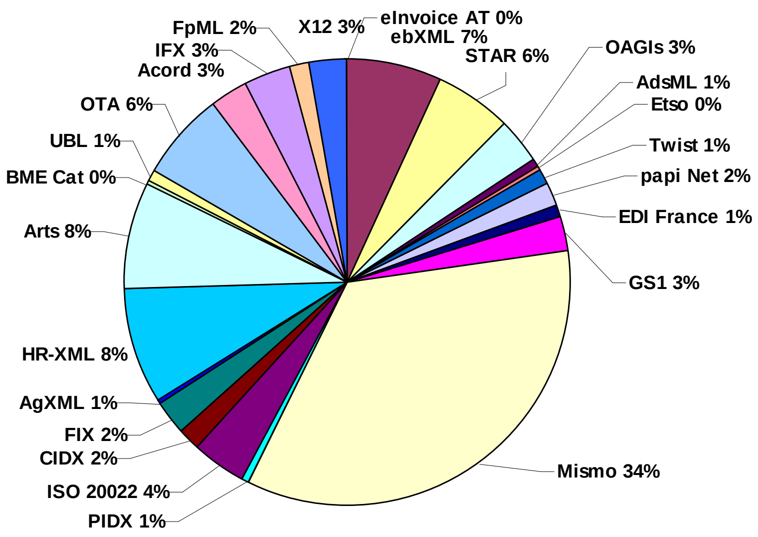


Figure 4.4 – B2B standard bodies' specifications extraction

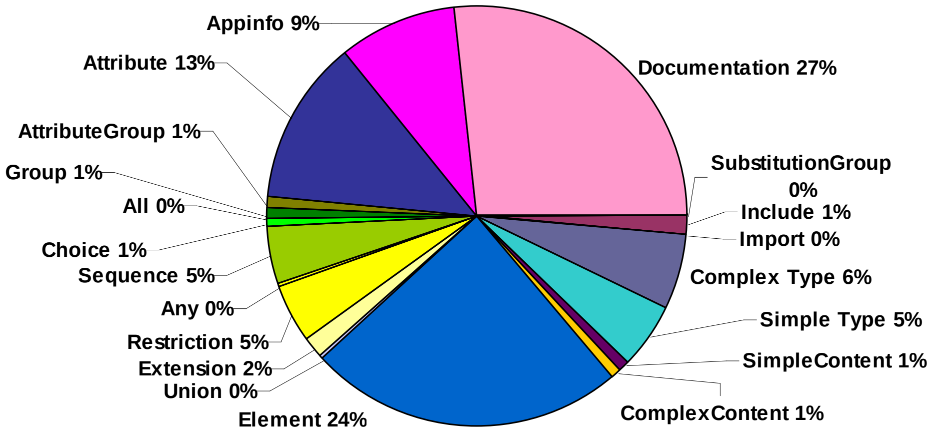


Figure 4.5 – XML Schema components extraction

Another time, we stress out the fact that our work target as much automation as possible, this is the reason why we try to look for the most generic and most relevant way to extract knowledge from this kind of documents.

4.2.2 Different Kinds of XSD Components Usage

Which standard is better reusable? Complex types are only used to define complex objects? Do we have to consider all XML Schema components to get satisfactory information retrieval results? What component is better representative for conceptual knowledge extraction? And of course, are XML Schemas a good source corpus for concept retrieval?

These are only a few of those questions that come with an undefined number of schemas. If this task can be easily done with a narrow number of schemas by a human, it becomes a real challenge when automating it. In the sub-sections below, we analyse some XML Schema constructs and their usage among B2B specifications in order to obtain some useful information to improve automatic retrieval of conceptual information from XML Schemas.

4.2.2.1 Which standard is more reusable?

Reuse hides different things, in one hand it permits to take benefit from external works, on the other hand reuse provides a good way to facilitate integration of data applications. Furthermore if we are able to define reusable components, it implicitly means that probably we get our hands on useful concepts.

As already mentioned in Section 4.1.2.1, XML Schemas reusability is provided by global components and mostly by global types rather than by elements. Indeed, unlike elements, types allow roles definition similarly to UML. This is a simple feature with large endow to the need for semantic tailoring when reusing components and thus looking for common high level concepts.

Listing 4.9 illustrates an example of such case: in (1) a global type describes a very simple structure for the `Address` "concept", and sub-elements of `PersonType` are used to redefine it; while in (2) is the element `Domiciliation` to be global (with an inline anonymous type) and a sub-element of `Person` uses the attribute `ref` to make reference to `Domiciliation`.

Even thought the two declarations are formally equivalents, in the first case we are able to redefine `Address` as `Residence` or `OfficeLocation` (that deserves the specific context needed by the concept of `Person`), while in the latter we can only refer the element, without the possibility to redefine the name of the role of the association among `Person` and `Domiciliation`. Therefore only global types offer the basic flexibility required to redefine concepts expressed in a schema, thus provide a better reuse. This is an important feature that enhances the inclusion of general schemas into others (with *include* and *import* XML Schema components), to reuse concepts and relationships.

Looking inside standard specifications, we can observe that a great part of them respects this practice of reusability. Figure 4.6 illustrates the usage of global and local complex type descriptions. From this test, it comes out that at least 3 out of 4 are global declarations. Normally a clear choice is done by each standard body about what kind of type declaration they apply.

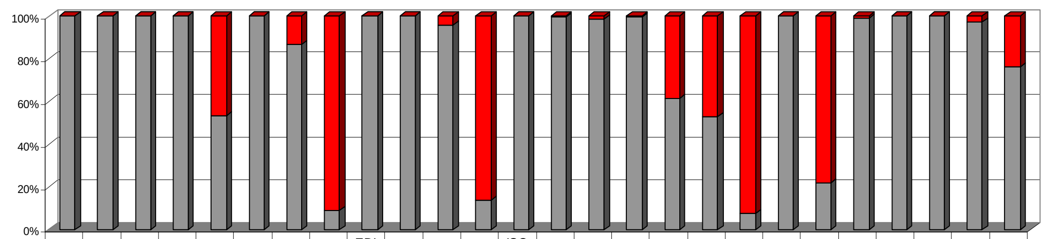
Figure 4.7 illustrates the same kind of statistics for element components and highlights three kinds of definitions: global elements with anonymous complex type declaration, elements linking a complex/simple type, and local elements referencing a global one. Again we observe that element with a declared type is still the most adopted design practice; in despite of its syntax verbosity, it still remains the preferred way to declare components.

```
<xsd:complexType name="AddressType">
  <xsd:sequence>
    <xsd:element name="Street" type="xsd:string"/>
    <xsd:element name="Country" type="xsd:string"/>
    <xsd:element name="PostalCode" type="xsd:string"/>
  </xsd:sequence >
</xsd:complexType>

<xsd:complexType name="Person">
  <xsd:sequence>
    <xsd:element name="Name" type="xsd:string"/>
   <xsd:element name="Residence" type="AddressType"/>
   <xsd:element name="OfficeLocation" type="AddressType">
 </xsd:sequence>
</xsd:complexType>

<xsd:element name="Domiciliation">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Street" type="xsd:string"/>
      <xsd:element name="Country" type="xsd:string"/>
      <xsd:element name="PostalCode" type="xsd:string"/>
    </xsd:sequence >
 </xsd:complexType>
</xsd:element>

<xsd:element name="Person">
  <xsd:sequence>
    <xsd:element name="Name" type="xsd:string"/>
    <xsd:element ref="Domiciliation"/>
  </xsd:sequence>
</xsd:element>
```

1

2

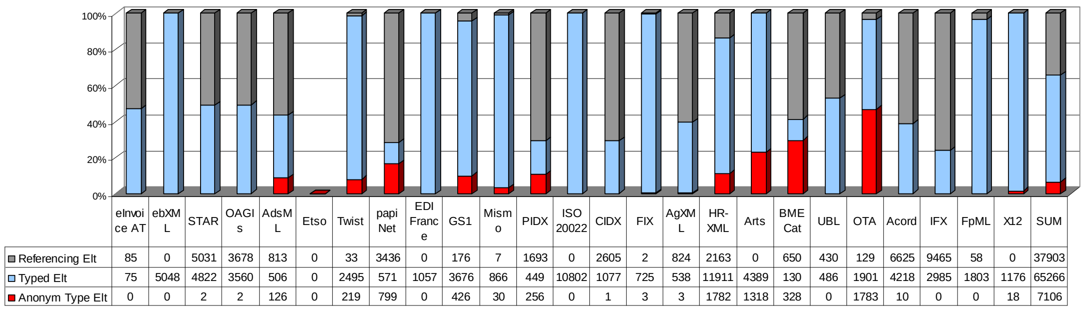
Listing 4.9 –Different element declarations with 'type' and 'ref'



	eInvoice AT	ebXML	STAR	OAGIs	AdsML	Etso	Twist	papiNet	EDI France	GS1	Mismo	PIDX	ISO 20022	CIDX	FIX	AgXML	HR-XML	Arts	BME Cat	UBL	OTA	Acord	IFX	FpML	X12	SUM
🟥 Local CT	0	0	0	0	126	0	85	794	0	0	30	256	0	1	3	1	1642	1155	243	0	1779	6	0	0	18	6139
▨ Global CT	30	780	1304	1665	144	34	560	78	118	1701	713	40	2838	803	219	268	2614	1291	19	209	496	1099	1331	674	622	19650

Figure 4.6 – Declarations of Global and Local Complex Types components percentage

We could deeply argue on the XML Schema "typing" feature and its direct relation with reusability, but seeing that the aim of standards is mainly to produce largely adopted harmonized components, we can consider enough the observation of B2B design practices to confirm the relation with reusability. However we observed that this condition alone is not enough to decide the goodness of an XML Schema, and other factors must be considered, like semantics and structures features.



	eInvoice AT	ebXML	STAR	OAGIs	AdsML	Etso	Twist	papi Net	EDI France	GS1	Mismo	PIDX	ISO 20022
▨ Referencing Elt	85	0	5031	3678	813	0	33	3436	0	176	7	1693	0
🟦 Typed Elt	75	5048	4822	3560	506	0	2495	571	1057	3676	866	449	10802
🟥 Anonym Type Elt	0	0	2	2	126	0	219	799	0	426	30	256	0

	CIDX	FIX	AgXML	HR-XML	Arts	BME Cat	UBL	OTA	Acord	IFX	FpML	X12	SUM
▨ Referencing Elt	2605	2	824	2163	0	650	430	129	6625	9465	58	0	37903
🟦 Typed Elt	1077	725	538	11911	4389	130	486	1901	4218	2985	1803	1176	65266
🟥 Anonym Type Elt	1	3	3	1782	1318	328	0	1783	10	0	0	18	7106

Figure 4.7 – Different descriptions of Elements

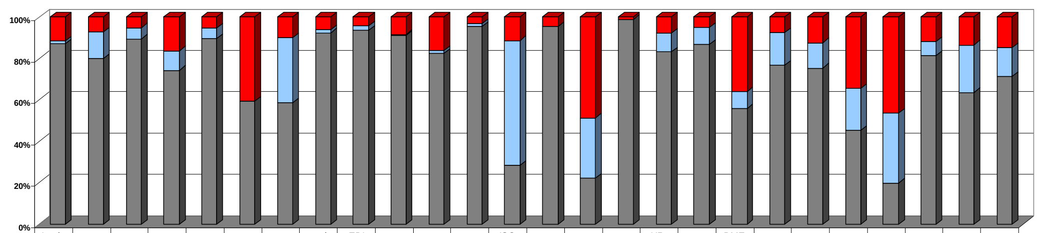
4.2.2.2 XML Schema & B2B Semantics

In this sub-section, we focus on adequacy of extracted tag labels to provide well formed names to concepts (i.e. no abbreviations, acronyms and in general non dictionary words) for the ontology to build. This is another important feature to analyse. Indeed if it is intuitively simple to believe that text documents contain words belonging to a dictionary, it is not always the case for XML Schemas. As we observed, several XSD specifications have different practices on naming conventions that not always are of direct understanding. Thus their automatic interpretation is not always a trivial task.

For example, XML tags are often compound words that can be expressed using the common *Upper Camel Case* convention with known terms (that we also call **dictionary terms**), like *OfficeLocation*, or using abbreviations to reduce XML tags size like *amt_ccy* (which should stand for amount currency). In addition, tags can contain compound words (like *cash-flow*), acronyms, bad spelled words, no separator between terms (like *foodservice*), specific terms, unrelated words with the meaning of the element (like *UnitOfMeasureBBIECommonData*), etc….

The string label matching is the basis of the machine correspondences detection algorithm. An ontology must have clear semantics for concept names. A particular way to use a not precise sequence of chars transforms the automatic definition of concepts into a complex task. For this reason, before starting the generation of ontological knowledge from this kind of information, we try to determine if XML Schema component names are semantically well formed. If not, we look for a simple way to transform such tags to dictionary words.

Of course the underlying question is that semantics provided by tag labels is not enough to generate correct ontologies. For this, we have developed a service (detailed in the next section) that extracts tag labels from named complex/simple types, elements, attributes, attribute groups and groups, i.e., from the greatest part of named XSD components. With these labels, we try to obtain known words and we mark as "unknown" those tags that contain at least one unrecognized word (that we also refer as **bad word**). Results of this simple test are presented in Figure 4.8, and summarized in the pie chart shown in Figure 4.9.



	eInvoice AT	ebXML	STAR	OAGIs	AdsML	Etso	Twist	papi Net	EDI France	GS1	Mismo	PIDX
🟥 Unknown	15	449	351	942	56	37	365	149	52	588	7325	31
🟦 Abbreviations	2	826	368	542	54	0	1144	41	26	39	543	13
⬛ Recognized	115	5047	6100	4301	955	54	2151	2305	1169	6444	37299	920

	ISO 20022	CIDX	FIX	AgXML	HR-XML	Arts	BME Cat	UBL	OTA	Acord	IFX 170	FpML	X12	SUM
🟥 Unknown	1848	106	3073	14	1751	549	211	69	983	2501	2550	331	695	25041
🟦 Abbreviations	9923	2	1833	0	1999	895	49	149	979	1455	1852	191	1195	24120
⬛ Recognized	4667	2274	1404	1271	18504	9456	327	722	6008	3284	1090	2278	3285	121430

Figure 4.8 – Results of the extraction of XML tags semantics

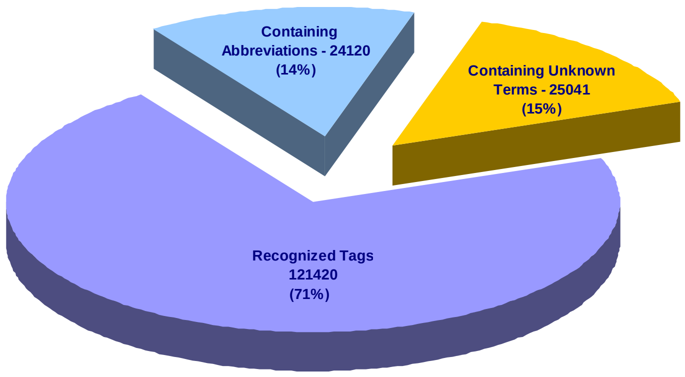


Figure 4.9 – XML tags semantics identification

We can see from this test that the results are encouraging. In fact 71% of tags are composed by recognized dictionary words, 14% contain recognized abbreviations that can be related to dictionary words, and only 15% of total tags contain unknown words. However, even though this is a good partial result, a satisfactory extraction system could be aware of these "bad" tags. In specific cases, the introduction of such a noise can lead to bad extraction/matching conclusions. To get optimal results, a system should execute this kind of test for each source to be included. On the basis of a predefined threshold, it should decide to use specific terms recognition algorithms, or in the worst case exclude a source from the corpus to be considered to generate the ontology. This is what we do in our system. More details on this feature are provided in Section 4.5.

4.3 Generating Automatically a B2B Taxonomy

Considerations seen above highlight some XML schema definition practices, such as the use of anonymous types for elements, rather than declared types; the adoption of Upper Camel Case, underscore or hyphen to separate compound words for tags; the trend that financial and related bodies (like IFX, FIX and ISO 20022) often use abbreviations rather than real terms. As we have seen, among all extracted tags the great part of them are composed by dictionary words. For this reason, we conducted a simple test aiming at studying the frequency and the attendance[35] of single terms, rather than tags, to determine if they can be used to define a core taxonomy to use as basis for a common ontology generation. In the next subsection, we explain the process our system implements to extract terms from XSD tag labels, while in a second subsection we present results and conclusions on this issue.

4.3.1 Extraction Process

The aim we give to the extraction process is to retrieve as much information as possible from the source corpus, to transform it into a normalized form and finally to organize information in a simpler machine understandable format to be used to generate the ontology. Figure 4.10 depicts the process we implement for the **extraction of terms**.

In our use case, we consider each B2B standard as providing a natural cluster of input sources. Later, for each of them we verify that the source has not been already included in the corpus. If it is not the case, we proceed with the extraction process. It is composed of the following steps: acquisition, normalization, filtering, and sources formalization. These steps are detailed below.

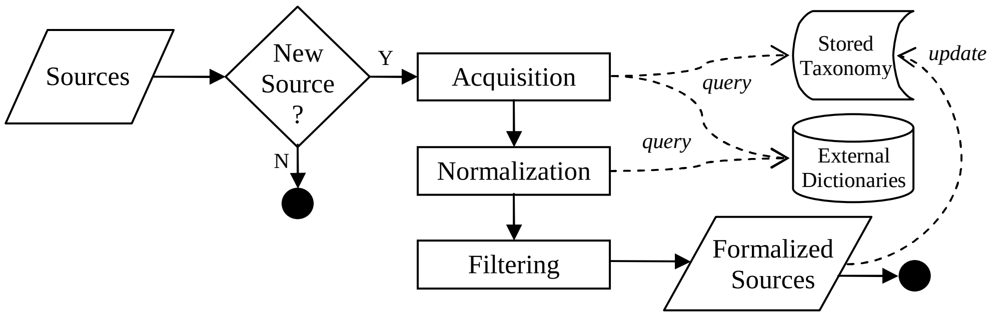


Figure 4.10 – Terms Extraction Process

Acquisition Step

The aim of this step is to organize the source corpus and to select useful terms for the base taxonomy. The sub-tasks are:

1. Parse XSD and extract XML tag values for named components.
2. Check for already normalized tags in the stored Taxonomy.
3. Check for composite words (e.g.: on-line).
4. Determine previously identified "useless" words, like systematic addition of unrelated semantic sense to the tag (e.g.: *CommonData* for *UnitOfMeasureCommonData*)
5. Split compound terms forming the tag, using the UCC convention, or '_' or '-' as separators, taking careful of special cases (e.g.: *PersonIDCode* = person + id + code)
6. Check for known abbreviations (e.g.: Addr = Address, PO = [Purchase Order, Post Office])

[35] In this case we define attendance as the number of standards using a given word.

7. Check for stop-word[36] (removes words like “of”, “a”, “for”,…);

For tasks 3, 4 and 6, we integrate specific external dictionaries to detect stop words and abbreviations. We also maintain a built-in list of words that can produce noise to the concept naming affectation. Finally, as output of this step, we produce a list of detected stop words, abbreviations, and a set of tags for each source in the form: $Term_1$_$Term_2$_..._$Term_X$ (ex.: *ABIEPostalAddressType* that becomes *ABIE_Postal_Address*)

Normalisation Step

At this step, the machine is not able to say if a term composing a tag is a real term or something else (acronym or unidentified abbreviation for example). Thus, to improve semantic tags recognition, we add the use of an electronic dictionary as external resource. It determines if a term is a real human word or not. In our case, we have integrated WordNet version 3.0. Tasks for this step are:

8. *Case normalisation*, all terms are converted to lower case;
9. *Bad words detection*, terms unknown by the dictionary are cast aside;
10. *Morphological and semantic normalisation*, which consists in finding the stem and lemma form for all terms composing extracted tags.

The output of this step is a list of normalized terms for those words that are present in the dictionary and a list of bad words for the others not detected in any list previously defined. Moreover, we use the linguistics canonical form of a word (i.e., the lemma) as final normalized form; it gives the most representative name for a concept.

Filtering Step

In this step, we analyse the words that have been rejected, in a first pass called *bad words reconciliation.* This is done by applying a modified version of the N-Gram algorithm and Levenstain distance to bad words. We detect as many abbreviations as possible that still are not present in the built-in abbreviation dictionary. We restrict ourselves to terms within the recognised terms list, because if we use the complete dictionary, we would detect too many similar terms, most of them out of context.

At this time if the ratio between number of unidentified words and those that have been recognized is upper than a fixed threshold, the source can be filtered and thus removed from the corpus used to generate the ontology.

Moreover we also perform *Useless words detection*. Using a lattice of compound words, we detect automatically those words that present disproportionate relationships between graph nodes (like *Type* or *CommonData*). They do not convey any semantics. Finally, we integrate as concepts new detected terms.

[36] A stop word is a word, usually one of a series in a stop list, that is to be ignored because considered as non influential to the semantic meaning of a sentence (like prepositions or conjunctions)

Sources Formalization

The aim of this step is to create a first level of semantic relationships and hierarchy between elements for the taxonomy and to provide a first measure of their relevance. For this we:

1. *Check Synonyms* (also meronyms[37] and holonyms[38] to define some kind of hierarchy among elements), in the words belonging to the taxonomy.
2. *Recompose tags.* All tags are recomposed using their lemma in order to be able to detect more similar terms.
3. *Calculate Terms/tags Frequencies.*
4. *Build Tags Lattice.* Tags are usually composed by more than one word. Thence, we build a graph, based on Galois lattice, to relate those tags having the same words (ex. *address* and *postal_address*); we calculate the frequency of graph nodes, and we remove the nodes that are insignificant (values below a threshold)

With this process applied to all input sources, we produce a list of words and normalized tags that can be used to build a core common taxonomy with respect to the selected corpus. The next section details the results we obtain for the specific B2B domain.

4.3.2 Results on B2B Taxonomy Creation

With the list of dictionary words, we have produced some tests to quantify the extracted semantics. The goal is to evaluate if the built taxonomy is representative for the domain. For this, we first measure the term attendance w.r.t. standards. Secondly, we measure also the global frequency of terms. These two measures are referred to as *Common Terms (CT)* as *Usage terms frequency (UTF)* they quantify the usage of words. For example, let use consider a collection of two standards *S = {A,B}*, with tag labels composed by three words *W = {invoice, order, price}*, where terms are distributed among the two standards as depicted in Table 4.1.

	A	B	*Sum*
Invoice	2	4	*6*
Order	-	3	*3*
Price	1	-	*1*

Table 4.1 – Simple example of attendance and occurrence (cell value)

In this example, the collection of terms with attendance = 2 has a *CT* value equals to 0,33 (= 1/3), because only *invoice* is present in both standards in a set of 3 words, while *UTF* value corresponds to 0,6 (= 6/(6+3+1)). For attendance = 1, *CT* = 0,66 (= 2/3) while UTF = 0,4 (= (3+1)/(6+3+1)). Formally the two measures are defined as follows:

[37] A meronym denotes a constituent part of, or a member of something

[38] A holonymy defines the relationship between a term denoting the whole and a term denoting a part of, or a member of, the whole.

$$CT_j = \frac{count(w)_j}{\sum n};$$

Where :

- $count(w)_j$ is the number of words with attendance value j

- $\sum n$ corresponds to the total number of words

$$UTF_j = \frac{\sum_j occur(w_i)}{\sum occur};$$

Where :

- $\sum_j occur(w_i)$ is the sum of words' occurencies having attendance j

- $\sum occur$ is the occurrence for all attendence values

Attendance	N. Terms	CT [%]	Relative CT [%]	Occurrences	UTF [%]	Relative UTF [%]
25	4	0,119	0,119	46052	8,397	8,397
24	15	0,447	0,567	89514	16,32	24,72
23	15	0,447	1,015	36055	6,574	31,29
22	15	0,447	1,463	29100	5,306	36,60
21	14	0,418	1,881	27711	5,053	41,65
20	14	0,418	2,299	23516	4,288	45,94
19	13	0,388	2,687	10613	1,935	47,88
18	20	0,597	3,284	21589	3,936	51,81
17	21	0,627	3,911	12301	2,243	54,06
16	29	0,865	4,777	15938	2,906	56,96
15	24	0,716	5,494	17015	3,102	60,06
14	58	1,731	7,226	22726	4,144	64,21
13	42	1,254	8,480	17871	3,258	67,47
12	41	1,224	9,704	13554	2,471	69,94
11	52	1,552	11,25	14011	2,555	72,49
10	65	1,940	13,19	13305	2,426	74,92
9	54	1,612	14,81	10905	1,988	76,91
8	65	1,940	16,75	12485	2,276	79,19
7	90	2,687	19,43	15723	2,867	82,05
6	110	3,284	22,72	11625	2,119	84,17
5	178	5,315	28,03	13687	2,495	86,67
4	201	6,001	34,04	13516	2,464	89,13
3	302	9,017	43,05	20956	3,821	92,96
2	513	15,31	58,37	16804	3,064	96,02
1	1394	41,62	100	21799	3,975	100
	Σn=3349			**Σoccur=548371**		

Table 4.2 – Common Terms and Usage terms frequency for the B2B source corpus

Applying these two measures to the B2B source corpus, we obtain the values depicted in Table 4.2, where we added correspondent relative values as simple sum of previous values, to provide a direct measure of the percentage of all words having attendance greater than the referred line.

Figure 4.12 and Figure 4.11 illustrate two different representations of Table 4.2 relative measures data. They clearly show that even if the collection of common normalized words used by more standards is not so high, a small set of words largely cover the number of total instances. Indeed if there are around 40% of words (~1400) that are used by only one standard at once, less than 2% of words (~60) is enough to cover 40% of the total occurrences. This means that if we randomly take a word from the B2B list of recognized terms, the probability that it is used by several standards is relatively low; inversely, if we take randomly a tag from a B2B XSD specification, we are almost sure to have composing words largely adopted.

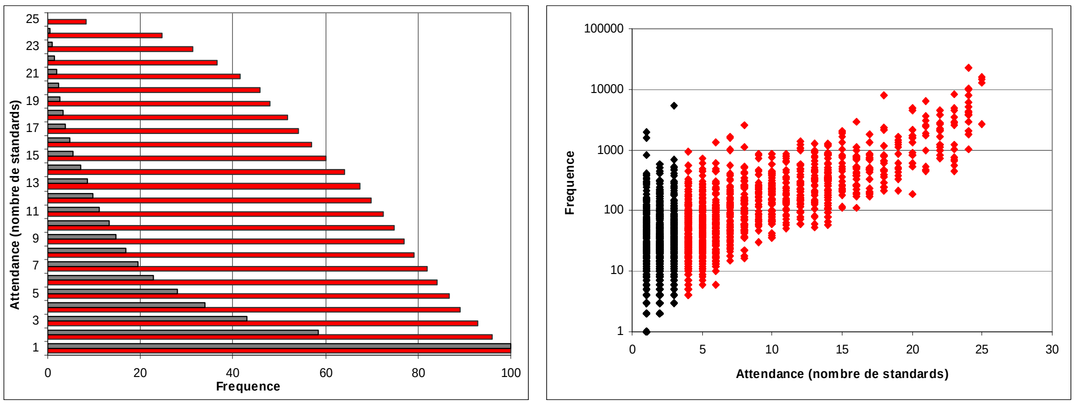


Figure 4.11 – Usage terms frequency and common terms stripes illustration

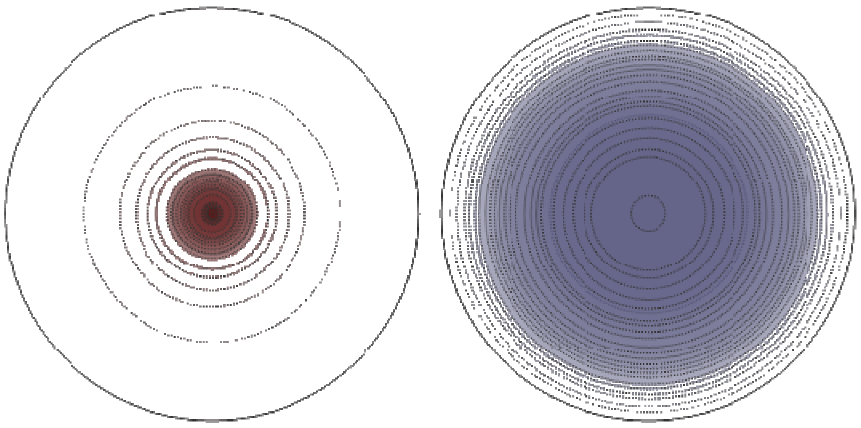
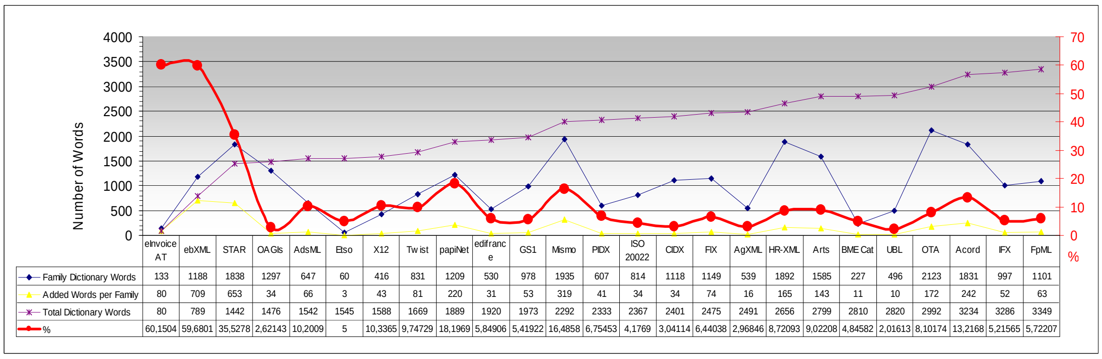


Figure 4.12 – Usage terms frequency and common terms circles illustration

This fact is confirmed by the figure below that details the construction of the list of normalized words as the sequential addition of a standard at once. Thus, as shown in Figure 4.13 and its associated table, by adding one standard at a time in a random order, we have observed that after few additions less than 10% of the words are really new, to obtain ~ 5% new words in the lasts standards to be added. We have noticed that these words usually represent terms characterizing the standard, but that the other, more general terms are already present in the global dictionary. So it shows that a dynamic list of words like this evolves smoothly and that a shared vocabulary emerges naturally.



	eInvoice AT	ebXML	STAR	OAGIs	AdsML	Etso	X12	Twist	papiNet	edifrance	GS1	Mismo
Family Dictionary Words	133	1188	1838	1297	647	60	416	831	1209	530	978	1935
Added Words per Family	80	709	653	34	66	3	43	81	220	31	53	319
Total Dictionary Words	80	789	1442	1476	1542	1545	1588	1669	1889	1920	1973	2292
%	60,1504	59,6801	35,5278	2,62143	10,2009	5	10,3365	9,74729	18,1969	5,84906	5,41922	16,4858

	PIDX	ISO 20022	CIDX	FIX	AgXML	HR-XML	Arts	BME Cat	UBL	OTA	Acord	IFX	FpML
Family Dictionary Words	607	814	1118	1149	539	1892	1585	227	496	2123	1831	997	1101
Added Words per Family	41	34	34	74	16	165	143	11	10	172	242	52	63
Total Dictionary Words	2333	2367	2401	2475	2491	2656	2799	2810	2820	2992	3234	3286	3349
%	6,75453	4,1769	3,04114	6,44038	2,96846	8,72093	9,02208	4,84582	2,01613	8,10174	13,2168	5,21565	5,72207

Figure 4.13 – Test for building a vocabulary with incremental addition

As detailed in the Filtering step above, we start from the B2B vocabulary (list of detected words) to implement a function to build a generic taxonomy based on the WorldNet hyperonymy and meronymy relations. These kinds of relations determine a basic hierarchy among discovered terms (tags), but although results are satisfactory for the vocabulary itself, the WordNet relations result to be too much generic and thus difficult to specialize for the domain. For this reason, we decided to go further in the implementation and to integrate what we call structural relations directly retrieved from XML sources.

4.3.3 Special Concern for “Bad Words”

As Figure 4.9 shows, a discrete number of unrecognised words still remain, at least at first sight. The analysis shows that these bad words are of the following type: mostly abbreviations (about 50%); about 30% are compound words not split by the system (for example compound words not written in UCC form like *worktime* or *preowned*); about 10% are words not included in the external dictionary; and another 10% are acronyms.

Several techniques can be implemented to improve the detection of hidden words. Our implementation of abbreviation discovery, based on a specific adaptation of the N-Gram algorithms, is able to detect more than 60% of them automatically. This in reality corresponds to 70% of total occurrences (for example *amt* => *amount* has 958 occurrences thus more important than *lqdty* with just one occurrence). Improving these results means: (a) adopting a more complex management of abbreviations to detect different words having the same abbreviation, (b) implementing NLP techniques to mine text documents that often come with XML files and; (c) improving the external dictionary capabilities. For the moment, these improvements have not been yet implemented.

In summary, we can say that solutions improving the quality of the extraction exist, but in order to fully exploit the potential of semantic technologies, a source document should be somehow *semantically well formed* alone. No semantic/linguistic algorithm will be able to understand the sense behind tags such as *AmortMktValDiffPct* or *setr.100.101*. The adoption of XML based standards has already notably improved the opportunity of automating the extraction of useful information, made this issue more apparent, and accelerated the drive towards convergence. But the cited cases show that simple patterns are both sufficient for ensuring a perfect extraction task.

4.4 A Basic Conceptualization Using SDMO

The semantics analysis produced so far represents a good start for the ontology generation, but it is not enough to obtain a complete representation of retrieved concepts. To achieve this topic we need more specific information about structural relationships. For this we go further in the information extraction to provide a clear distinction among concepts classes, concepts properties, and printable-types, as defined in our model in Section 3.1.4. This organization of concepts is fundamental to produce an ontology.

Starting from the semantics analysis results, we introduce in this section the retrieval of structural information from XML sources. We use SDMO as intermediary model for storing structures and semantics.

4.4.1 Deriving Conceptual knowledge

As already mentioned above, unlike simple text documents, XML documents provide likely annotated text with important information about objects and their structures. This helps in organizing concepts for the ontology to build. As also stated by Klein *et al.* [156], ontologies and XML schemata serve very different purposes. Ontology languages are a mean to specify domain theories and XML schemata are a means to provide integrity constraints for information sources (i.e., documents and/or semi-structured data). It is therefore not surprising to encounter differences when comparing XML schema with ontology languages. However, XML schema and OWL ontologies have one main goal in common: both provide vocabulary and structure for describing information sources that are aimed at exchange.

It is simple to imagine equivalences between OWL classes and XSD elements, like *Person* or *Employee* presented previously in Listing 4.2. As shown in

Listing 4.10 we can also retrieve information about relationships like sub-classes (e.g., *GeographicalCoordinate* is a *Coordinate*) and object properties (like *Longitude* and *Latitude* for *Coordinate*). These simple equivalences between OWL and XSD permit to provide not only concepts for a target ontology, but also hierarchies and structures for relating concepts.

We can summarize this conceptualization of XSD sources as a cloud of components to be isolated and used to obtain precise SDMO concepts. For instance Figure 4.14 illustrates a first classification of components using the SDMO classification of concepts as classes (for *Coordinate* and *Position*), class properties (for *Latitude* and *Longitude*), printable properties (for *AltitudeMeasure*, *DegreeMeasure* , *MinuteMeasure*), and printable types (just for *Measure*).

From this basic consideration, we define the extracted conceptual knowledge from XSDs as the **domain conceptualization.** We assume that given a set of XSD files *X*, it is possible to retrieve a complete set of related concepts *O* by a surjective mapping *m* [39], $m : X \rightarrow O$. The section below details

[39] A mapping from set A onto B is called surjective (or 'onto') if every member of B is the image of at least one member of A. ➔ $f : A \rightarrow B$ *is surjective if* $\forall b \in B \ (\exists a \in A \ (f(a)=b))$

this function as a transformation from XSD constructs to SDMO entities. An XSD construct can be either a simple schema component, or a specific combination of nested components.

```
<xs:element name="GeographicalCoordinate" type="CoordinateType"/>
<xs:complexType name="CoordinateType">
    <xs:sequence>
        <xs:element name="Longitude" type="PositionType"/>
        <xs:element name="Latitude" type="PositionType"/>
        <xs:element name="AltitudeMeasure" type="MeasureType"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="PositionType">
    <xs:sequence>
        <xs:element name="DegreeMeasure" type="MeasureType"/>
        <xs:element name="MinuteMeasure" type="MeasureType"/>
    </xs:sequence>
</xs:complexType>
<xs:simpleType name="MeasureType">
    <xs:restriction base="xs:decimal"/>
</xs:simpleType>
```

Listing 4.10 – Coordinate definition (excerpt from OAGIS standard)

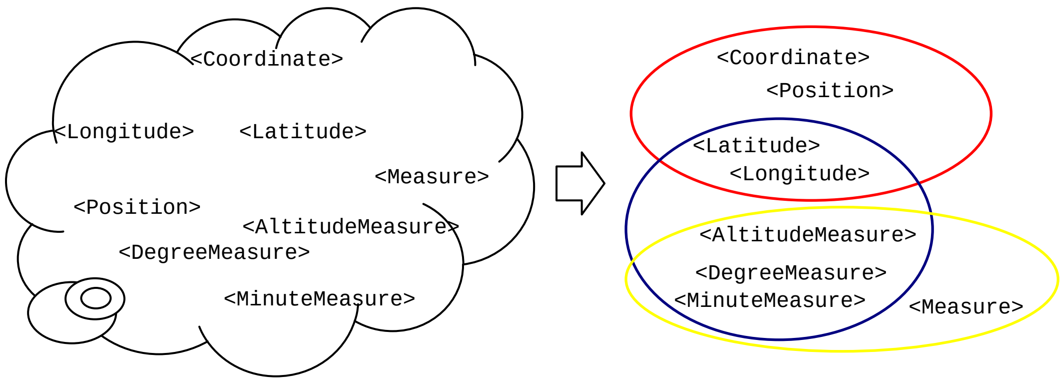


Figure 4.14 – Cloud of XML concepts and relative SDMO representation

4.4.2 XSD to SDMO Transformation Rules

As seen in Chapter 1, some systems already derive an OWL file from XML Schemas. More often it is obtained with a direct mapping of XSD components either to OWL entities or by adopting an intermediary conceptual model. In our system we follow the latter method, but rather than providing a close set of mapping procedures, we develop a system based on rules similar to [170]. The rules already defined are capable of mapping the most part of XSD constructs to our model. Moreover we propose some rules integrating some specific design practices. This behaviour ensures a better interpretation of XML schema sources with the possibility to improve the extraction of the conceptual information handling exceptions. Our rule-based system can be also extended simply adding new rules to fit other specific constraints. An example of a specific rule differentiate the usage of complex types that normally stand for concept classes, but in some cases define simple types with attributes.

```
<xsd:complexType name="TextType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="languageID" type="xsd:language" use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Listing 4.11 – Example of complex type definition for describing a data type extension (excerpt from UBL Unqualified Data Type standard components)

Listing 4.11 provides an example where, following our interpretation, the XSD complex type component better represents an object data type property than a concept class. The *TextType* complex type is used to extend the built-in XSD data type *string* and not to define a complex class with specific properties. Following the direct mapping defined in XML2OWL [55], this component corresponds to an OWL class. It also indirectly means that the typed *Name* element is a class. If this differentiation may seem somewhat trivial in this context, its usefulness will become clear to compare and merge several concepts extracted from different sources, as shown in Chapter 5. This transformation is detailed in Table 4.3 with rules 1 and 1a.

One must be aware that we work only on XSD, thus we target TBox statements and we do not integrate XML instances (that may be better compared to ABoxes). The reason of our choice is that within the B2B domain, message contents generally are private data (think to messages among a bank and its customers, should they be happy that we read their content just to discover a concept?). Albeit security problems surely limit the capacity of the system to discover ontological assertions. A system like our should be better exploitable if based on only XSD knowledge. In any case, as we have shown in Chapter 3 we are able to detect subsumption, equivalence, disjunction, and classification relationships between concepts. For example, we can observe that the concept *Drinker* subsumes *Person* ($\mathsf{Drinker} \subset \mathsf{Person}$) because it is less general (e.g., expressed through XSD file declaration as an extension). Also when tasting, *Coca* and *Wine* are disjoint classes (where a supposed XSD *WineTasting* element proposes a choice between these two classes). Furthermore, we can classify concepts as classes or properties and look for equivalences like *Owner* and *Person*.

We can summarize the rules we apply with the following macro principles:

- XSD complex types with complex content (i.e., a combination of attributes and sequence of elements like for *Coordinate*) produces SDMO classes, otherwise either properties or printable types;
- XSD elements can assume different facets as simple properties if they point to a simple type or a printable structure (like *AltitudeMeasure*), as classes if they declare a complex content, as a specialization of a class or a role, if they are named elements with declared type (like *GeographicalCoordinate*);

XSD Construct	SDMO element	Comments	Rule
Concepts			
Named *complexType*	SDMO Concept	SDMO concept nature (class, property or printable) can not be directly defined. More details on its content must be observed.	1
Named *complexType* with declared *simpleContent*	SDMO Concept datatype	Although complex type can have attributes, in this case a it merely represents a printable type	1a
Named *simpleType*	SDMO Concept datatype	Named *simpleType* can be only printable type	2
Element	SDMO Concept	As *complexType*, *element* declarations can be of different nature (class, property and datatype).	3
Element linked to a simple type, simplecontent or xsd datatype	SDMO Concept datatype	The same than 1a and 2	3a
Attribute	SDMO Concept datatype	Attributes are limited to printable (simple types) declaration	4
Relations			
Element with declared type	*Is a* among the derived concepts	The concept derived from the element is considered as a specialization of the type	5
Attributes with declared type	*HasDataType* among the derived concepts	Attribute can be only simple types, that in SDMO are considered as data-types	6
Attributes	*Properties*	Attributes are considered as properties of the concept derived from the element. Currently no difference is done among possible metadata properties and simple properties	7
sequence, all, choice	*Properties*	All sub-elements contained in the sequence are linked as properties	8
choice	Disjoint group of concepts	Within the SDMO Lattice of properties a flag count and links disjoint groups	9
extension et *restriction*	*is a* relationship	If c_1,c_2 are SDMO concepts derived from xsd entities, than we consider the relations as follows: - c_1 *extend* $c_2 \Rightarrow c_2$ *is a* c_1 - c_1 *restrict* $c_2 \Rightarrow c_1$ *is a* c_2	10
Union	*hasDatatype*	Elements of the *union* construct can be data-type for the derived SDMO concept	11
Any, anyAttribute	*hasDatatype*	This tag means that the tag the "super" element can contain any additional information. So a link to the generic data type *any* is added	12
simplecontent	hasDataType	Simple contents are considered as simple type, thus the SDMO concept is considered as a specialized data-type and a *hasDataType* relation is created for the derived concept	13
minOccurs, maxOccurs	Property cardinalities	Created as an attribute of the SDMO property relation	14
SubstitutionGroup	Equivalent concept name	Alternatively the concept name can be used, like a confirmed synonym	15

Table 4.3 – XSD to SDMO correspondent mapping basic rules

Table 4.3 provides the set of basic rules that we defined to realize the transformation from XSD to SDMO. The final mapping to OWL has been already proposed in Chapter 3 with the mapping of SDMO to OWL. In the table below, the first column lists the main XSD constructs, while the SDMO element column provides the corresponding domain conceptualization. The rule column identifies the rule number as detailed below in this section.

As already mentioned, a set of rules define the surjective mapping seen above. More formally, rules are defined following the formalisation adopted in the STASIS project to map different data models to their Logical Data Model in [170]. Our rules are as follows:

Table 4.4 – XSD to SDMO transformation rules

Rule 1: xsd:complexType mapping to sdmo:concept
Declarations of xsd:complexType are individuals of sdmo:conceptClass, sdmo:conceptProperty or sdmo:conceptDatatype. The choice among class or property is done dynamically depending on properties relations (see Rule 8), while datatype nature is verified by Rule 1a.
Transformation Rule
[TR1] For each x in xsd:complexType
create sdmo:concept(*c*), i.e. create *c* as a new instance of sdmo:concept
Rule 1a: xsd:complexType with xsd:simplecontent mapping to sdmo:conceptDatatype
Declarations of xsd:complexType are individuals of sdmo:conceptDatatype if it contains a xsd:simplecontent declaration.
Transformation Rule
[TR1a] For each x in xsd:complexType:
if x has xsd:simplecontent then add sdmo:conceptDatatype(*cdt*) i.e. update *c* as instance of sdmo:conceptDatatype *cdt*
Rule 2: xsd:simplType mapping to sdmo:conceptDatatype
Declarations of xsd:simpleType are individuals of sdmo:conceptDatatype.
Transformation Rule
[TR2] For each x in xsd:simpleType
create sdmo:concept(*c*) and add sdmo:conceptDatatype(*cdt*) i.e. create *c* as a new instance of sdmo:concept and update *c* as instance of sdmo:conceptDatatype *cdt*
Rule 3: xsd:element mapping to sdmo:concept
Definitions of xsd:Element are individuals of sdmo:concept. Similarly to Rule 1, the nature of the created concept is derived with Rule 8 and Rule 3a.
Transformation Rule
[TR3] For each x in xsd:element
create sdmo:concept(*c*), i.e. create *c* as a new instance of sdmo:concept
Rule 3a: xsd:element typed with a xsd:simpleType, xsd:datatype or xsd:complexType with xsd:soimplecontent
Definition of xsd:element are individuals of sdmo:concept with sdmo:conceptDatatype if it links through *xsd:type* attribute to xsd:simpleType, xsd:datatype or xsd:complexType declarations.
Transformation Rule
[TR3a] For each x in xsd:element:
if <x,y> in xsd:element and y in sdmo:conceptDatatype then add sdmo:concept(*c*) and add sdmo:conceptDatatype(*cdt*)i.e. create c as new concept and create link to sdmo:conceptDatatype *cdt* for *c*
Rule 4: xsd:attribute mapping to sdmo:conceptDatatype
Declarations of xsd:attribute are individuals of sdmo:concept with sdmo:conceptDatatype.
Transformation Rule
[TR4] For each x in xsd:attribute
create sdmo:concept(*c*) and add sdmo:conceptDatatype(*cdt*) i.e. create *c* as a new instance of sdmo:concept and update *c* as instance of sdmo:conceptDatatype *cdt*
Rule 5: xsd:element with declared xsd:type mapping to sdmo:isa or sdmo:hasDataType relation
Definitions of xsd:element *x* with declared xsd:type *y* are related with the referred complex/simple type by the sdmo:isa/sdmo:hasDataType R/R_{dt} relation.
Transformation Rule
[TR5] For each <x,y> in xsd:element and y in xsd:type
if <x,y> in xsd:element and y in sdmo:conceptDatatype then create sdmo:hasDataType($xR_{dt}y$) else create sdmo:isa(*xRy*) i.e. create *Rdt* as a new instance of sdmo:hasDatatype if the related *type* attribute is

```
instance of sdmo:conceptDatatype else
 create r as instance of sdmo:isa relation
```

Rule 6: xsd:attribute with declared xsd:type mapping to sdmo:hasDataType relation

Declarations of xsd:simpleType *x* with declared xsd:type *y* instance of sdmo:conceptDatatype engender sdmo:hasDatatype relation R_{dt}.

Transformation Rule

```
[TR6] For each x in xsd:attribute and y in xsd:type
```

```
if <x,y> in xsd:attribute and y in sdmo:conceptDatatype then
create sdmo:hasDatatype($xR_{dt}y$) i.e.
 create Rdt as a new instance of sdmo:hasDatatype relation
```

Rule 7: named xsd:complexType with declared xsd:attribute mapping to sdmo:hasProperty relation

Declarations of named xsd:complexType *x* with xsd:attributes *y* engender instances of sdmo:hasProperty *r* relation.

Transformation Rule

```
[TR7] For each <x,y> in xsd:complexType and y in xsd:attribute
```

```
create sdmo:hasProperty(xRy) i.e.
 create R as a new instance of sdmo:hasProperty relation
```

Rule 8a: Named xsd:complexType with xsd:sequence or xsd:all, or xsd:choice mapping to sdmo:hasProperty relation

Declarations of named xsd:complexType *x* having sub-elements *z* contained in xsd group constructs (xsd:all, xsd:sequence and xsd:choice) *y*, are instances of sdmo:hasProperty relation *r*.

Transformation Rule

```
[TR8a] For each <x,y,z> in xsd:complexType and ((for each <y,z> in (xsd:sequence or
xsd:all or xsd:choice) and (for each z in xsd:element)) then
```

```
create sdmo:hasProperty(xRz) i.e.
 create R as a new instance of sdmo:hasProperty relation
```

Rule 8b: xsd:element with inline xsd:complexType with xsd:sequence or xsd:all or xsd:choice mapping to sdmo:hasProperty relation

Definitions of xsd:element *x* having with anonymous xsd:complexType *w* with sub-elements *z* contained in xsd group constructs (xsd:all, xsd:sequence and xsd:choice) *y*, engender instances of sdmo:hasProperty relation *r*.

Transformation Rule

```
[TR8b] For each <x,w,y,z> in xsd:element and (for each <w,y,z> in complexType and
((for each <y,z> in (xsd:sequence or xsd:all or xsd:choice) and (for each z in
xsd:element))) then
create sdmo:hasProperty(xRz) i.e.
 create R as a new instance of sdmo:hasProperty relation
```

Rule 9a: xsd:choice mapping to sdmo:PropertyGroup and sdmo:disjointGroups

Sub-group *P* of elements *y* of declarations of xsd:choice *x* are individuals of sdmo:disjointGroups.

Transformation Rule

```
[TR9a] For each <x,y> in xsd:choice then (
 create sdmo:propertyGroup(P) and if y in xsd:element  then
   add sdmo:propertyGroup(c)
)
 update sdmo:disjointGroups($P_i$) i.e.
 create P as a new instance of sdmo:propertyGroup and
 add the sub-element sdmo:concept c as instance of the new property group P and
finally
 update generated groups Pi as instance of sdmo:disjointGroups
```

Rule 9b: xsd:sequence, xsd:all, sxd:group mapping to sdmo:PropertyGroup

Sub-group *P* of elements *y* of declarations of xsd:sequence/xsd:all/xsd:group *x* are individuals of sdmo:propertyGroup *P*.

Transformation Rule

```
[TR9b] For each <x,y> in (xsd:sequence or xsd:all or xsd:group) then (
```

```
  create sdmo:propertyGroup(P) and (
  for each y in xsd:element  then
   add sdmo:propertyGroup(c)
 )
 create P as a new instance of sdmo:propertyGroup and
 add elements sdmo:concept c as instance of the new group P i.e.
```

Rule 10a: xsd:extension mapping to sdmo:isa relation

Declarations of xsd:complexType *x* with xsd:extension *y* are individuals of sdmo:isa relation *R*.

Transformation Rule

```
[TR10a] For each <x,y> in xsd:complexType and y in xsd:extension then
 create sdmo:isa(yRx) i.e.
 create R as a new instance of sdmo:isa
```

Rule 10b: xsd:restriction mapping to sdmo:isa relation
Declarations of xsd:simpleType *x* with xsd: restriction *y* are individuals of sdmo:isa relation *R*.
Transformation Rule
[TR10b] For each <x,y> in xsd:simpleType and y in xsd:restriction then create sdmo:isa(*xRy*) i.e. create *R* as a new instance of sdmo:isa

Rule 11: xsd:union mapping to sdmo:hasDataType relation
Declarations of xsd:simpleType *x* with declared xsd:union *y* engender individuals of sdmo:hasDatatype R_{dt}.
Transformation Rule
[TR11] For each <x,y> in xsd:simpleType and for each y in xsd:union then (add sdmo:hasDatatype($xR_{dt}y$)) i.e. add elements sdmo:concept *c* as instance of sdmo:hasDatatype R_d

Rule 12: xsd:any and xsd:anyAttribute mapping to sdmo:hasDataType relation
Declarations of xsd:complexType *x* with xsd:any and xsd:anyAttribute *y* are individuals of sdmo:hasDatatype linked to the special sdmo:conceptDatatype(*#any*).
Transformation Rule
[TR12] For each <x,y> in xsd:simpleType and y in (xsd:any or xsd:anyAttribute) then create sdmo:hasDatatype(*xRdt(#any)*) and add sdmo:conceptDatatype(*cdt*) i.e. create *Rdt* as a new instance of sdmo:hasDatatype

Rule 13: xsd:simpleContent mapping to sdmo:hasDataType relation
Declarations of xsd:complexType *x* with xsd:simplecontent *y* are individuals of sdmo:conceptDatatype.
Transformation Rule
[TR13] For each <x,y> in xsd:complexType and y in xsd:simplecontent then update sdmo:conceptDatatype(*c*) i.e. update *c* as a instance of sdmo:concept sdmo:conceptDatatype *cdt*

Rule 14: xsd:minOccurs and xsd:maxOccurs mapping to sdmo:hasProperty cardinality attribute
Definitions of xsd:element *x* with declared xsd:minOccurs/xsd:maxOccurs *y* and value *n* are individuals of sdmo:hasProperty:cardinality.
Transformation Rule
[TR14] For each <x,y> in xsd:element and (if y in xsd:minCardinality then update sdmo:hasProperty:cardinality:min(*n*) else if y in xsd:maxCardinality then update sdmo:hasProperty:cardinality:max(*n*) i.e. update *hasProperty(r)* cardinalities.

4.4.3 Some Elements of Comparison

In this Section, we provide some elements about the evaluation of our mapping with respect to other similar implementation seen in the survey of Chapter 1 and 3. We have not looked over the produced ontology using exact measures like precision and recall. This is motivated by the fact that defined mappings from the XML Schema meta-model to another conceptual model is more an interpretation specific to the targeted model then an objective transformation. It is highly dependent from the analyst making the operation itself. At most we can measure the number of considered constructs, to estimate if there is information lost in the translation. Another way to measure the quality could be done at the usage of the resulting ontology itself, like how many reasoning elements can be calculated from it. This is a preliminary step of the whole generation process difficult to evaluate. Thus, until now no one has provided such test cases.

Further we have evaluated some available systems providing a detailed XML Schema transformation, which are XML2OWL [55], OWLMAP [57], LDM [170] with our system, called Janus. Mainly our analysis highlights the following aspects of the different systems:

- Number of XSD constructs, that permit to appreciate the completeness of the map with the possibility to maintain as much information as possible;

- XML instances, which normally means that the resulting ontology is directly populated with OWL individuals. However as far as we know we remark that no systems further investigate the possibility to use instances knowledge to extend the ontology expressivity. At most XML instance with a back engineering is transformed in pseudo XML Schema and used to produce the mapping to OWL;
- Extensibility just says if the system can be simply extended to add more XSD constructs or rules;
- Exception management tells if a system is able to look forward the simple direct mapping and manage exception of specific design practices;
- Semantic normalisation looks at the capacity of the system to resolve linguistic and semantic normalisations (like abbreviations, tag lemmatisation and so on);
- Concept structures evaluates the possibility to resolve hierarchical, properties and datatype relations;
- Concept relations provides a quality measure about the richness of semantic relations extracted like equivalent classes, functional properties and other specific relations that can subsist among constructs
- OWL expressivity is a theoretical interpretation of the retrieved information expressivity using the DL naming convention reported in Table 1.1 (the corresponding value is an evaluation we made on the basis of the available documentation).

Table 4.3 summarizes the evaluation described above.

	XML2OWL	**OWLMAP**	**LDM**	**Janus**
N. of XSD construct	8	9	18	19
XML instances	✓	✓		
Extensible			✓	✓
Exception management	limited	limited	✓	✓
Semantic normalisation				✓
Concept structures	✓	✓	✓	✓
Concept relations	limited	✓	limited	✓
OWL expressivity	ALUHN	tbd	tbd	*ALHOINQF*(D)

Table 4.5 – XML Schema information extraction considerations

Table 4.5 details the XML Schema constructs that are considered for the information extraction of each system.

As we can see, our system improves existing solutions. This thanks to the integration of more XSD constructs and of specific extensible rules. As already mentioned at this level, we cannot provide real quality estimation because of the objectivity of the resulting mapping of XSD constructs. Nevertheless we can at least be sure that our approach provides a satisfactory transformation.

[XSD construct]	**XML2OWL**	**OWLMAP**	**LDM**	**Janus**
All	✓	✓	✓	✓
Annotation			✓	
Any			✓	✓

Appinfo				
Attribute		✓	✓	✓
AttributeGroup		✓	✓	✓
Choice	✓	✓	✓	✓
Complexcontent				✓
ComplexType	✓	✓	✓	✓
Documentation				
Element	✓	✓	✓	✓
Extension		✓	✓	✓
Group		✓	✓	✓
Import			✓	✓
Include			✓	✓
Restriction		✓	✓	✓
Sequence	✓	✓	✓	✓
SimpleContent				✓
SimpleType	✓	✓	✓	✓
SubstitutionGroup		✓		✓
Union			✓	✓
List			✓	
Min/Max Occurs	✓	✓	✓	✓
Namespace		✓		

Table 4.6 – Details on the extracted XSD constructs for the transformation to ontology

4.5 Measuring XSD Semantics and Structures

Before concluding this chapter, we like to stress out the importance of input sources. This issue reflects the well known phrase "*Garbage In, Garbage Out (GIGO)*[40]" in computer science. That means computers will unquestioningly process the most nonsensical of input data and produce nonsensical output. Indeed, to automate the ontology generation as best as possible, the quality of the output is directly dependent from the definition of input elements. So when retrieving information it is important to know how sources are built to be able to decide if a source can be included in the corpus or not. In our use case, we are building a semantic network of concepts, thus it is obvious that having correct semantics and structure is an essential condition to get better quality results.

Regarding XML Schema instances, XML specifications already provide a definition of *well-formedness* of XML documents. But it focuses on XML entities as logical and physical structures that in an XML document must be properly nested. This is limited to the fact that no start-tag, end-tag, empty-element tag, element, comment, processing instruction, character reference, or entity reference can begin in one entity and end in another. No concerns are done over semantics and conceptual structures of XML entities.

For this we add the definition of **XML documents semantically well structured** in order to define some basic rules to have real semantics and concepts defined in an XML schema document. This kind of classification of such documents can be used to settle on the adoption of either a specific

[40] http://en.wikipedia.org/wiki/Garbage_In,_Garbage_Out

algorithm or excluding it, thus to be able to evaluate input source quality before adding them to the input corpus.Thus we say that a concept *c* derived from an XML Schema source is **semantically valid** if its label is composed by clearly identifiable words belonging to a standard common dictionary (like the English Oxford dictionary for the English language), rather than unrecognized abbreviations, acronyms or any other sequence of chars.

On the same line we say that a set of extracted concepts *C* is **well structured** if the ratio between obtained SDMO structural relationships (*#Rs*) and the total number of extracted concepts (*#C*) is higher of a predefined threshold (α). This last definition prevents the integration of only flat definition of XML elements. For example, applying this test we were able to discard some XBRL files. Indeed their specifications are defined with the help of XLink constructs that our system was not able to detect. As consequence, retrieved information presented some inconsistencies that produced some bad concept definitions.

Finally, following the definitions above we say that a non empty set of SDMO concepts *C*, obtained from a given source, is **semantically well structured** if at least a considerable number of its concepts are semantically valid (on the basis of a predefined threshold β) and *C* itself is well structured. We adapted this measures to our corpus using α = 0,75 and β = 0,5 following some empirical observations and we were able to slightly improve the acquisition step. This is at the price of losing some information that in certain cases could be integrated with simple human intervention.

4.6 Conclusion

This Chapter provides important elements to realize the automatic generation of ontology from XML Schemas. We show that even though concepts are sometimes defined using unclear semantics including “bad words”, at least for the B2B domain, we are able to obtain a common vocabulary of terms. This vocabulary includes at least 95% of all words used for defining XML components. Consequently, we demonstrate that collected sources contain a common base of information useful to build the conceptual knowledge.

We observe also that the generation of a taxonomy with only information extracted from tag labels is not enough, even with the introduction of WordNet relations like meronymy and hyponymy. Reasons for this are motivated by the fact that a generic dictionary is inadequate to provide information on a too specific domain. This also highlights the inadequacy for our use case of those systems completely based on WordNet surveyed in Chapter 1. Consequently it consolidates our choice to provide a new system and to go further in the process of information extraction.In addition, this Chapter presents our contribution on the transformation of XML schemas that we prove to be more complete than others, and thus capable of improving current B2B technology. Finally with the adoption of SDMO as a semantic intermediary model, we are able not only to provide a transformation to OWL of each source, but also to develop a system improving the capabilities of merging different sources thereby transformed. This system, called Janus, is described in the next Chapter.

Chapter 5.

Janus:

Automatic Ontology Building System

Over the past ten years, the Semantic Web wave has shown a new vision of ontology use for application integration systems. Researchers have produced several software tools for building ontologies (like Protégé [79] or OntoEdit [176]) and merging them two by two (like FCA Merge [52] or Prompt [50]) or producing alignments (like S-Match [154], OLA [177], Mafra [178], H-MATCH [54], COMA [53]). Nevertheless these solutions, as well as adopted ontology building methodologies, are mainly human driven or, as shown in Chapter 1, sometimes assisted by semi-automatic software tools.

Limitations to their adoption for integration of enterprise applications, among others reasons, are: (i) the lack of tools capable of extracting and acquiring information from a large collection of XML files (the “de-facto” format for applications information exchange definition); (ii) the complexity of aligning and merging more than two sources, a complex task excessively consuming of computational time; (iii) the difficulty of validation based on background knowledge hard to produce and maintain.

The aim of this Chapter is to introduce Janus, the software that we have developed. This system is an implementation of our approach to ontology generation integrating SDMO, extracting information from XML Schemas and is capable of providing a solution to the limitations described above. Indeed as we show with our experimental results, it is able to automatically generate and maintain a collective memory resource that facilitates the discovery of alignments when matching concepts in a given domain with satisfactory results.

The Chapter is outlined as follows. In the first Section we introduce our system. We firstly depict a common problem of current integration approaches to generate ontology from multi source inputs. As consequence of the shortcomings of the studied architectures we propose our solution to solve the multiple inputs integration problem. We finish the first section with the overall presentation of our prototype.

Throughout Section 5.2 we present some of the difficult implementation details and highlight some choices we made to solve them. The first challenge faced was the generation of the two lattices capable of resolving a large corpus in acceptable computation time. After an explanation of the lattice of shared terms and the lattice of properties we focus on some implementation features with their final algorithms. Then we illustrate the generation of the *similarity network* that provides a global graph of an SDMO instance. Another topic we studied was the balance between a measure with the best results and a measure with acceptable constraints for an incremental system. We present the final decision with motivations.

Section 5.3 details the integration process and the adoption of the similarity network to unveil similar concepts in a faster way. In a first step we explain how multiple sources are integrated in an efficient manner, using SDMO and then we present the procedure/algorithm we developed.

In section 5.4 we present our experimentation that provides different elements for the final evaluation of our work. Among them we have an evaluation of the speed and scalability of the system, its capacity to maintain information in a compact way, a quality measure of the system and some considerations on its performances. In addition we also show the graphical interface that we have developed. The final section provides an overall analysis and concludes this chapter.

5.1 Janus

In the golden age of the Roman Empire Janus was a god, the god of gates, doors, doorways, beginnings and endings. Janus was usually depicted with two heads looking in opposite directions. We have chosen this name for our system because its representation fits our purpose: a system able to look at different directions at once, and a system that merges different views into one.

Throughout this Section we introduce our implementation of SDMO and XML Schemas conceptualization to attain a prototype that allows users to automatically generate a first skeleton of an ontology.

5.1.1 Handling Multi Sources Input

As seen in the first Chapter there are several possible approaches to automatic generation. Among them we motivated in Chapter 3 our choice to adopt the approach including an intermediary conceptual model because it reduces the complexity of the integration process. In this subsection we also discuss another problem that we encountered even with the approach we chose and we detail the solution we adopted in our implementation.

5.1.1.1 Ontology Merging vs. Progressive Merging Dilemma

In Section 1.3 we depicted the matching problem and our vision about the different operations when matching two or more ontologies (i.e. learning, matching, alignment, merging and mapping). To our knowledge, systems following the intermediary model approach begin the automatic generation process by filling the conceptual model from a given input source. Then they proceed with the transformation of the model into the corresponding ontology. Finally sources are merged. This process is mostly studied for only two input sources simultaneously and it is almost the same whether input sources are ontologies or schemas. This is probably due to the hypothesis that the process reiterated over more than two inputs is still adequate and produces the same result. However, the integration of more than two sources can be carried out in different ways. Indeed, as illustrated in Figure 5.1, sources can be added either in a single step process, that we call **direct merging** (Figure 5.1 (a)), or recursively one source at a time, in what we call **progressive merging** (Figure 5.1 (b)).

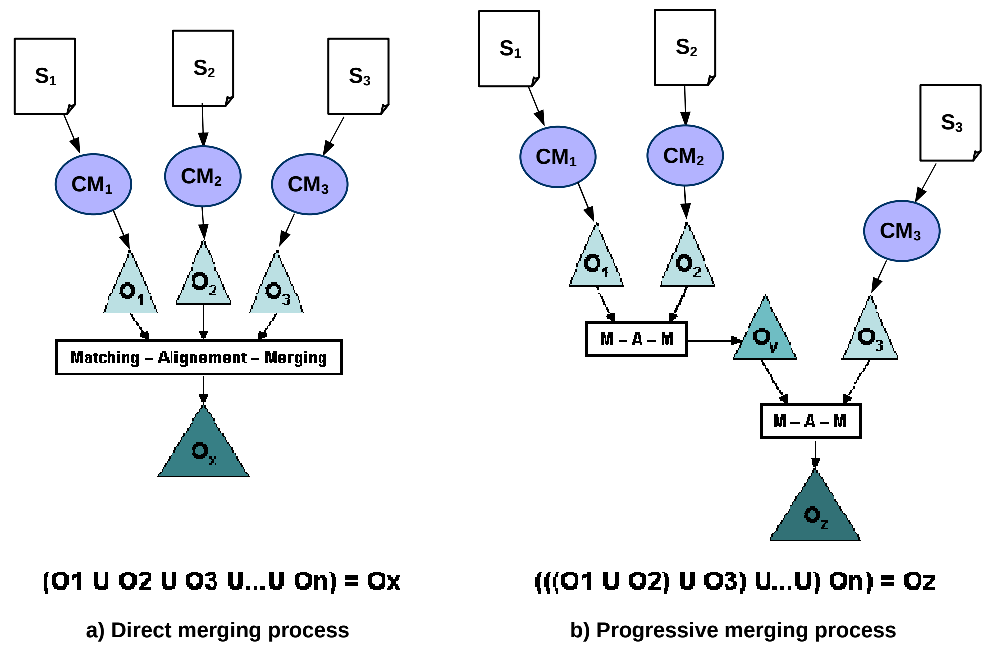

S1
S2
S3
CM1
CM2
CM3
O1
O2
O3
Matching – Alignement – Merging
Ox
(O1 U O2 U O3 U...U On) = Ox
a) Direct merging process
S1
S2
S3
CM1
CM2
CM3
O1
O2
M – A – M
Oy
O3
M – A – M
Oz
(((O1 U O2) U O3) U...U) On) = Oz
b) Progressive merging process

Figure 5.1 – Direct merging (a) and Progressive merging (b) processes representation

At first we implemented the generation process following the progressive merging approach, but we observed that the resulting ontology was different depending on the sources' integration order. Going further in the analysis we deduced that the main problem was triggered by the merging operation. As a consequence, merging sources in a single operation can produce different outcomes from merging sources progressively. Of course this problem did not arise in the systems we studied because the process does not change with only two input sources.

Reasons leading to different final results are due to the fact that the merging operation often implies choices about the best representation to maintain in the integrated ontology. This fundamentally means that we lose information after each iteration. Such information can be useful in certain circumstances, depending on the matching and merging algorithms adopted. For example the method proposed by FCA-Merge [52] is based on individuals who appear in ontologies to merge. So doing, concepts having the same individuals are then supposed to be merged. But what happens if we merge two ontologies at once? The list of concepts to discard can be different. The same happened for us when we used algorithms based on statistical calculations. By adding a source, values can change and consequently the merging operation too.

Finally the best solution should be the merging of all sources at once rather than progressively. But remember that in our use case we made the hypothesis that sources can be added *on the fly*, consequently this solution does not fit our needs of dynamism as defined in Section 2.2.2 and we opted for the method described below.

5.1.1.2 Approach to the Adoption of Progressive Merging

As seen above the approach of progressive merging when developing ontologies automatically can produce inadequate results. For this reason we designed SDMO to maintain a greater quantity of information necessary to produce progressive merging limiting data loss. This requirement is expressed by the completeness rule expressed in Section 3.2. Subsequently we modified the progressive merging approach in order to integrate sources at the conceptual model level rather than at ontology level as depicted in Figure 5.2. The main difference is that here the ontology is just a view of the resulting conceptual model and not the complete final outcome of the integration process.

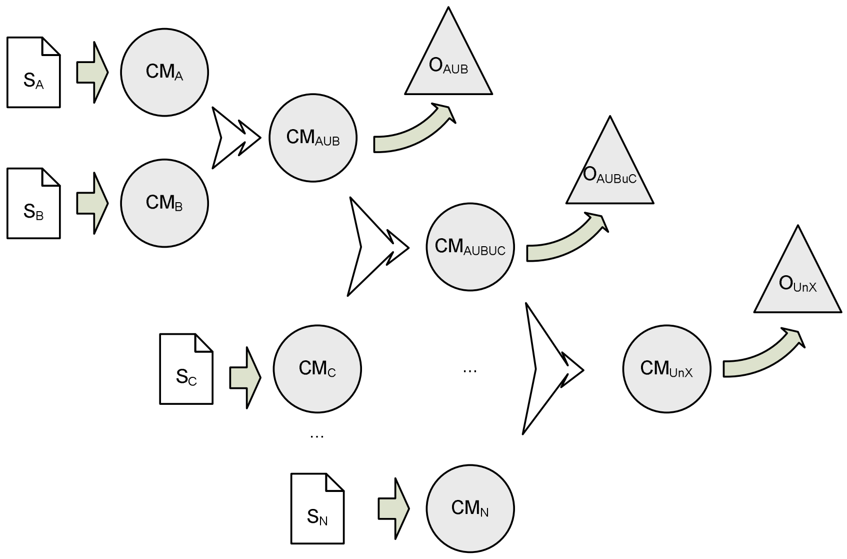


Figure 5.2 – Progressive merging of concept model approach

Another difference is that using OWL as serialization format for large scale inputs leads to a reckless size of the resulting file. On the contrary the storage of the conceptual model can be significantly reduced by using other methods of storage less verbose and more efficient. With the progressive merging at concept model level approach we leave to the transformation task the possibility to generate the ontology (i.e. in OWL) using only more relevant concepts and relationships or, if needed, the whole conceptual model content.

Of course this method presents some disadvantages. These disadvantages are dictated by the fact that we have no direct control on the evolution of the ontology that we generate at different stages of the progressive integration. As a consequence if a user modifies the ontology rather than sources or the model itself, we are not able to maintain changes unless we build an inverse transformation from the ontology to SDMO. Another solution could be to maintain all generated ontologies as versions and use existing tools, like Anchor Prompt [51], to maintain coherence among them. However our main goal is to maintain a "memory" for matching engines and this is assured by the model. Thus this last point remains out of the scope of our thesis.

In the following Section we specify our system implementation and provide more detail for each implemented module.

5.1.2 Overall Presentation

Janus is a tool that enables the automatic generation of ontologies from XML Schemas. In practice it is an implementation of the system described throughout previous Chapters and Sections. Figure 5.3 shows the overall architecture of Janus. We can identify the modules described above.

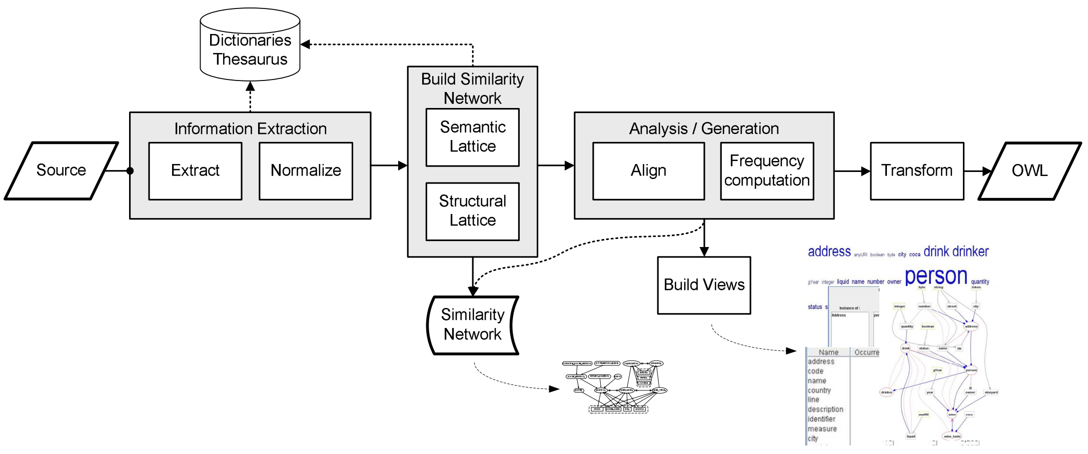


Figure 5.3 – Janus overall architecture

The extraction task represented by the **Extract** arrow and **Normalize** rectangle in Figure 5.3 supplies the knowledge needed to generate the ontology. This knowledge is merely composed by candidate concepts, properties, printable types, relationships of different nature and at the same time it contains counters and ranks for each element. Implemented techniques for knowledge acquisition are a combination of different types, such as: NLP (Natural Language Process) for morphological and lexical analysis, association mining for calculating term frequencies and association rules, semantics for finding synonymy, and clustering for grouping semantic and structural similar concepts. We call **XML Mining** the adaptation of these techniques applied to XML schemas.

XML Mining is used to parse sources to extract XML constructs, as specified in Section 4.4.2, and to process XML tags declarations. In addition it also includes a pre-matching treatment that aims to mutualize element's processing that are clustered in a Galois Lattice and Formal Concept Analysis based form. This treatment provides as output a pre filled model ready for automatic analysis.

The following step is **build semantic network** represented by the corresponding block in Figure 5.3. This step finalizes the model integrating information coming from external sources, like other existing ontologies or thesaurus. Moreover at this stage we do not look at similar concepts to be merged, but only execute matching algorithms to collect as much correspondences as possible among them. All these connections are stored and maintained in the model in order to be quickly detected and not recalculated in future integrations.

The **Analysis** step aligns correspondences and looks for equivalent concepts to be integrated. This step establishes the best similarities and analyses the model to unveil new possible relations and correspondences not directly detected by matching algorithms and computes frequency and rank measures.

The **Generation** step finalizes the meta-model used by the tool into a final semantic network. The final model can be serialized in OWL following the transformation described in Section 3.2, built by the **Transform** module. The **Filtering** step can integrate new matching algorithms or simply refines concepts' correspondences to update the global semantic network. Finally the **Build Views** module derives useful views from the network provided to users.

The following section details some implementation features about the construction of the SDMO instances.

5.2 Implementation Features

In this Section we present some features of our system. The purpose of this presentation is to detail some specific solution we adopted to build the SDMO instances.

5.2.1 Building the Shared Terms Lattice

In Section 3.1.3.2 we formally defined the Shared Terms Lattice as a way to maintain relationships among concepts having common words in the label name. Indeed different designers define tags' labels with different composed terms even when expressing the same concept. For example if we consider the following tags: `Address`, `RetailTransactionAddress`, `AddressInformation`, `PostalAddress`, `StructuredLongPostalAddress`, `ScreeningPostalAddress`, `PostAddr` or also `WorkLocation`, `DeliveryReceiptLocation`..., they are probably all expression of the same concept, an *address*, with only a different level of detail and usage context. But if it is humanly simple to understand this correspondence, a machine requires more elements. For this we implemented the Galois Lattice based system because it allows the creation of a graph where nodes have common elements in a simple and efficient way.

To illustrate the construction of the graph we consider as example the following tags: `Address`, `PostalAddressBase`, `ScreeningPostalAddress` and `DeliveryLocation`. The nodes of the lattice and the correspondent graph are illustrated in Figure 5.4. In the picture bold rectangles correspond to normalized tags, while rectangles with thinner lines represent decomposed labels with only subparts of the respective compound word. The number represents the word occurrence of each node. We also group together nodes with the same number of words (e.g. *postal_address* and *screening_address* as belonging to the G_2 group).

Furthermore we add the synonym relationship among *address* and *location* to complete the graph with semantic relations and a linguistic relation among *addr* and *address*. This graph allows us to highlights our starting hypothesis that address is the main concept among those provided in input. About the graph elements, following the definition given in Section 3.1.3.2 we say that:

- A **label** of a node is the composed word defining the node (e.g.: *postal_address*);
- a word w_i belongs to a node if it is contained in the node label (e.g.: *postal* belong to the node *postal_address*, *screening_postal_address* and *postal*);
- the **length** of node is the number of words composing the node label (e.g.: *postal_address* has length 2);
- the cardinality of a node corresponds to the number of times that the whole label appears in tags (e.g.: cardinality of *PostalAddress* is 2 ➔ `[PostalAddressBase, ScreeningPostalAddress]);`
- the **upper nodes** of a node n_i with length *l* are all nodes of length *l+k*, where k >= 1, containing the words belonging to n_i (e.g.: *postal_address* is upper node for *address* and *postal*);
- inversely **lower nodes** of a node n_i of length *l* are all nodes of length *l-k*, where *1 <= k < l*, where their label is composed by a word of the node n_i (e.g.: *address* and *postal* are lower nodes of *postal_address*)

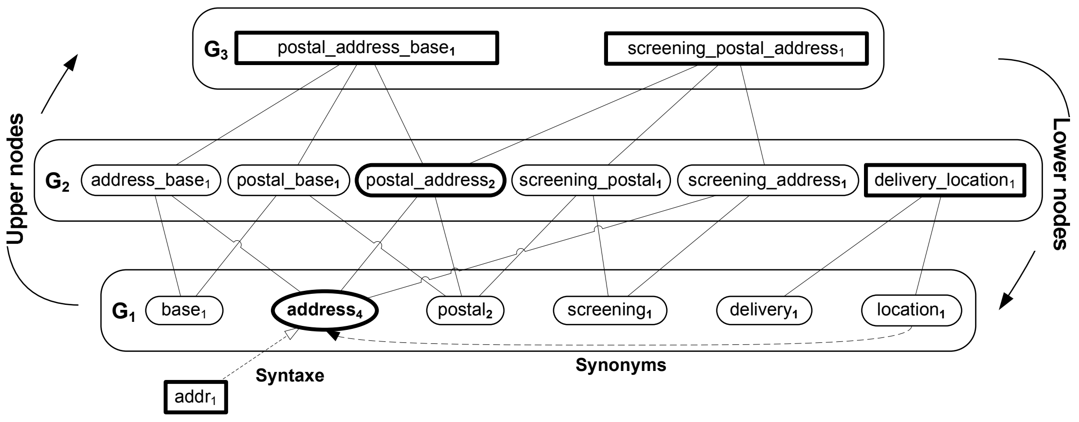


Figure 5.4 – Galois Lattice nodes representation

Thus as shown by the example above this graph provides a very useful way to organize concepts by their label name. Furthermore it offers a very fast computation to adapt string matching to XML tag naming features. However we can observe that the obtained complete graph contains worthless nodes, such as *screening_postal*. Consequently we built an algorithm that reduces the size of the graph (this can be of many orders of magnitude) dropping all nodes that do not provide any supplementary information. These nodes can be quickly recognised since all those nodes have the same cardinality as their upper nodes (e.g.: `screening_postal` has the same cardinality than `screening_postal_address` so it is dropped, indeed `postal_address` is maintained because its cardinality is greater than `screening_postal_address` and `postal_address_base`).

Appling this step to our example we obtain that G_1 = *{address (4)}* (considering location as valid synonym of address), G_2 = *{postal_address (2), delivery_location (1)}* and G_3 = *{screening_postal_address (1)}*

This step looks for nodes most representative of a concept. These are the nodes having the higher cardinality within a sub-tree of the graph. Following the upper and lower relationships between nodes we leave such nodes having the higher cardinality, with priority for nodes with higher length in the case of equal value. By applying this last step we obtain that the root node is `Address`, which represents the main concept for our summarized input tags set.

The overall algorithm that constructs the Word Lattice is detailed in Listing 5.1. The input of the algorithm is a list of normalized tags. Each tag is added to the lattice and is recursively decomposed to create sub nodes at the same time. If a node or a sub-node already exists then the node occurrence value is incremented and the function stops.

The algorithm we implemented creates all sub-nodes even when they could seem superfluous. Indeed they are created as soon as they are met because the input list is crossed sequentially and other tags could contain the same sub-nodes or be themselves one of them. So doing, we are sure to finally have the right occurrence values and edges; we need only walk through the graph again to find them out. It is only at the end that we remove useless nodes.

5.2.2 Building the Properties' Lattice

In Sections 3.1.3.3 and 3.1.3.4 we defined structural relationships as hierarchy of concepts. This hierarchy can be established among different kinds of concepts, i.e. among concept classes, between a concept class and its properties or between properties and printable types. With the lattice of properties we focus on a data structure organization for concept classes and their properties. Similarly to the Shared terms lattice it provides a fast algorithm to detect common groups of properties and consequently detect close concepts on the basis of their structures. In this section we detail the construction of the lattice of properties.

To illustrate its construction and usefulness we now consider two schemas defining two XML entities that we simply call *A* and *B* as shown in Figure 5.5. From these two simple schemas, derived XSD components correspond to concept classes and concept properties in an SDMO instance. Respectively *A* and *B* are classes, and in their normalized name *building_number*, *street_name*, *city_name* and *postal_code* are properties.

As mentioned above the only semantic relation holding among concepts' labels is not enough to say if two concepts are indeed equivalent. For instance using WorldNet we see that *address* is semantically related to *name* but this information requires to be consolidated or rejected from the automatic system. Moreover if we are not able to detect any semantic or linguistic relation among two concepts, we lose such information. Thus we integrate this second view of relations among concepts that provides further information to this purpose. Albeit the example is voluntarily trivial. On the basis of their complete structural resemblance (i.e. the sub elements, or properties), it clearly shows that these two concepts can be considered equivalent.

```
Let be NTL the input data list of normalized tags;
Let be WL the lattice of shared terms;
Let be N a lattice's node;
Let be T a normalized tag;
Let be Nt := N(T) the correspondent lattice node for the tag T;
Let be Ln := L(N) a lower-node for N;
Let be Un := U(N) an upper-node for N;
Let be GLn := GL(N) the group of lower-nodes Ln of N;
Let be GUn := GU(N) the group of upper-nodes Un of N;
Let be Gx the group of nodes having length = x (node label composed by x words);

BuildSharedTermsLattice(NTS)
. For each T in NTL do
. . checkNode(Nt)
. end for
. finalizeSharedTermsLattice(WL)
End BuildSharedTermsLattice

Function checkNode(N)
. if N in WL then
. . increment counter of N
. . if GLn > 0 then
. . . for each Ln of GLn do
. . . . increment counter of Ln
. . . end for
. . end if
. else
. . create new node N
. . if GLn > 0 then
. . . for each Ln of GLn do
. . . . create lower-node edge N -> Ln
. . . . create upper-node edge Ln -> N
. . . . checkNode(Ln)
. . . end for
. . end if
. end if
End function

function finalizeSharedTermsLattice(WL)
. for x = 0; x < WL.deep-value; x++; do
. . for each N of Gx do
. . . if GUn > 0
. . . . if all Un have Un.counter >= N.counter then
. . . . . update Un lower-nodes edges Un -> N
. . . . . update Ln upper-nodes edges N -> Ln
. . . . . remove N
. . . . end if
. . . end if
. . end for
. End for
End function
```

Listing 5.1 – Word Lattice construction algorithm

Now leaving the simple example to come into a more real scale, the construction of the lattice itself Thus the aim of the property lattice is to build a data structure that permits to define common groups of properties and detect low variances among them. From the example above we obtain a unique group of interest composed by the four properties which are common to both concepts.

Now leaving this simple example to consider a more realistic one, the construction of the lattice itself was the first challenge to overcome. Contrary to the lattice of shared terms, where a simple iterative algorithm was enough to obtain acceptable computational time, the number of nodes for a lattice of properties can increase quickly. Since this fact is directly related to the construction algorithm computational time and space, it required more effort.

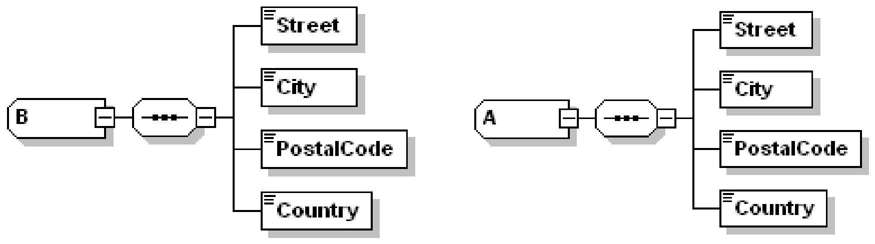


Figure 5.5 – Simple XML schema representation of Address and DeliveryLocation

Indeed, we estimated the size of a complete lattice to be of the order ~2^c nodes, where *c* is the number of concept classes, the number of nodes that we can have at most in a complete lattice with only 20 elements is approximately one million ($2^{20} \approx 10^6$). Considering our use case where we target a domain with potentially several thousands of concepts, the need for a very efficient algorithm is clear.

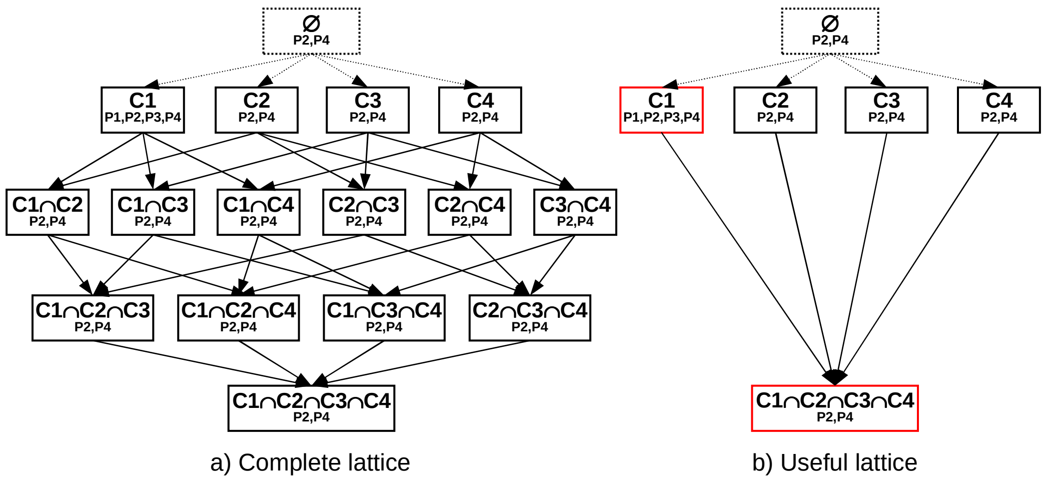


Figure 5.6 – Example of complete lattice and its correspondent useful part

The problem of generating the set of all concepts and the diagram graph of the concept lattice is extensively studied in the literature. Without delving into deep investigation of existing algorithms, we can cite an interesting comparative study of several algorithms constructing the concept set and the graph of the line diagram in [179]. The authors consider, both theoretically and experimentally, several algorithms that generate concept lattices for clearly specified data sets. Among different algorithms presented we found of interest for our purpose the one proposed in [180] , where authors suggest a parallel algorithm to build the lattice. To achieve the parallelisation authors propose to divide the construction of the whole lattice into several sub-lattices and to share them among different processes or machines.

In our java implementation we followed this solution spreading the different defined sub-lattices among few java threads, at least one for each CPU of the machine executing the algorithm.

Besides that we also optimised the number of lattice nodes to create in order to minimize the size of the lattice itself. For this we introduce the notion of **greatest rectangles** that aims to retain only lattice nodes of interest, that we call *useful* nodes of the lattice.

	P1	**P2**	**P3**	**P4**
C1	1	1	1	1
C2	1	1	0	0
C3	1	1	0	0
C4	1	1	0	0

Table 5.1 – Example of greatest rectangles, correspondent matrix of a lattice

As example of greatest rectangles, let us consider the lattices shown in Figure 5.6 composed of four concepts classes (*Cx*) and four properties (*Py*) related as shown in the lattice matrix in Table 5.1. In this matrix the value 1 means that an element *Py* is a property of the corresponding concept class *Cx*. In this figure, we can see the complete lattice (a) for the four concepts set and the *useful* lattice (b) which retains only the nodes with interesting information. This example is intentionally extreme in the sense that of the four concepts, three have exactly the same properties, while the latter has two extra properties. With this kind of input data set, nodes that really contain useful information are:

1. $(C_1)[P_1,P_2,P_3,P_4]$;
2. $(C_2)[P_2,P_4]$
3. $(C_3)[P_2,P_4]$
4. $(C_4)[P_2,P_4]$
5. $(C_1,C_2,C_3,C_4)[P_2,P_4]$;

This is because nodes 1 to 4 maintain the original information about concepts structure while the latter is the maximal intersection of properties with the maximal number of concepts. In other words, in the example shown above among *address* and *delivery_location* it is interesting to know that the four properties are always encountered together. For only two concepts the information itself can be not really relevant, but if the group of properties is repeated several times then it becomes significant. This can lead to the designation of **characteristic properties**, like for example it is reasonable to expect that a concept having as property *first name* will also have *last name*.

What we can observe from this simple example is that what we maximise in the lattice matrix is both, the number of concepts properties and at the same time the number of concepts classes. This corresponds to the research of the greatest rectangles. In Table 5.1, where we represented the matrix, these two rectangles are marked with dotted lines. Listing 5.2 depicts the algorithm implemented for the construction of the property lattice looking for the greatest rectangle.

Some figures on experimental results are presented in Section 5.4.

5.2.3 Building the Similarity Network

Following the construction of the two lattices seen above we generate a graph that combines different concepts' relations to form a unique and complete SDMO instance that we also call the **similarity network**. Figure 5.7 below illustrates a summarized instance of the similarity network. This graph is obtained by the merging of the lattice of words and the lattice of properties with some additional relations among concepts. These relations are in particular synonyms and syntactically close terms. The former can be obtained by querying an external resource, like a thesaurus, while the latter are generated by the application of specific algorithms focusing on the discovery of close terms. These can be algorithms like N-Gram in order to also include abbreviations and misspelled words already discussed in Section 4.3.3 into the SDMO instance.

```
Let be N a lattice node
Let be GPn := GP(N) the property group for N
Let be GCn := GC(GPn) a concept group for GPn
Let be MaxGPGCn := MaxGP(GCn) the properties intersection for GCn

If GPn != MaxGPGCn Then
. Create new node Nn
. Assign MaxGPGCn to Nn
. Assign GCn to Nn
. Create link N -> Nn
Else
. Let be TGPGCn := TGP(GCn) the concepts' properties union for GCn
. Let be ExtGPGCn := ExtGP(Gn) = (TGPGCn - GPn) the GPn complementary group
. For each property extP of ExtGPGCn
. . Create the property group GP := GPn + extP
. . Look for the maximal groupe of concepts GCm such that every element of GCm
has at least GP
. . Look for maximal group of properties GPm for GCm
. . Look for node Nm such that Nm has GPm
. . If Nm does not exist Then
. . . Create Nm
. . . Assign GPm to Nm
. . . Assign GCm to Nm
. . . Add Nm to the lattice
. . . Create link N -> Nm
. . End If
. End For
End If
```

Listing 5.2 – Property Lattice construction overall algorithm

The ambition of this potentially huge graph mixing structural relations with morphological and semantics relations, is to be a practical way to store and maintain true information as concise as possible. This is what we consider to be a memory for the system. In this graph we can find a lot of general correspondences that are founded or at least considered applicable to a certain domain, independently of the specific usage context. Even though at first sight this graph can seem incomprehensible, confusing or complex to a human being, we stress that it has been conceived for a machine use. Nevertheless as we will show in Section 5.4.6 it is possible to transform it into a human friendly form.

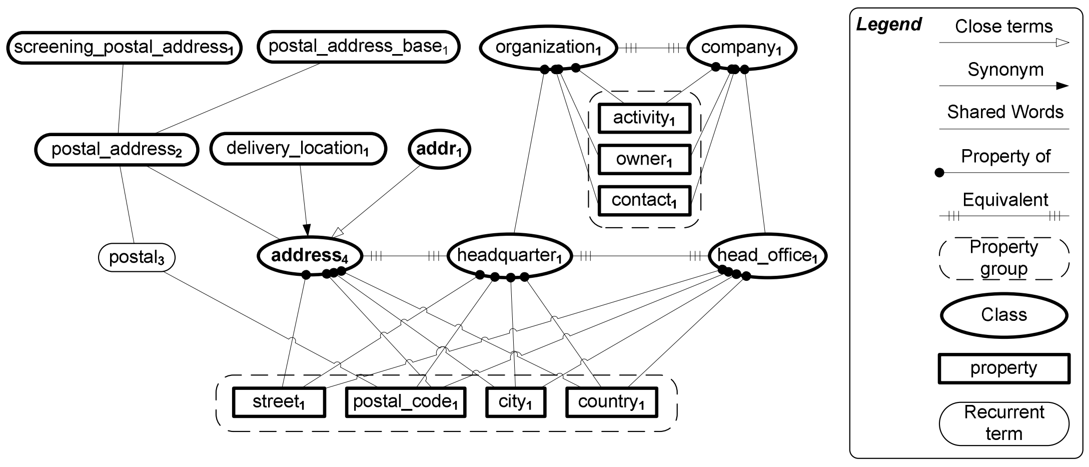


Figure 5.7 –Similarity network representing the graphical view of a SDMO instance

Of course the graph can be enriched with numerous other relations that can be later specifically used by matching algorithms in a contextualised usage. In addition every concept and relation of the graph are also measured in terms of frequency and attendance as explained below.

5.2.4 Frequency Measure

Term Frequency (TF) is one of the major factors in how text mining techniques, search engines and generally information retrieval systems determine relevance. These systems analyze how often keywords appear in relation to other words in a document. Those with a higher frequency are often deemed more relevant than other words in the document itself. However the TF factor alone cannot ensure acceptable retrieval performance. Specifically, when the high frequency terms are not concentrated in a few particular documents, but instead are prevalent in the whole collection, all documents tend to be retrieved, and this affects search precision. To fill in this gap TF measure is often combined with Inverse Document Frequency (IDF) as a means of determining which documents are most relevant to a query. The term discrimination brought by IDF suggests that the best terms for document content identification are those able to distinguish certain individual documents from the remainder of the collection. The combination of both measures gives the TF-IDF weight which is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. A deeper presentation of these three measures can be found in[157].

From our standpoint these classical measures do not completely fill our needs. This is mainly because our corpus is composed of XML Schemas instead of pure text and that what we want find out are just the most common concepts rather then discriminating elements of a document. We want point out two aspects, the frequency of a term within a family[41] (in one's capacity as concept name) and how many families share it. For this, one inconvenient presented by TF-IDF is that it tends to give more importance to low used terms even though they are representative for a document.

A second aspect that we were obliged to follow was the information storage size. Indeed among the different measures TF based we tested, the ones producing better results were a combination of three TF measures. The first one computes the frequency measure of each term for each document for each family. The second calculates the frequency for all weighted terms of a family, using as weight the document frequency values. The last one works out the final value on the global set measuring the frequency of each term taking the family value into account as weight. This kind of measure is relatively precise but forces the complete re calculation every time a source is added. And even though the whole reckoning does not represent a great computational time with respect to all other operations, the storage of all values needed by the complete formulae can become really expensive. We estimated it to be around several mega bytes, just for measuring frequency value. This has been considered disproportionate without real benefits.

Finally we opted for a simpler global measure with term attendance as weighing factor as follows:

$$WeightedTF_j = \frac{occur(t_j)}{\sum occur} * att_j * \frac{1}{\max(TF)};$$

Where *occur(t_j)* is the number of occurrences of the considered term, the denominator is the sum of number of occurrences of all terms and att_j is the attendance for the term *j*. The last element of the formulae just normalizes to 1 as max value.

This kind of measure suggests high values for common terms (read candidate concepts names) with respect to their usage in different standard bodies. It only requires the storage of two integers: the global occurrence and the attendance.

Moreover we observed that a real difference is provided by a measure that considers the nature of a concept. Indeed the final purpose here is to highlight the most representative concepts w.r.t. input sources, and they are normally classes rather than attributes.

5.3 Integration Procedure

So far we have dealt with the transformation of one XML Schema at a time to SDMO and we have seen that our system already improves those solutions met in Sections 1.2 and 4.4.3 thanks to the integration of more XSD constructs and of specific extensible rules. So even though we already improved the most part of other current systems, in practice the real challenge concerns the integration of information extracted from several sources at once. Indeed when concepts are extracted from more sources, the retrieved information often has some heterogeneous design of the same set of concepts. This implies that we might run into conflicting constructs. Regarding this, we have shown in previous

[41] We recall that for family here we mean a logically grouped set of documents, like the set of XML schemas specifications of a sole standard body.

sections the construction of the two lattices that already address a part of the information integration clustering structures and semantics. However it still lacks the harmonization task with the comparison of similar concepts and the presentation of more relevant concepts. Thus we address the implementation of the procedure to discover correspondences and combine similar concepts using SDMO. Finally we detail some implementation features and present some limitation we met during the current implementation. Experimental results will be presented in Section 5.4.

5.3.1 Integrating Multiple XML Schemas

As already mentioned above, our system provides a solution which targets the information extraction from multiple sources natively. For this we do not have the ambition to provide the perfect integration of all different data designs, simply because it probably does not exist. Even for the same domain the design of data can be carried out following different standpoints and there is no absolute way to build data fitting every situation. Thus in our approach we aim to provide an automated process to generate the most probable view of the domain we can find.

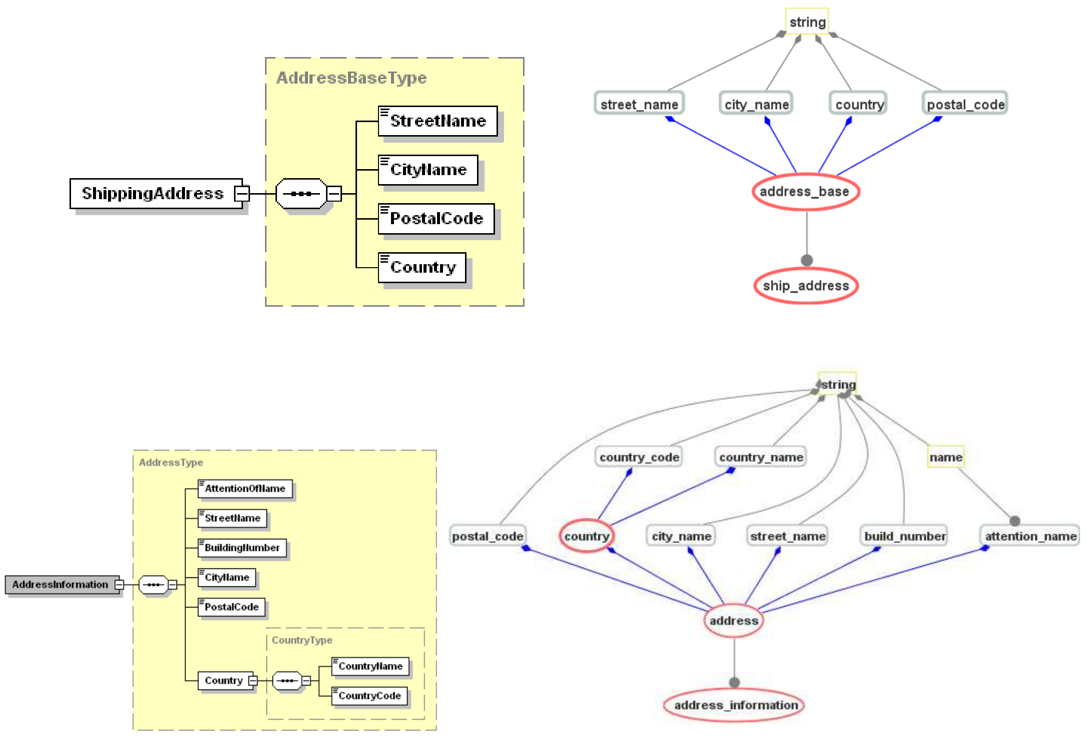


Figure 5.8 – Example XML schemas presenting a simple granularity design difference with their correspondent SDMO graphical representation.

In Section 3.2.2.3 we have already formalised our vision of the fusion of similar concepts having different granularity and larger description with the principle of maximum inclusive. This position is more an adaptation of the CCTS model than a completely new one. The CCTS model defines the more generic components, called *Core Components*, as those concepts fully containing all other specializing components, referred to as *Business Information Entities*. For example in Figure 5.8 we have *shipping_address* and *address_information* concepts classes with similar semantic meaning, in addition they also share relevant structural properties (*country, postal_code, city_name* and *street_name*). As it is perceptible these two concepts refer to the same "upper" concept and thus we would like to provide a unified view of them in the resulting ontology.

Applying the **maximum inclusive principle** formula seen in Section 3.2.2.3 we obtain a satisfactory result to consider the two concepts as equivalent and thus integrated. Integration follows the principle enounced above and thus we maintain the larger definition of the resulting concept as illustrated in Figure 5.9. Relations among the different address concepts simply mean that they have been integrated into the one most representative. All concept properties are maintained and related to the integrated address concept. Rounded properties' group stands for the so called common causality which represents the most characteristic properties for address. Finally country is maintained as class with its own sub-properties.

More precisely in the implementation phase we have introduced a double threshold to the maximum inclusive formula with $0 < a < b < 1$ as suggested by the work done in [181]. In their work authors outline a method of aligning ontologies using the structure matching based on such double level. This is because traditional methods using a single threshold, with a low threshold value give a lot of similarities but some results can be wrong (better recall but lower precision), while a higher threshold gives less results but with fewer errors (better precision but lower recall). So to avoid these kinds of errors he suggested the usage of two thresholds. For values greater than the highest threshold, the similarity is kept and under the lowest, it is refused. Between the two, the suggestion is filtered by a deeper study to validate or invalidate the similarity. We detail our usage of the double threshold in the section below.

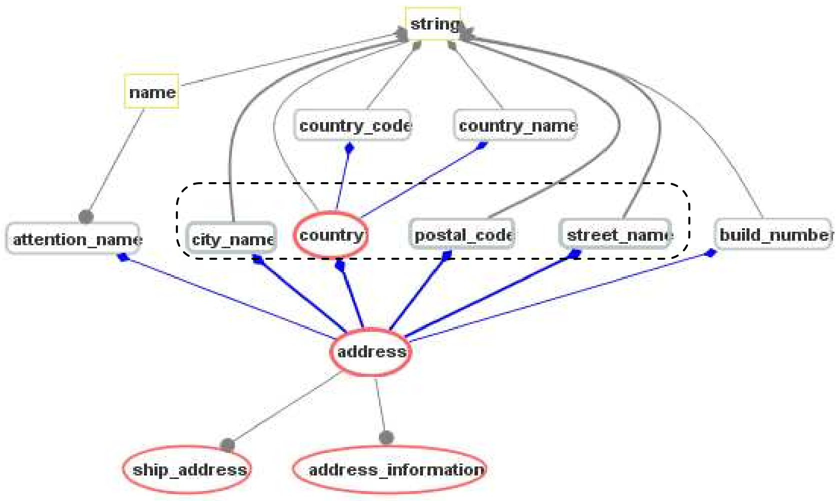


Figure 5.9 – Integration of sources with different granularity, SDMO graphical representation

5.3.2 Combining Concepts Similarities using SDMO

This approach overcomes the various disadvantages of the different techniques aiming for direct integration of input sources. It differs from existing alternatives in its approach to the problem of finding connections between data belonging to different sets. The greater the input corpus is, the more the problem is accentuated. Our approach proposes to reverse the problem by investigating / identifying first the common features of the underlying concepts and then by focusing exclusively on these common elements previously identified to determine the best match between input data sets. The main challenge we try to solve is to collect the larger set of factors to get all the elements suitable for determining the final decision about concepts relations. This information is thus collected and stored into an SDMO instance.

The overall automated process to discover correspondences is characterized by three main steps: a step determining common characteristics of data between inputs provided by the construction of the two lattices seen above; a stage for the generation of the similarity network putting together determined characteristics, also seen above; the final step, which is an iterative deeper comparison of input data sets focusing only on related concepts in the similarity network.

One of the advantages of maintaining such information, the SDMO instance, is that it ensures that algorithms for similarity detection are performed only once. It also helps to ensure that the refinement of the research for correspondences is made only with respect to data that has been previously identified as related. This approach overcomes the matching problem analysed in Section 1.3, where we showed how the matching operation is applied to every pair of input elements. For instance by this way we prevent the execution of matching algorithms over the pair of input concepts like (*person, washing machine*).

The overall algorithm that we implemented to reach this goal is depicted in Listing 5.3.

The algorithm simply queries the similarity network and creates different groups of related concepts depending on the nature of their relations. Depending on the confidence we give the relation we decide to merge directly related concepts or refine the decision. For instance in the presented algorithm only concepts highly related structurally are considered equivalents. In all other cases the procedure uses at least two relations in order to decide if concepts can be considered equivalents. In other words the procedure looks for the intersection over the different groups giving priority to the structural relation, which at least for our use case was the more convincing one.

All non empty obtained intersections are submitted to the so called *merge* function that designates the most important concepts of the set and updates relationships. This function does not prune concepts, but maintains all concepts in the model. It just refines the relations among them and tries to establish the nature of the relation (e.g. saying if a concept is a sub-concept or is the same using the hyperonymy dictionary relation).

The *remove_link* function just invalidates the relation among two concepts of the input set. Here an implementation could decide to maintain the relation in the model and just mark it as not valid rather than remove it. This feature has the advantage that subsequent addition of input sources does not require further recalculation. On the other hand it increases the size of the model.

Experiments validating our procedure are provided below.

```
Let be SN the similarity network
Let be ST the sub-graph of SN maintaining the shared terms relations
Let be C the set of concepts classes
Let be c,d concepts of C
Let be 0 < L < H < 1 two thresholds for the maximum inclusive formula

For each c of C then
  Let be RSMc := RSM(c) the group concepts semantically related to c in SN
. Let be RSYc := RSY(c) the group concepts with syntax relation to c in SN
. Let be RSTGc := RSTG(c) the group concepts with high structural relation to c
   in SN (i.e. structural similarity > H)
. Let be RSTLc := RSTG(c) the group concepts with lower structural relation to c
   in SN (i.e. L < structural similarity < H)

. if RSTGc is not empty then
. . For each c of RSTGc then
. . . Merge all RSTGc concepts
. . End For
. End If

. if RSMc is not empty then
. . For each d of RSMc then
. . . if d also in (RSTLc or RSTGc) then
. . . . Merge(c,d)
. . . else
. . . . Remove_link(c,d)
. . . End if
. . End For
. End If

. if RSYc is not empty then
. . For each d of RSYc then
. . . if d also in (RSTLc or RSTGc) then
. . . . Merge(c,d)
. . . else
. . . . Remove_link(c,d)
. . . End if
. . End For
. End If
End For
```

Listing 5.3 – Similarity Network Refinement, overall algorithm

5.4 Experimental Results

Implementation was constantly present during this work, accompanying our research with continued reification of theoretical aspects and programming issues. We mainly focused our developments on four phases: the information extraction from XSD files, the model generation, the similarity network analysis and the OWL export module. Furthermore all phases permit the integration of new sources incrementally. The result is a software prototype that implements a great part of the automatic generation process and proposes a java graphical interface. One realizes the algorithms for the different phases and glues together the modules, while the other permits their representation for the analysis and a first simple validation.

The most difficult parts to develop were the information extraction, and especially the normalisation steps, the second was the implementation of a scalable similarity network construction algorithm and the alignment was the last one. Main difficulties were that the former must reflect and take care of the different semantic formalisations and design practices. The second uses lattices and graphs that can grow exponentially. The latter problem was that for building a complete and correct similarity network targeting pre-matching information storage, one needs to have at least a creditable alignment. Besides that, the OWL generation and other parts were less complex to program.

Experimental results for information extraction have been already presented and discussed in Chapter 4. Therefore we do not further detail them here and we can state that final quality result are satisfactory with an average precision value higher than 0,95. It has been calculated on the basis of the correctness of the number of XML components to be extracted, their relative structural composition and semantics. Concerning the OWL generation, the choices we made for the modelization and the consequent translation from our semantic model can be theoretically discussed but it is difficult to provide a real quality measure. Moreover some details about the richness of the resulting ontology have been discussed in Chapter 3. Consequently in this section we present our experiments on the conceptualization of input sources, with figures on the generation of the model and its possible adoption. Sub-sections below are outlined as follows, the first sub-section presents the input corpora we used to produce our experiments and provide its dimensions. In Sections 5.4.2 and 5.4.3 we present speed, scalability and storage consideration results. Precision, recall and performance estimations are discussed in sections 5.4.4 and 5.4.5. Finally sub-section 5.4.6 shows main aspects about the resulting graphical interface.

5.4.1 Test Corpora Details

To validate our thesis we have defined four test corpus derived by B2B standards XML Schemas. Each corpus is composed of a set of sub-groups to simulate the incremental addition of XML sources. Moreover the presence of different sub-groups to analyse was useful to validate our hypothesis that it is possible to retrieve similar common information among different sources belonging to a same application domain. This is in opposition with classical matching and merging systems that focus on the mapping of only two sources without considering the amount of information carried by larger corpora. Table 5.2 details the four corpus sources with their dimensions in terms of XML components they have and the number of different entities that we have collected for our tests. The first corpus source is named *Coordinate* and is a simple subset of XML Schemas defining the coordinate entity and its related elements (like latitude, longitude, position, etc.) as they are exchanged in B2B messages. On the same line we have *Address* and *Invoice* groups focussing respectively on the definition of related address and invoice components. The last one that we also have already analysed for its semantics in Chapter 4 is named *Complete B2B* and is the complete set of B2B Schemas we collected.

A problem we encountered was the lack of referent ontology for these sets, in order to provide exact evaluations. Firstly because such sets probably do not even exist. Secondly this is the reason motivating at least in part our work! Therefore we have produced a correct expected reference result just for the firsts two corpuses because they are humanly accessible in term of size and a reference common representation can meet consensus. The last two sets were mainly used for scalability tests and overall observation. Corresponding values are thus approximations estimated with our software, with structural high and low thresholds respectively of 0,9 and 0,3 which performed best in our tests (see Section 5.4.4).

Test corpus name	Groups	Files	Extracted XSD Components	Main entities
Coordinate	7	7	94	83
Address	10	15	463	337
Invoice	9	196	10002	7663
Complete B2B	25	3432	69270	36294

Table 5.2 – Groups adopted for the validation tests details

Table 5.3 provides dimensions in terms of number of concepts and their nature, as defined in Section 3.1.4. In detail the total number of concepts they have and their respective proportion with respect to SDMO definitions of classes, properties, printable types and components with no relevant structural information. The latter are normally generic basic components that are often systematically integrated in XML Schemas even when they are not used for the definition using the *include* or *import* XML construct. These are typically code lists, enumerations or basic XML types definitions.

Corpus	Resulting concepts	Classes	Properties	Printable types	Not structured
Coordinate	26	5	12	9	-
Address	192	33	151	63	-
Invoice	5942*	1291	3613	809	1287
Complete B2B	24703*	6297	17175	5797	898

* Estimated values.

Table 5.3 – Corpora great order with respect to the handled concepts per group

The last set of values presented in Table 5.4 furnishes the dimension of the correspondent SDMO model and relative similarity network. Precisely it provides the number of correspondences among concepts and relations of the different specifications. Lattice of words (WL) and lattice of properties (PL) columns provide the number of useful nodes for each lattice. Finally the last column details the number of synonyms as retrieved from WordNet using the JWNL java implementation.

Corpus	Correspondences	Relations	Classes WL	Properties WL	PL nodes	Synonym
Coordinate	57	86	10	31	13	-
Address	145	655	46	232	63	137
Invoice	1722*	9128*	6120	9165	1726	3182
Complete B2B	11591*	ND	18409	53727	19026	8404

* Estimated values.

Table 5.4 – Concepts dimensions details

Finally we highlight the fact that each corpus is the extension of the previous one, precisely *Complete B2B ⊃ Invoice ⊃ Address ⊃ Coordinate.*

Seeing the complexity of the generation of reference ontologies for these tests we have also conducted human tests using graphical behaviour of our tool. These tests permit us to receive a very good feedback especially when focusing the analysis of corpus that normally is difficult to study manually. Indeed our tool proven its capacity to highlight different sub parts of the whole knowledge easily. This behaviour of our system is presented in Section 5.4.6.

5.4.2 Speed and Scalability Observations

Even when a system achieves good quality results it is vital to study its feasibility in terms of scalability and speed. In this section we detail and argue the required computational time that our system needs to accomplish each phase, the response of the model and the consequent implementation when stressed with large inputs.

Table 5.5 details the computational time required by each task that we obtained using a personal computer equipped of an Intel Core Duo 2GHz as CPU and 2Gb of RAM. Even if the referred time can vary at each execution we observed that a main tendency is respected. In detail the table provides total time to execute the process for the information extraction phase, the construction of the similarity network until the analysis. The sub steps are the extraction from XSD files and the normalization step that also include the query of external resources like dictionary for the first phase. The second phase considers the construction of the Word Lattices (WL) and the Property lattice (PL) detailed above. The last phase includes the computation for the global frequencies determination and merging steps.

From these results we can observe that normalization and synonyms research steps require more time with small inputs but that, as it is natural to expect, the lattices' construction can grow with respect to the size of the input. We noticed this phenomenon especially for the construction of the WL and indeed presented results for the complete B2B corpus already including an optimisation of the original construction algorithm. This optimisation makes that a part of the lattice is built directly at run time during the merging step only if required. Besides that, extraction, merging and above all frequency computation are quite negligible in terms of execution time on the whole process.

Unit of measure [msec]	**Information Extraction phase**		**Similarity Network construction/integration**			**Analysis computation**		
Corpus	**Extraction**	**Normalization**	**WL**	**PL**	**Synonyms**	**Freq.**	**Merging**	**Total**
Coordinate	406	656	94	47	1062	0,171	5,486	2312
Address	1251	4546	235	94	6015	0,834	6,444	12219
Invoice	22843	109813	165093	2375	57595	22	406	371797
Complete B2B	97374	423532	591600	146000	138590	96,389	138984	1561125

Table 5.5 – Model generation main steps time sharing

Figure 5.10 clearly shows that the parts requiring more attention and optimization are the normalization and word lattices construction steps. Instead Figure 5.11 shows in terms of great order the growth of the execution time with respect to the growth of the input source size. What we observe is that the growth remains constant and often decreases and even when it increases it is done in the order of *O(nx)* with *n* max equal to ~5 for the synonyms detection.

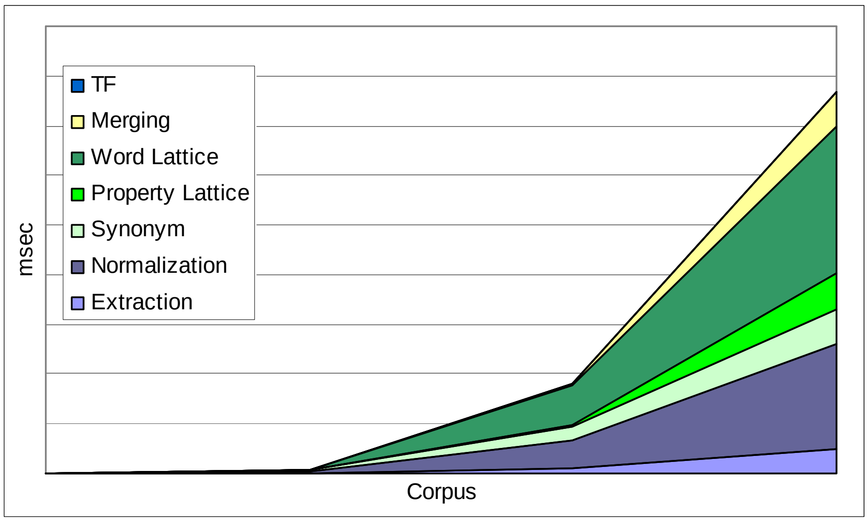


Figure 5.10 – Model generation scalability

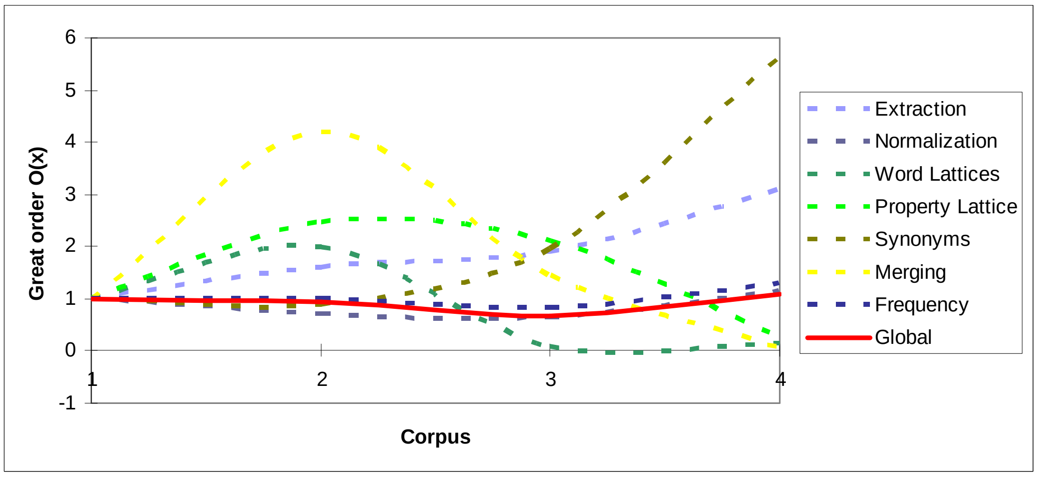


Figure 5.11 – Model generation scalability great order

Table 5.6 and the correspondent Figure 5.12 show the same results of the execution time but this time expressed as percentage. We show these figures because they better express the fact that lattices construction has a different behaviour with the input growth. Indeed whereas the time for normalization and synonyms detection decrease they increase a lot. The growing of the merging is motivated by the optimization we introduced for this test as explained above. The difference is explained by the fact that the incremental sources addition requires at most a correspondent linear augmentation for the synonyms and normalization steps, whereas for the lattices the growth of the number of nodes can be of a greater great order. In several cases the curve also decreases which is a good behaviour to have.

Unit of measure [msec]	**Information Extraction phase**		**Similarity Network construction/integration**			**Analysis computation**	
Corpus	**Extraction**	**Normalization**	**WL**	**PL**	**Synonyms**	**Frequencies**	**Merging**
Coordinate	17,561	28,374	4,066	2,033	45,934	0,007	0,367
Address	10,238	37,204	1,923	0,769	49,227	0,007	0,053
Invoice	6,144	29,536	44,404	0,639	15,491	0,006	0,109
Complete B2B	6,237	27,130	37,896	9,352	8,878	0,006	8,903

Table 5.6 – Model generation main steps percentage sharing

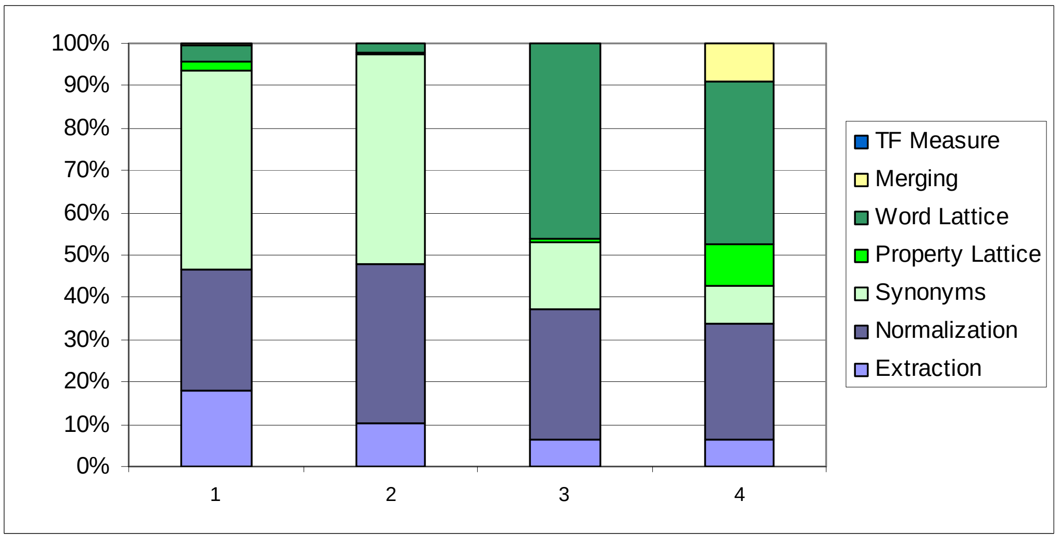


Figure 5.12 – Model generation percentage sharing among the different phases

Just for information the whole process time expressed in minutes corresponds respectively to few seconds for the firsts two groups, around 6 minutes for *Invoice* and more than 20 minutes for the *complete B2B* set. We can claim that with respect to other tested systems implementing matching and merging for only two sets at once our system already provides an adequate response.

5.4.3 Janus Storage Format

SDMO provides a rich organized model that can be stored in several ways. Currently the java implementation that we have developed creates a main java class containing several java hash-tables, in some cases multiple hash-tables, to maintain the model their relations and the lattices. These classes are serialized and stored as files. Table 5.7 presents details on the size of the original XML files in the first column and the relative serialized file. The following columns show the significant gain of space we have that also increase with source size and the gain in terms of execution time.

The Janus file currently permanently stores only the extracted concepts but not yet the whole model. This is because to maintain the whole lattices can require more relevant space. But the operation is viable seeing that the model is constructed incrementally and no further information is required to merge new sources. This becomes clearly interesting if we target real time matching as use case because as shown above the computational time for merging is very low with respect to the whole process.

Corpus	Size [Kb]	Janus file [Kb]	Ratio XSD/JUS [%]	Generation time [msec]	Gain [%]
Coordinate	28	6,19	77,892	125	94,593
Address	493	40	91,886	469	96,161
Invoice	6942,72	606	91,271	142969	61,546
Complete B2B	243712	15926	93,465	1121109	28,185

Table 5.7 –Physical space and computation time gains with the Janus storage format

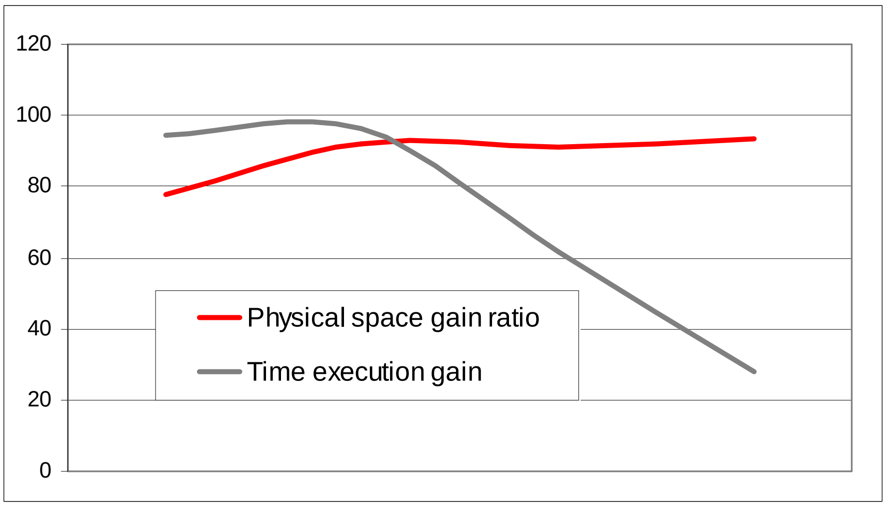


Figure 5.13 – Ratio observation for physical space and process time execution

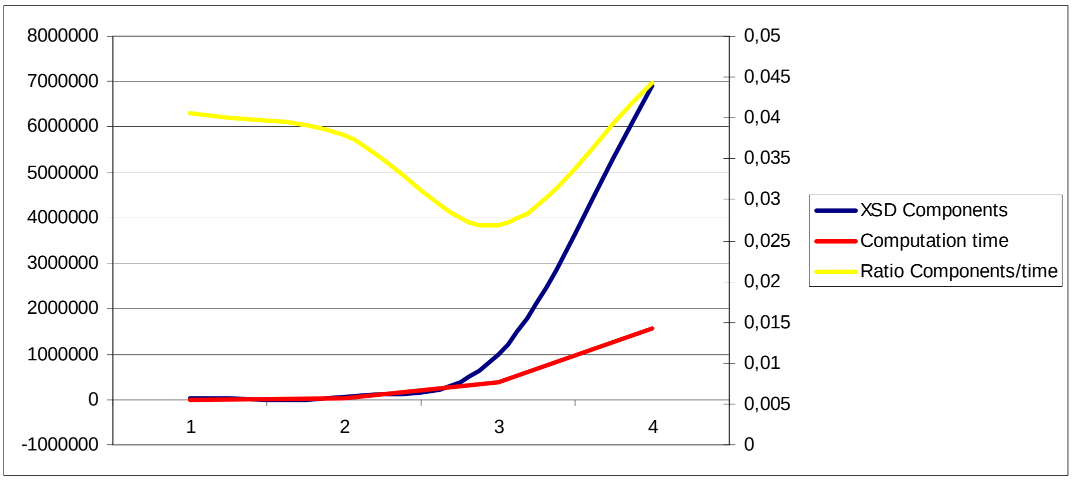


Figure 5.14 – Ratio of the corpora dimension and relative computational time

Figure 5.13 and Figure 5.14 show the trend of the execution time and the physical space gain with the integration of the storage format. As we can see the curve of the time required to reload the whole system still provides interesting gain of time but it decreases. This fact confirms the observation done previously that with larger inputs the system could take large benefits from the introduction of a storage format that already includes the whole model. This is of course to the detriment of physical space, but seeing results it can be justified.

We can say that the solution for this system is fast and permits to obtain relevant gains. Nevertheless we have already seen its limits by testing it with the *Complete B2B* collection. The problems are the following i) the java serialization can fail; ii) loading the whole model's instance directly in memory could be not viable if we want to maintain huge quantity of information (like the web environment could provide).

For these two reasons we are considering storing the model as OWL-full format either in an XML database, or using the relational database correspondence provided by the OWL Protégé API. In this case the active memory will maintain a set of hash-maps corresponding to the concepts and one for each kind of relationships, which prevents memory overload when building graphical views of the whole model.

5.4.4 Quality Measures

In this section we provide a quality measure using the widely known *precision* and *recall* criteria [13]. Precision and recall are based on the comparison of the expected results and the effectively correspondences that are discovered correctly and which are not. So the *precision* measures the ratio of correctly found correspondences over the total number of returned correspondences. In practice it measures the degree of the correctness of the system. The *recall* measures the ratio of correctly found correspondences over the total number of expected correspondences that should be met. Logically it measures the missing correspondences. Both measures are a value comprised between 0 and 1, higher are the values better is the result.

The problem we met with these measures was that they requires a reference set of exact correspondences that, as already mentioned above, it was difficult to provide for our use case. Moreover we stress out that the purpose of our thesis is not the fact to be able to obtain the perfect matching and merging of input sources but rather to be able to maintain and highlight more relevant concepts and their possible relations. Nevertheless we defined a reference set of correspondences that reflects at least in number of concepts the desired outcome, and we have done that for the two corpus *Coordinate* and *Address*.

Finally we executed several tests calculated directly on the obtained SDMO model instance rather than on the derived ontology. This is mainly because at least for this test the correctness of our model should be reflected in the correctness of the final derived ontology. Thus tests have been run over the model generated using the correspondences detection procedure defined in Section 5.3.2 and a partial implementation that emphasize structural correspondences of the algorithm defined in Listing 5.3 using different thresholds values.

Table 5.8 and Table 5.9 provide the results we obtained respectively with a fixed high threshold to 0.8 and 0.9 and a varying low threshold between 0 and 0.5.

These results are provided only for the *Address* corpus because the *Coordinate* corpus does not represent a real challenging set and all measures performed very well. Practically it was very useful for the algorithm implementation phase but not for measuring the feedback.

Address concepts: 192 - Correct correspondences to provide 145						
High threshold	0.8	0.8	0.8	0.8	0.8	0.8
Low threshold	0.5	0.4	0.3	0.2	0.1	0.0
Resulting Concepts	222	214	212	211	211	207
Mergings done	115	123	125	126	126	130
Correct	115	122	122	123	123	125
Precision	1	0,992	0,976	0,976	0,976	0,962
Missing	30	23	21	20	20	20
Recall	0,793	0,841	0,841	0,848	0,848	0,862

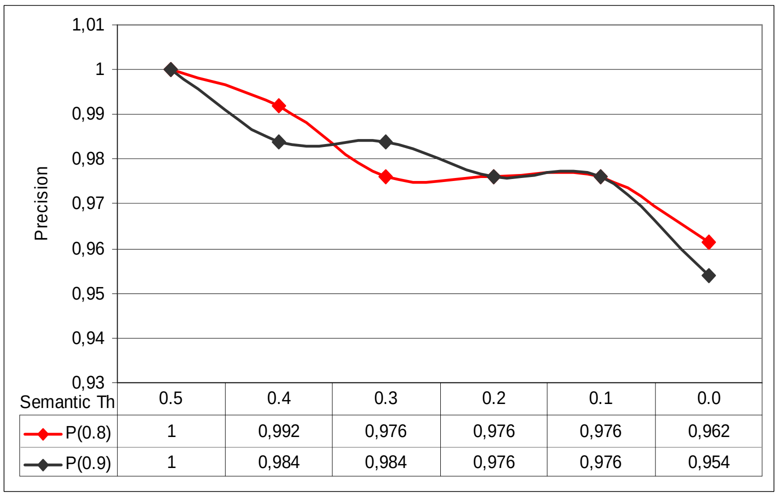
Table 5.8 – Precision and recall measures with fixed high threshold to 0.8

Address concepts: 192 - Correct correspondences to provide 145						
High threshold	0.9	0.9	0.9	0.9	0.9	0.9
Low threshold	0.5	0.4	0.3	0.2	0.1	0.0
Resulting Concepts	222	214	212	211	211	207
Mergings done	115	123	125	126	126	130
Correct	115	121	123	123	123	124
Precision	1	0,984	0,984	0,976	0,976	0,954
Missing	30	24	22	22	22	19
Recall	0,793	0,834	0,848	0,848	0,848	0,855

Table 5.9 – Precision and recall measures with fixed high threshold to 0.9

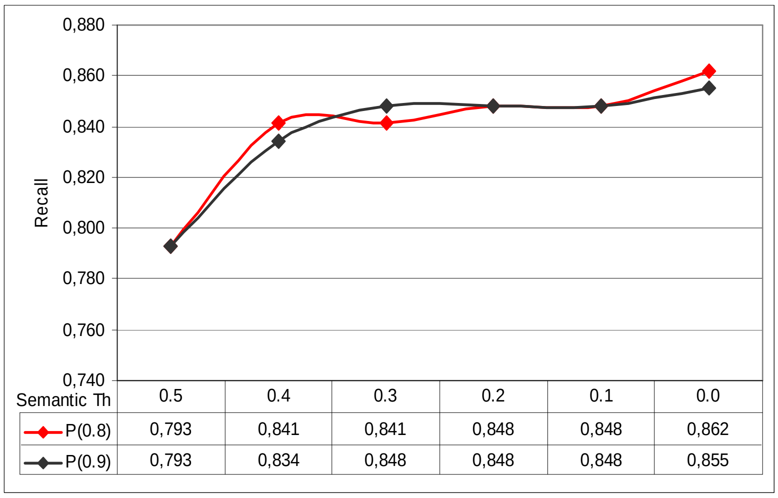
Figure 5.15 and Figure 5.16 illustrate graphically the obtained results. As we can see we have very good results for precision but as we expected from our current implementation results on recall are lower. The best couple of thresholds we got from these tests is (0.9, 0.3) with 0.984 as precision and 0.848 for the recall.

The low recall is motivated by the fact that for the moment our implementation does not integrate advanced matching algorithms combining structures of concepts and matching of different semantics on their attributes. Just to provide an example the current structural matching among two concepts is done over the number of matched concepts properties, which means that if the attribute providing the postal code information is spelled as *postal_code* for the first one and as *postcode_code* for the second concept our algorithm will not consider it similar. This is typically a matching that the most part of currently available matching tools are capable of finding out. However, they are not able to determine the best choice among *postcode_code* and *postal_code* automatically. Indeed using our model, we can claim that the best choice as property for the concept *address* is the second one because it has a frequency value equal to 0.253 while the first one is only 0.009. Moreover we observed that even if a correspondence has not been finally achieved it does not mean that in our model two concepts are not linked. Almost all concepts that should be merged had at least one relation. This is another advantage that our approach proposes. It is the capacity to provide a small set of most probable similar concepts with a very fast simple query to the model on which we could perform more specific matching algorithms before to obtain the final decision. Currently, very few systems are capable of providing this feature.



Semantic Th	0.5	0.4	0.3	0.2	0.1	0.0
P(0.8)	1	0,992	0,976	0,976	0,976	0,962
P(0.9)	1	0,984	0,984	0,976	0,976	0,954

Figure 5.15 – Precision for tests with Address corpus



Semantic Th	0.5	0.4	0.3	0.2	0.1	0.0
P(0.8)	0,793	0,841	0,841	0,848	0,848	0,862
P(0.9)	0,793	0,834	0,848	0,848	0,848	0,855

Figure 5.16 – Recall for tests with Address corpus

5.4.5 Performance Measures

In this section we provide a measure that we performed in the middle of our research and that we have not renewed with more recent set and implementation because it just confirms what we have just said above. With this test we simply validate the fact that the adoption of our model can highly improve time performance when looking for best matching among different sources as response to the problem of the *umbrella* and *washing machine* shown in Section 1.3.1.

The test is performed by using the same two basic matching algorithms but each one following a different approach. In the first one we generate the similarity network before to compute the final research of correspondences. In the second we apply a more classical approach that executes the matching algorithm over each pair of input concepts. As we would expect from this test, the first one performs better and bigger is the corpus size higher is the gain. Among different reasons, it can be simply explained by the fact that the first one performs alignments only once, while the second is obliged to execute them every time an identical pair is met. And of course larger the corpus is, the higher the probability to meet the same set of pairs.

Corpus	**N. of groups**	**N. of Files**	**N. of Concepts**	**Matchings discovery With SDMO [msec]**	**Matchings discovery Without SDMO [msec]**	**Gain of time (%)**
Address	8	12	195	328	396	17
Small Invoice	3	55	1183	3373	4217	21
Invoice	8	187	5808	38130	68586	45

Table 5.10 – Matchings performance increase with SDMO adoption

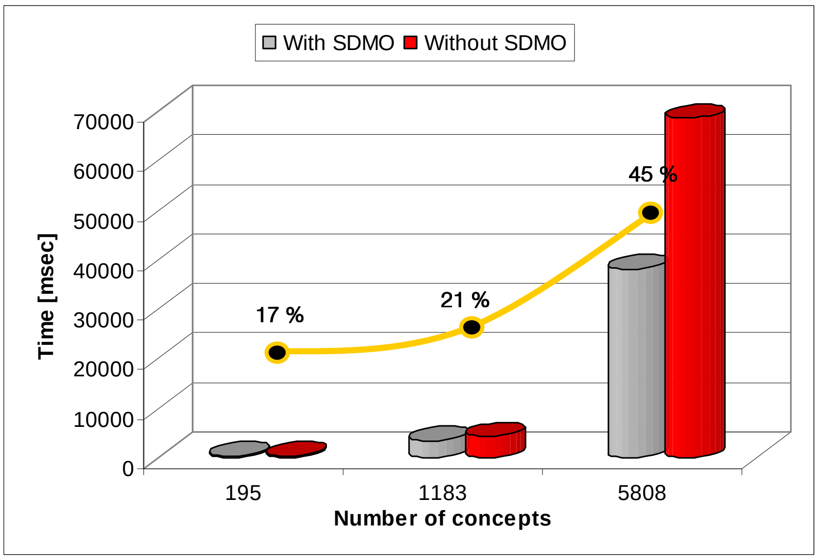


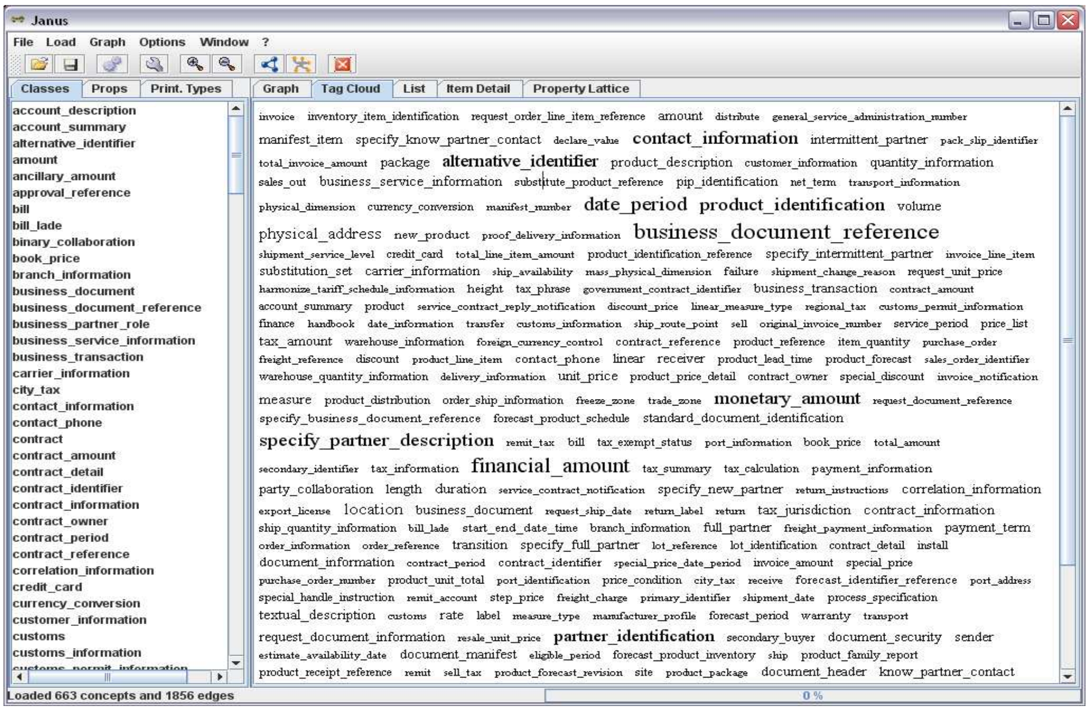
Figure 5.17 – Matchings performance increase, graphical representation

Precision and recall for the two approaches we performed were almost the same, this because the underlying algorithms were almost the same too. Just in the first case we got a lower recall with respect to the second one but as counterpart a better precision.

5.4.6 Functionalities and Views

The tool we have developed currently offers five visualization methods to view the acquired knowledge and a module able to generate a first ontology in OWL format. These are tag cloud, list, detailed, property lattice and graphical views.

The **tag-cloud view** shows the list of concept names adapted to the *tag cloud*[42] format as shown in Figure 5.18. This representation of the model just provides a quick overview of the source inputs highlighting the most representative concepts.


Janus
File Load Graph Options Window ?
Classes Props Print. Types
Graph Tag Cloud List Item Detail Property Lattice

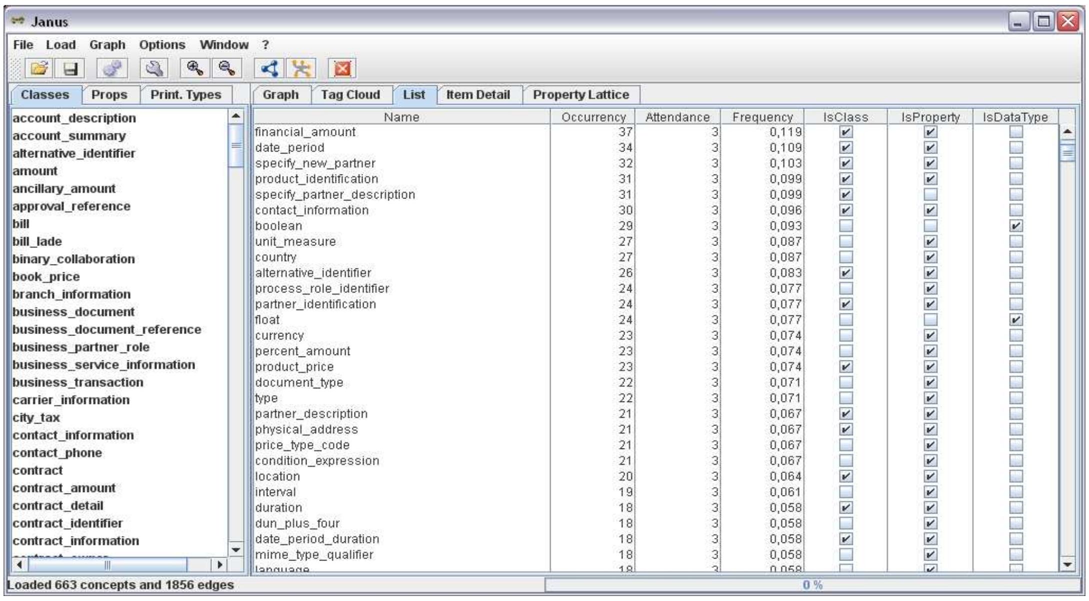
account_description
account_summary
alternative_identifier
amount
ancillary_amount
approval_reference
bill
bill_lade
binary_collaboration
book_price
branch_information
business_document
business_document_reference
business_partner_role
business_service_information
business_transaction
carrier_information
city_tax
contact_information
contact_phone
contract
contract_amount
contract_detail
contract_identifier
contract_information
contract_owner
contract_period
contract_reference
correlation_information
credit_card
currency_conversion
customer_information
customs
customs_information

invoice inventory_item_identification request_order_line_item_reference amount distribute general_service_administration_number
manifest_item specify_know_partner_contact declare_value contact_information intermittent_partner pack_slip_identifier
total_invoice_amount package alternative_identifier product_description customer_information quantity_information
sales_out business_service_information substitute_product_reference pip_identification net_term transport_information
physical_dimension currency_conversion manifest_number date_period product_identification volume
physical_address new_product proof_delivery_information business_document_reference
shipment_service_level credit_card total_line_item_amount product_identification_reference specify_intermittent_partner invoice_line_item
substitution_set carrier_information ship_availability mass_physical_dimension failure shipment_change_reason request_unit_price
harmonize_tariff_schedule_information height tax_phrase government_contract_identifier business_transaction contract_amount
account_summary product service_contract_reply_notification discount_price linear_measure_type regional_tax customs_permit_information
finance handbook date_information transfer customs_information ship_route_point sell original_invoice_number service_period price_list
tax_amount warehouse_information foreign_currency_control contract_reference product_reference item_quantity purchase_order
freight_reference discount product_line_item contact_phone linear receiver product_lead_time product_forecast sales_order_identifier
warehouse_quantity_information delivery_information unit_price product_price_detail contract_owner special_discount invoice_notification
measure product_distribution order_ship_information freeze_zone trade_zone monetary_amount request_document_reference
specify_business_document_reference forecast_product_schedule standard_document_identification
specify_partner_description remit_tax bill tax_exempt_status port_information book_price total_amount
secondary_identifier tax_information financial_amount tax_summary tax_calculation payment_information
party_collaboration length duration service_contract_notification specify_new_partner return_instructions correlation_information
export_license location business_document request_ship_date return_label return tax_jurisdiction contract_information
ship_quantity_information bill_lade start_end_date_time branch_information full_partner freight_payment_information payment_term
order_information order_reference transition specify_full_partner lot_reference lot_identification contract_detail install
document_information contract_period contract_identifier special_price_date_period invoice_amount special_price
purchase_order_number product_unit_total port_identification price_condition city_tax receive forecast_identifier_reference port_address
special_handle_instruction remit_account step_price freight_charge primary_identifier shipment_date process_specification
textual_description customs rate label measure_type manufacturer_profile forecast_period warranty transport
request_document_information resale_unit_price partner_identification secondary_buyer document_security sender
estimate_availability_date document_manifest eligible_period forecast_product_inventory ship product_family_report
product_receipt_reference remit sell_tax product_forecast_revision site product_package document_header know_partner_contact

Loaded 663 concepts and 1856 edges

0 %

Figure 5.18 – Janus Tag- cloud view


Janus
File Load Graph Options Window ?
Classes Props Print. Types
Graph Tag Cloud List Item Detail Property Lattice

account_description
account_summary
alternative_identifier
amount
ancillary_amount
approval_reference
bill
bill_lade
binary_collaboration
book_price
branch_information
business_document
business_document_reference
business_partner_role
business_service_information
business_transaction
carrier_information
city_tax
contact_information
contact_phone
contract
contract_amount
contract_detail
contract_identifier
contract_information

Name	Occurrency	Attendance	Frequency	IsClass	IsProperty	IsDataType
financial_amount	37	3	0,119	✓	✓	
date_period	34	3	0,109	✓	✓	
specify_new_partner	32	3	0,103	✓	✓	
product_identification	31	3	0,099	✓	✓	
specify_partner_description	31	3	0,099	✓		
contact_information	30	3	0,096	✓	✓	
boolean	29	3	0,093			✓
unit_measure	27	3	0,087		✓	
country	27	3	0,087		✓	
alternative_identifier	26	3	0,083	✓	✓	
process_role_identifier	24	3	0,077		✓	
partner_identification	24	3	0,077	✓	✓	
float	24	3	0,077			✓
currency	23	3	0,074		✓	
percent_amount	23	3	0,074		✓	
product_price	23	3	0,074	✓	✓	
document_type	22	3	0,071		✓	
type	22	3	0,071		✓	
partner_description	21	3	0,067	✓	✓	
physical_address	21	3	0,067	✓	✓	
price_type_code	21	3	0,067		✓	
condition_expression	21	3	0,067		✓	
location	20	3	0,064	✓	✓	
interval	19	3	0,061		✓	
duration	18	3	0,058	✓	✓	
dun_plus_four	18	3	0,058		✓	
date_period_duration	18	3	0,058	✓	✓	
mime_type_qualifier	18	3	0,058		✓	

Loaded 663 concepts and 1856 edges

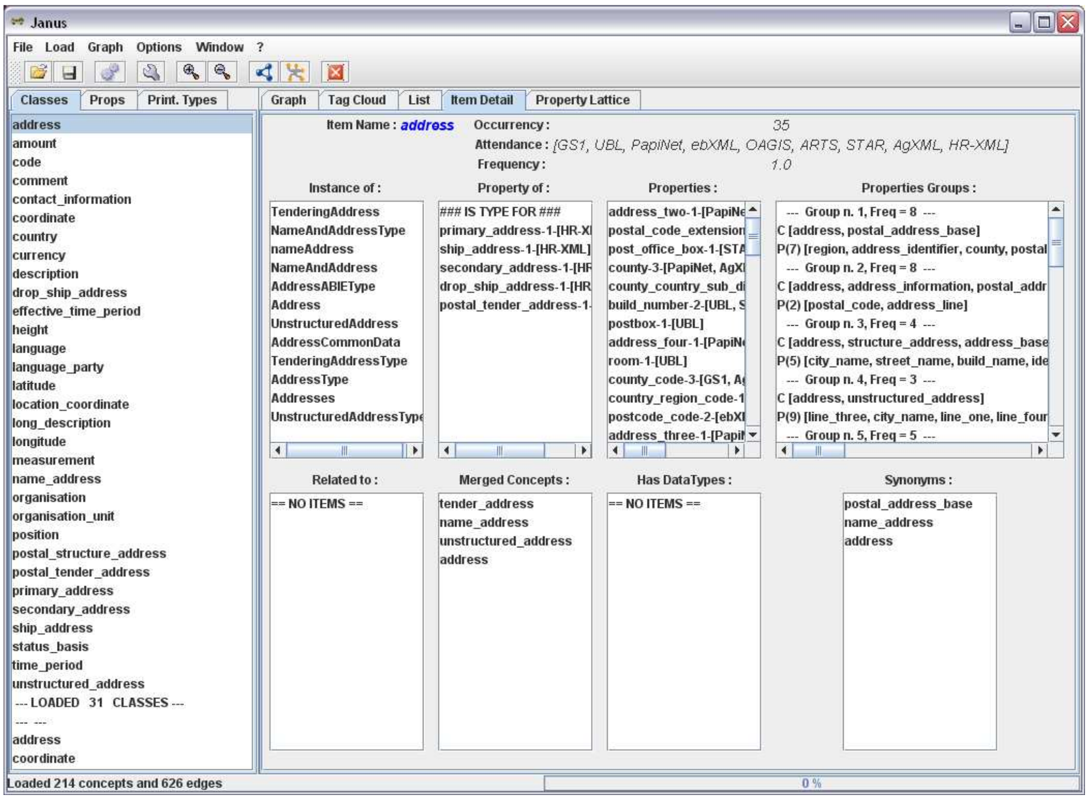
0 %

Figure 5.19 – Janus List Overview

42 A tag cloud or word cloud is normally a visual depiction of user-generated tags

The **list view**, in Figure 5.19, gives detailed information about each concept like frequencies, group attendance and nature (class, printable-type or property). Each value can be used to order the complete list of concepts.

The **detail view** depicted in Figure 5.20, provides all discovered relationships for a specific concept with other concept of the ontology. Between them we can find its properties shared in two lists, one for the properties of the item itself and properties groups, both with their respective frequency. This distinction permits to consider those concepts sharing common properties and the other that we can find for the selected concept.



address
amount
code
comment
contact_information
coordinate
country
currency
description
drop_ship_address
effective_time_period
height
language
language_party
latitude
location_coordinate
long_description
longitude
measurement
name_address
organisation
organisation_unit
position
postal_structure_address
postal_tender_address
primary_address
secondary_address
ship_address
status_basis
time_period
unstructured_address
--- LOADED 31 CLASSES ---
--- ---
address
coordinate

Item Name : *address* Occurrency : *35*
Attendance : *[GS1, UBL, PapiNet, ebXML, OAGIS, ARTS, STAR, AgXML, HR-XML]*
Frequency : *1.0*

Instance of :
TenderingAddress
NameAndAddressType
nameAddress
NameAndAddress
AddressABIEType
Address
UnstructuredAddress
AddressCommonData
TenderingAddressType
AddressType
Addresses
UnstructuredAddressType

Property of :
IS TYPE FOR
primary_address-1-[HR-XI
ship_address-1-[HR-XML]
secondary_address-1-[HR
drop_ship_address-1-[HR
postal_tender_address-1-

Properties :
address_two-1-[PapiNe
postal_code_extension
post_office_box-1-[STA
county-3-[PapiNet, AgXI
county_country_sub_di
build_number-2-[UBL, S
postbox-1-[UBL]
address_four-1-[PapiNe
room-1-[UBL]
county_code-3-[GS1, Ag
country_region_code-1
postcode_code-2-[ebXI
address_three-1-[Papil

Properties Groups :
--- Group n. 1, Freq = 8 ---
C [address, postal_address_base]
P(7) [region, address_identifier, county, postal
--- Group n. 2, Freq = 8 ---
C [address, address_information, postal_addr
P(2) [postal_code, address_line]
--- Group n. 3, Freq = 4 ---
C [address, structure_address, address_base
P(5) [city_name, street_name, build_name, ide
--- Group n. 4, Freq = 3 ---
C [address, unstructured_address]
P(9) [line_three, city_name, line_one, line_four
--- Group n. 5, Freq = 5 ---

Related to :
== NO ITEMS ==

Merged Concepts :
tender_address
name_address
unstructured_address
address

Has DataTypes :
== NO ITEMS ==

Synonyms :
postal_address_base
name_address
address

Loaded 214 concepts and 626 edges
0 %

Figure 5.20 – Janus Detail Overview

The **graph view** displays the semantic network. Figure 5.21 shows a very simple representation of a single XML source defining a wine drinker. Figure 5.23 shows the resulting graph for *address.* As we can see it can be really huge and thus low usable to reveal interesting information to a human observation as is. For this we have implemented different sorts of filters over relations and nature of the concepts that permits a simpler and detailed view of some parts of the model. Indeed the graph view can show the whole graph or only the part related to selected concepts with different layouts (hierarchical, tree, …). For example Figure 5.22 shows the whole graph derived by the complete extraction from the *address* corpus and displays only concept classes. Acquired relationships, and thus that can be visualized, are of different types: *propertyOf*, *synonym*, *shared terms* (i.e. nodes belonging to the WL), *relatedTo*, *isA* and *equivalent classes* (these lasts mainly represent mainly merged concepts or other of type *owl:sameAs* and *owl:equivalentClass*).

The graph view also provides the possibility to choose the relationships to highlight, as well as concepts classes and/or properties. This feature is very useful when the model is too large to be browsed with the global view.

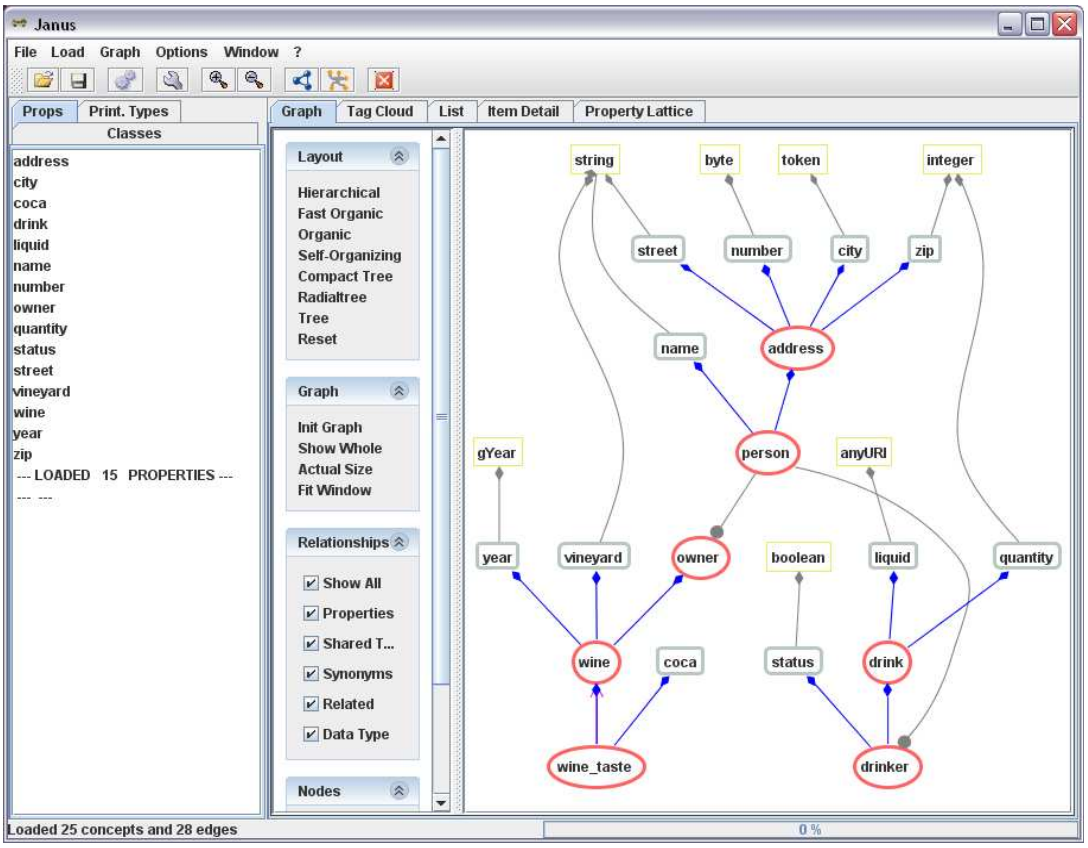


Figure 5.21 – Janus graphical view showing the "wine drinker" single file extraction

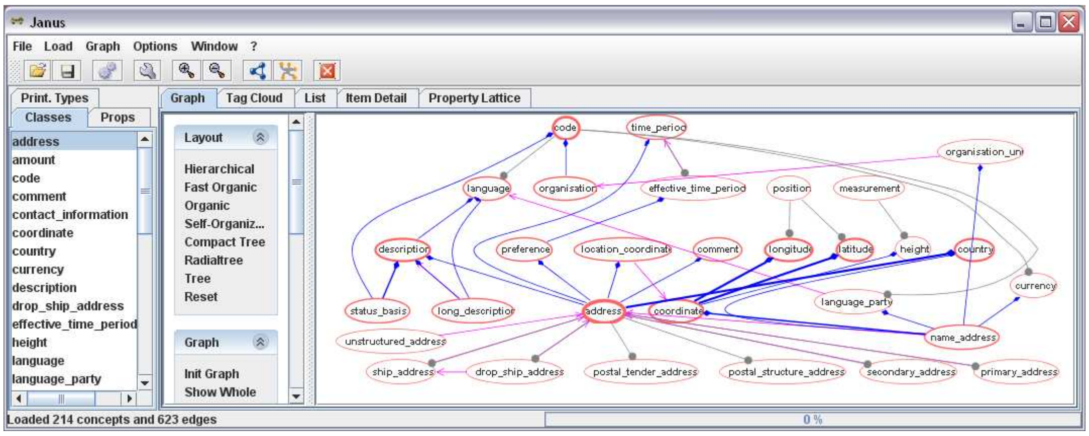
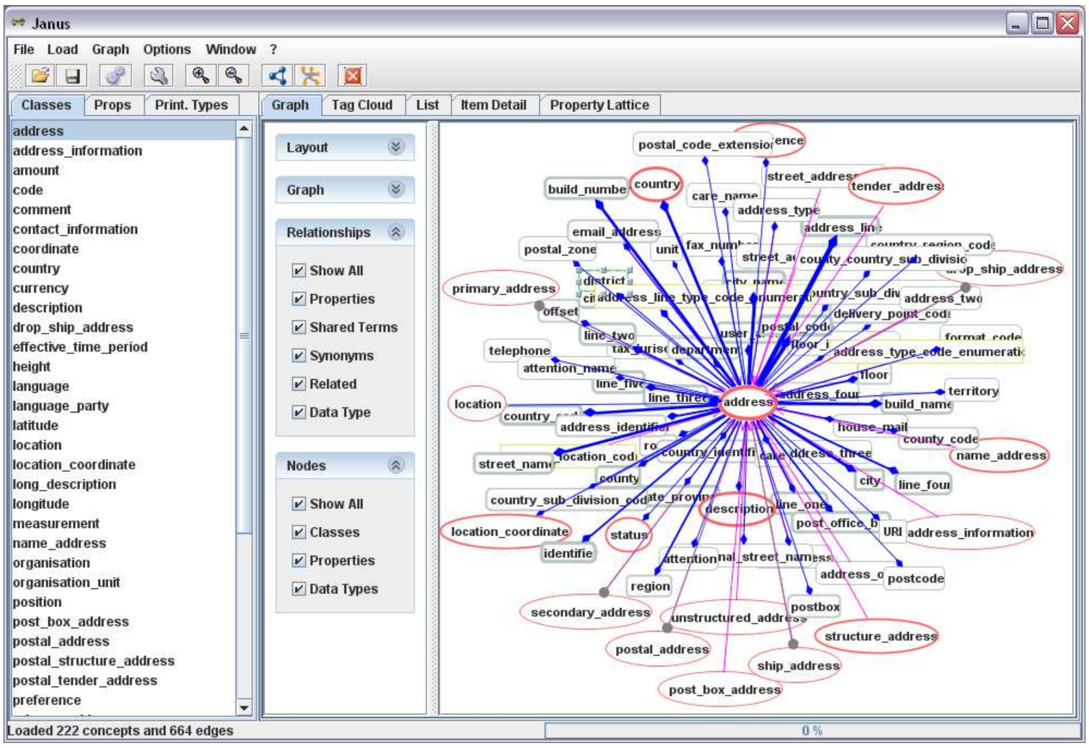


Figure 5.22 – Janus graphical view with only classes view option active



address
address_information
amount
code
comment
contact_information
coordinate
country
currency
description
drop_ship_address
effective_time_period
height
language
language_party
latitude
location
location_coordinate
long_description
longitude
measurement
name_address
organisation
organisation_unit
position
post_box_address
postal_address
postal_structure_address
postal_tender_address
preference

☑ Show All
☑ Properties
☑ Shared Terms
☑ Synonyms
☑ Related
☑ Data Type

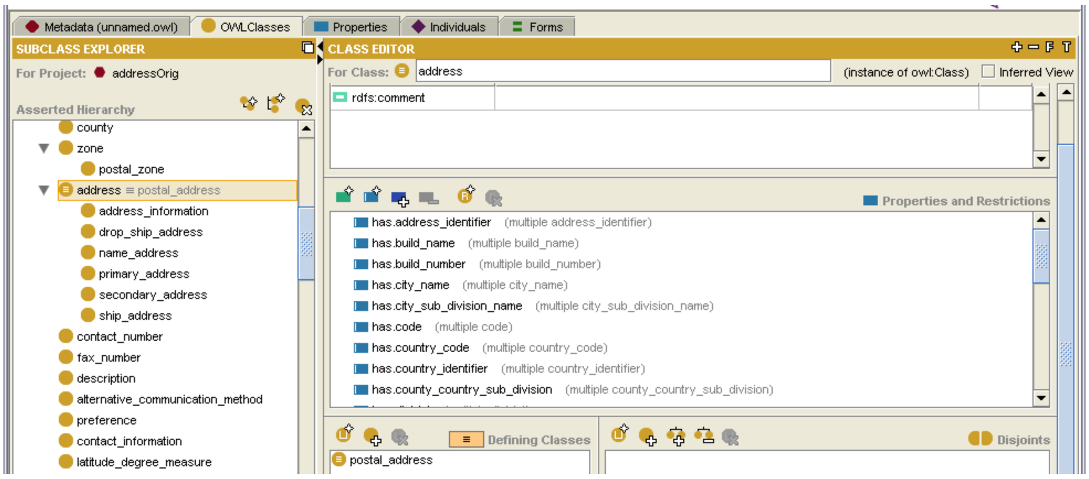
☑ Show All
☑ Classes
☑ Properties
☑ Data Types

Loaded 222 concepts and 664 edges

Figure 5.23 – Janus graphical view focusing the address concept

Other views like “Concepts Social Network” could be implemented. Currently, as always shown in Figure 5.22 and Figure 5.23, we already provide a view that emphasizes elements of the model following their absolute frequency value for both concepts and relations.

Finally the generated ontology can be exported in OWL format. This is an important feature because permits to transform the Janus generated meta-model in a more generic format that can be used by other tools like Protégé [79], as shown in Figure 5.24.



For Project: addressOrig

For Class: address (instance of owl:Class)

rdfs:comment

- county
- zone
  - postal_zone
- address ≡ postal_address
  - address_information
  - drop_ship_address
  - name_address
  - primary_address
  - secondary_address
  - ship_address
- contact_number
- fax_number
- description
- alternative_communication_method
- preference
- contact_information
- latitude_degree_measure

has.address_identifier (multiple address_identifier)
has.build_name (multiple build_name)
has.build_number (multiple build_number)
has.city_name (multiple city_name)
has.city_sub_division_name (multiple city_sub_division_name)
has.code (multiple code)
has.country_code (multiple country_code)
has.country_identifier (multiple country_identifier)
has.county_country_sub_division (multiple county_country_sub_division)

postal_address

Figure 5.24 – Janus Generated OWL View with Protégé

Moreover the graphical interface also offers the possibility to parameterize thresholds for merging operations directly, save the model as Janus file format (as explained in Section 5.4.3) and of course to load directly a saved model.

5.5 Overall Analysis and Conclusion

Throughout this Chapter we have provided a detailed view of the most interesting parts of the implementation of our thesis elements. These are the SDMO semantic model seen in Chapter 3, the information extraction of XML Schemas components seen in Chapter 4 applied to the generation of the model, the overall algorithms for the generation of lattices and Similarity Network (an "alias" that we give to SDMO instances) and the overall algorithm for querying the model.

In detail we have also presented some implementation issues like the multiple merging problem and the integration of sources with different granularity design and we have detailed our approach and solution we adopted. The last part of this Chapter has been dedicated to show a part of experimental results we made to validate our thesis and to present the final outcome represented by the Janus software.

We summarize the work presented in this Chapter with a global satisfactory result.

The implementation phase of our thesis has been more complex than expected in the beginning and this for a lot of more or less little problems we met. Problems generally were not directly linked to our thesis but more of a technical nature. Like the lack of matching API adequate to our scope, the lack of software capable of extracting information from XML schemas rather than text corpus or OWL and last but not least the lack of reference ontologies for our tests and developments. Despite these numerous problems that brought us to the development of a lot of software (finally we can count more than 30.000 lines of java code) necessary to reach a sufficient framework, we have been capable of proving our initial statement.

It has been done by showing that the model we designed to maintain a sort of memory of concepts correspondences is realisable and its implementation is scalable. It can manage large input sources and new sources can be added incrementally. Current problems are more linked to implementation issues and a good compromise between storage and real time requirements can resolve the most part of them. In the first case if we target a system with low physical space requirement we can store only information extracted. Conversely if we target run time applications we can store the whole generated model that provides very fast similarity detection with acceptable precision.

The final evaluation of our system is also supported by the graphical interface we have developed that even if it was out of the initial scope, it has been presented in several occasions at its different level of implementation ([17][18][14]) and a general agreement on its behaviour has been generally manifested. The general subjective satisfaction on the graphical interface was mainly due to the fact that our system does not need any human input except to change very few threshold default parameters, if needed. This behaviour acquired a large consensus because reduces the entry barrier for final users. Indeed they are not supposed to know the meaning of every matching parameter that for some systems count in several dozens. It is quite fast and is only costly in computing resources during the generation of the model calculations. Nevertheless, output correctness is not immediate to test, errors must be discovered, but results of the process are presented in several ways with the possibility to select only little parts to observe in detail, in order to help verification. The graphical representation is very powerful and with a lot of visualizations options and visual measures (like importance of an edge or a concept with respect to others) are available and of simple understanding.

These are the reasons why we believe that our system achieved the initial requirement to be able to extract very useful knowledge from a large set of XML Schemas belonging to a common domain that can be simply translated into an ontology.

Conclusion & Perspectives

In this thesis, we have studied the automatic generation of ontologies derived from XML Schemas. We have done so with the B2B domain in mind. Its ”standardized” but varied and complex use cases still requires research to overcome the "human-bottleneck" generated by the need for managing a large quantity of similar but heterogeneous information. For developing automatic integration tools, we have supported the adoption of Semantic Web technologies. We also investigated the complex B2B architecture to determine the most relevant parts for the integration of such technologies in order to have the highest benefits. There are at least three main topics where this technology can improve the B2B domain limitations: (i) the semantic enterprise content repository, (ii) the automatic mapping among business documents (like messages and business processes), (iii) tools that facilitate the generation of more Web Semantic-oriented business documents.

The first topic is a "completely new" piece of software in the B2B architecture. It targets a repository where enterprises can publish their message semantics and structures. This module could help create a new ecosystem in the domain that permits to share and reuse common B2B documents. Consequently it could lower the barrier to enterprise application integration.

The second topic aims at facilitating the design phase of any B2B collaboration with the automatic matching and integration of business documents. In absolute terms, it could provide a complete automation of the message exchange integration process.

The last topic is motivated by the current lack of real Web Semantic-enabled business documents in the B2B domain. Hence it aims at facilitating the creation of business documents with new and richer contents, thus capitalising on the notable amount of work already produced in more that 20 years of B2B history and experiences.

Finally putting these topics in a priority stack, the third represents the *conditio sine qua non*. Thus, as the title of our thesis says, we have focused on this last topic.

Our thesis aims at improving the capacity to automatically derive conceptual knowledge from XML Schemas. This knowledge can be enriched dynamically by adding new input sources incrementally and it can be used to cover two main issues: i) to improve the automatic generation of ontologies and: ii) to build a memory of discovered correspondences that improves matching performances.

To elaborate our thesis, we began with the evaluation of more relevant works on systems aiming at automatic ontology generation. Throughout the analysis, we have highlighted some limitations of current systems. In particular, current systems usually provide part of the whole ontology generation process only; the generation process is often done over a collection of text documents. Unfortunately, even though the integration of this kind of corpus in our work could improve the information extraction, it is not applicable to our initial statement, which is to derive ontologies from XML Schemas. Additionally, when looking at the rare systems accepting this latter format, they mostly handle few input sources at once and are rarely capable of handling larger corpuses. Concerning the approach, we have observed that systems adopting a framework approach with the integration of an intermediary semantic model perform the automation of the ontology generation better. This comforted us in our approach to build a modular framework-based system as defined by our ontology life-cycle process. Furthermore, all over the analysis, we have shown that, even though few ontology generation systems start from XML sources, the extraction of ontological knowledge from XML sources is viable.

Throughout this dissertation we have proved that our initial statement is true by: i) showing that XML Schemas well fit the required semantics and structural knowledge to build an ontology (refer to Chapter 4); ii) showing that we are able to automatically extract conceptual knowledge from such sources and build a semantic data model that maintains most relevant information coming from a large corpus (refer to Chapter 3 and 4); iii) providing an implementation that brings together all elements (i.e., extraction, semantic data model and ontology generation framework) allowing us to reach our goals. It uses a specific algorithm that we have defined which intelligently queries our model to mix different correspondences so as to obtain the final result (refer to Chapter 5).

Moreover we have shown that our system better fits the need for dynamic ontology generation (i.e. the capacity to add information on the fly) by using our model to store and maintain conceptual information rather than using a final formal ontology.

Below we summarise our work focusing on the main contributions.

Results and Main Contributions

We propose an automatic ontology generation life-cycle as basic approach. The life cycle splits the whole process into five main modules necessary to gather the ontology knowledge and its evolutions. One of the main behaviours of this process is that it aims at building a system capable of integrating new information incrementally. This is a distinctive aspect that provides a real plus with respect to most systems we have seen. This new approach, where new elements can be added at any time, increases the complexity and could imply uncontrollable changes on the final generated ontology. But, as a result, we have a system that is not frozen and that is capable of accepting new knowledge incrementally. This makes our system well adapted to use cases where the input can increase over time.

The choice we made to pass through an intermediary dedicated semantic model, consolidated by our analysis of existing systems, has been successful. It brings a real advantage. An alternative could be to generate several ontologies directly, at least one for each input cluster, and to apply an ontology merging tool. But this approach has the inconvenient that an ontology definition language is not appropriate to maintain fuzzy correspondences, but rather targets a precise representation of concepts with valid relationships. With this kind of representation, it is also not possible to maintain uncertain data on which to perform reasoning. This means that the matching/merging operations must be executed every time a new source is added. In a certain sense, it would be like asking to a search engine to rebuild its index tables each time a new page is created. We observed also that every matching system uses an intermediary place to put the "garbage" indispensable to achieve the final alignment. But it is rarely structured and stored intelligently for further reuse.

The **Semantic Data Model for Ontology** we built is tailored to matching systems. It has been conceived to maintain the information that a matching engine needs. Our model is capable of storing some percepts we naturally have of the real world, including "is a" relationships. It supports the sharing of attributes between classes. It also maintains more specific union type information, like the fact that in computer science a postal code is represented by either a *string* or an *integer*. Finally we provide other information about possible correspondences, like synonymy, similar syntaxes, and frequency measures. This last point provides a real plus to identify more relevant concepts and resolve some ambiguities we can have when merging multiple sources.

The Information Extraction from XML Schemas is another interesting issue we have addressed. As the state of the art highlights, systems that perform such extraction are still underdeveloped and even though existing solutions prove notable results, they often are not available as services that could simply be integrated in applications. Moreover they are often limited to the extraction of only one file at a time. On the contrary, we have designed a system capable of extracting knowledge from a large corpus of XML Schemas. We have initially applied this tool to the B2B corpus collected from standards. The goal was to study the feasibility of extracting semantic and structure information required for ontology generation. First, we have demonstrated throughout several tests that XML Schemas represent a rich source of information to generate first level ontologies. Secondly we have shown that our solution improves existing systems. This task required a great effort but it was necessary to overcome this difficulty for the realization of our thesis. The result has been the generation of a first B2B taxonomy later enriched with more structural knowledge derived from the XML Schemas structures. It can be a good starting point to produce final ontologies.

The automatic generation of first level ontologies was a relatively easy task, thanks to our approach that considers the ontology as a "simple" view of our semantic model. We have provided a complete translation of the model to OWL while respecting the ontology expressivity as much as possible. We propose a consistent set of valid assertions that could be used by a reasoner to produce new subsumptions and other useful deductions. Following the DL naming convention presented in Table 1.1, we have estimated that our ontology corresponds to a *S*HOI*N*Q*F*(D) expressivity where italic elements refer to some limitations (e.g., concept negation and NF are dependent on the integration of cardinality information that has not been implemented yet). This goes beyond the OWL Lite expressiveness.

Janus (which is the nickname of our software) assembles all the pieces mentioned above. It is the final complete system that generates ontologies and that displays results. It also allows user interaction with the process through a graphical interface. This graphical interface permits to oversee the whole process, to visualize partial results, and consequently to modify the parameters default values. Even though we targeted a complete automation, the need for a graphical interface is a must for this kind of work and its existence is really useful.

The software has been demonstrated on several occasions and a common agreement on its behaviour has been generally manifested. Results on this task summarize our work. We have shown that our model, even when stressed with more than 70000 XML components coming from 25 different B2B standards, is able to be built in an acceptable time. This means that even if we are still far from the Web scale, our system is somehow scalable. The gain of time we can have with respect to a common matching approach can reach nearly 50%, and this in the situation where the model is generated at the same time. As we have shown, the model can be stored and reloaded, which means that the time needed to generate the model can be further shortened. Concerning quality results, we have produced some tests that measure the *precision* and *recall* over a subset of our corpus; the best result we obtained is 0.984 as *precision* and 0.848 for the *recall*. Even though these results have been produced with a prototype, they show that the thesis and the approach we chose are satisfactory.

We are aware that a lot of work still remains to be produced to get better results. However, after analysing some results, it appears that the approach with the semantic model, and more generally the Janus system are robust and bring new solutions to the ontology automation problem.

Perspectives and Future Works

We first plan to improve the integration of advanced Web Semantic technologies into the process, like reasoners and other matching systems. So far we have concentrated on developing of modules that were fundamental for our research, like the engine to extract information from XML sources and the semantic model repository. Integrating more advanced semantic web technologies remains to be investigated. For example, the integration of more specific matching and alignment algorithms should improve precision and recall values. These algorithms being often complex, the best way to include them in our system should be to use a specific API like the one offered by the OLA project [182].

Conversely we also plan to provide our implemented modules as APIs in order for them to be widely adopted and integrated into alignment systems in the future. We observed a lack of APIs that can be integrated; hence we have implemented the modules of information extraction and Similarity Network generation as independent components. Thus they can be adapted to become a generic API easily. Few other specific query interfaces should be added to allow such integration.

At the same time we are already using our approach to test dynamic knowledge generation in the SERVERY research project[43]. SERVERY's goal is to enable a Service Market Place that bridges the

[43] Servery is part of the Celtic Telecommunication Solutions EUREKA cluster. Which is an European R&D program in ICT fully dedicated to end-to-end telecommunication solutions. (http://www.celtic-initiative.org/)

Internet and Telco worlds. In this context our contribution aims to provide a more dynamic integration of deployed services in the market place ontology, through a new notion of abstract and concrete services [183] [184]. Thanks to this experience we are already checking some current limitations of our implementation and some specific adaptation to the specific targeted ontology model will be made.

Another aspect that we would like to explore more deeply is to apply Janus to a very large corpus and use it as a generic Web matching engine. A possible direction to follow is a system capable of offering a more adequate response to machines using Web resources. As we have shown in Chapter 1, the integration of the Web as external resource can supply the lack of upper level reference knowledge, but the automatic interpretation of search engines answers is a very complex task. Moreover a machine executes the matching operating on each possible couple of input items and the number of queries to execute, and answer to interpret, can become unsustainable. What we would provide is a web service that receives lists to be matched and returns a formalized answer giving the top-k best matches among the lists. In other words a search engine for machine should use the Web resources to remove false positives reducing the initial corpus to a limited set, rather than answering with huge amount of documents. Semantic engines already are going in this direction, like Watson [185] and Sindice [186], but still provide an answer to a simple keyword. Indeed, even though the answer is generally an RDF/OWL file, thus already machine interpretable, the pruning operation over the returned answer remains human.

Appendix A. Sdmo.owl - OWL Representation for SDMO

A.1 The "Guide" to the SDM 2 OWL Mapping

Once we have defined the basis of our model in Section 3.2.2, we detail here the mapping to an SDMO representation to OWL.

SDM	**OWL**
In SDM we have defined 3 kinds of concepts: classes, properties and datatypes. The OWL representation of SDM class and property entities, is realized with OWL named class, sub class of **`sdm:concept`**. For each SDM property concept we also create 2 object properties: an object property whose name is **`has.`** concatenated to the name of the concept and the inverse property named **`is.`** concatenated to the name of the concept and **`.Of`**. For the "has" property we set its range to the class previously created; for the "is.*class_name*.Of" property, as it is the inverse property, the class will constitute the domain. To represent a SDM datatype, we create an OWL Datatype Property, sub-property of **`sdm:hasDatatype`**.	
`Concepts`	
`Class:` *`SDMClass1`* `- name:` *`classname`*	`Named Class:` *`OWLNClass1`* `- name:` *`classname`* `- sub-class of:` *`sdm:Concept`* `Ex.: <owl:Class rdf:ID="content"> <rdfs:subClassOf rdf:resource="http://janus.orange.org/sdm#Concept"/> </owl:Class>` `Object Property:` *`OWLOProperty1`* `- name:` *`"has."+classname`* `- range:` *`OWLNClass1`* `- sub-property of:` *`sdm:hasProperty`* `- inverse of:` *`OWLOProperty1Inverse`* `Ex.: <owl:ObjectProperty rdf:ID="is.address.Of"> <rdfs:domain rdf:resource="#address"/> <owl:inverseOf> <owl:ObjectProperty rdf :ID="has.address"/> </owl:inverseOf> <rdfs:subPropertyOf rdf:resource="http://janus.orange.org/sdm#isPropertyOf"/> </owl:ObjectProperty>` `Object Property:` *`OWLOProperty1Inverse`* `- name:` *`"is."+classname+".Of"`* `- domain:` *`OWLNClass1`*

	- sub-property of: *sdm:isPropertyOf* - inverse of: *OWLOProperty1*
Property: *SDMProperty1* - name: *propertyname*	Named Class: *OWLNClass2* - name: *propertyname* Object Property: *OWLOProperty2* - name: *"has."+propertyname* - range:*OWLNClass2* - sub-property of: *sdm:hasProperty* - inverse of: *OWLOProperty2Inverse* Object Property: *OWLOProperty2Inverse* - name: *"is."+propertyname+".Of"* - domain: *OWLNClass2* - sub-property of: *sdm:isPropertyOf* - inverse of: *OWLOProperty2*
Datatype: *SDMDatatype1* - name: *datatypename*	Datatype property: *OWLDProperty1* - name: *datatypename* - sub-property of: *sdm:hasDatatype*

Relationships

Let's see how we export semantic relations from SDM to OWL. As seen in the model file (see Section below), we defined an Annotation Object Property named **sdm:synonymOf** that links two Concepts. To export the relation that a Concept "1" is the synonym of a Concept "2", we add the Concept "2" as value of the "synonym of" property for the Concept "1". More precisely, if we want to define that two classes, or two properties, or one class and one property are synonyms, we link the two OWL Classes with the "synonym of" relation and also the two OWL object properties. In the case where we want to define that two datatypes are synonyms, we only have to associate them with the "synonym of" property.

Semantic

Concept1 *synonymOf* Concept2	For the *synonymOf* annotation object property of the *Concept1* we add the value: *Concept2*

A) For 2 classes or 2 Properties or 1 Class and 1 Property:
Concept1: Class or Property
Concept2: Class or Property
	[Concept1 => OWLNClass1 + *OWLOProperty1*]
	[Concept2 => OWLNClass2 + *OWLOProperty2*]
	We set *OWLNClass2* as the value of the *synonymOf* property of OWLNClass1:
	Ex.:
	`<owl:Class rdf:ID="OWLNClass1">`
	  `<sdm:synonymOf>`
	    `<owl:Class rdf:about="#OWLNClass2"/>`
	  `</sdm:synonymOf>`
	`</owl:Class>`
	
	We *OWLOProperty2* as the value of the synonymOf property of *OWLOProperty1*.
	Ex.:
	`<owl:ObjectProperty rdf:ID="OWLOProperty1">`
	  `<sdm:synonymOf>`
	    `<owl:Objectproperty rdf:about="#OWLOProperty2"/>`
	  `</sdm:synonymOf>`
	`</owl:ObjectProperty>`
	
	As *synonymOf* is symmetric, *OWLNClass1* is also value of this annotation property for *OWLNClass2*:Idem for the properties.

B) For 2 Datatypes	[Concept1 => *OWLDProperty1*]
	[Concept2 => *OWLDProperty2*]
	We set *OWLDProperty2* as value of the *synonymOf* property of *OWLDProperty1*.
	Ex.:
	`<owl:DatatypeProperty rdf:ID="OWLDProperty1">`
	  `<sdm:synonymOf>`
	    `<owl:DatatypeProperty rdf:about="#OWLDProperty2"/>`
	  `</sdm:synonymOf>`
	`</owl: DatatypeProperty>`

C) For 1 Datatype and 1 Class or 1 Property
Concept1: Datatype
Concept2: Class or Property
	[Concept1 => *OWLDProperty1*]
	[Concept2 => *OWLOProperty2*]
	We set *OWLOProperty2* as value of the *synonymOf* property of *OWLDProperty1*.
	Ex.:
	`<owl:DatatypeProperty rdf:ID="OWLDProperty1">`
	  `<sdm:synonymOf>`
	    `<owl:ObjectProperty rdf:about="#OWLOProperty2"/>`
	  `</sdm:synonymOf>`
	`</owl:DatatypeProperty>`

To export abbreviations of a given concept (assuming it is a class or a property), we've got two different steps. The first step is the one where we want to export abbreviations for a new concept, having no previous abbreviations. To do that we need to create a new OWL enumerated class whose name will be the name of the concept concatenated to ".Abbreviation". The second step is the creation of an instance of Abbreviation. We take the class representing the abbreviations of the concept and create a new instance of it. The name of the instance is formed by: "the name of the concept" concatenated to ".Abbreviation." and the abbreviation. For example, if we want to define the abbreviation "addr" for the concept address, we will create an instance named **`addr`** of the class **`address.Abbreviation`**.	
`Syntax`	
`Abbreviation`	

`1) For a new concept`	`[Concept1 => `*`OWLNClass1`*`] (`*`concept1name`*`)`
	`Enumerated Class: `*`OWLEClass1Abrv`*
	`- name : `*`concept1name+".Abbreviation"`*
	`- subclass of `*`sdm:Abbreviations`*
	`We link `*`OWLNClass1`*` and `*`OWLEClass1Abrv`*` with `*`sdm:has Abbreviation`*` OWLNClass1 sdm:hasAbbreviations OWLEClass1Abrv OWLEClass1Abrv`
	`- equivalentClass: + `*`OWLIndividual1`*

`2) Adding an abbreviation to an existing concept `*`Concept1 has`*` Abbreviation `*`Abrv1`*	`Individual: `*`OWLIndividual1`*
	`- name: `*`concept1name+".Abbreviation."+Abrv1`*
	`- instance of `*`OWLEClass1Abrv`*
	`Ex :`
	`<owl:Class rdf:ID="address.Abbreviation">`
	`  <rdfs:subClassOf rdf:resource="http://janus.orange.org/sdm#Abbreviation"/>`
	`  <owl:equivalentClass>`
	`    <owl:Class>`
	`      <owl:oneOf rdf:parseType="Collection">`
	`        <address.Abbreviation rdf:ID="address.Abbreviation.addr"/>`
	`      </owl:oneOf>`
	`    </owl:Class>`
	`  </owl:equivalentClass>`
	`  <sdm:isAbbreviationOf>`
	`    <owl:Class rdf:ID="address"/>`
	`  </sdm:isAbbreviationOf>`
	`</owl:Class>`
	`<owl:Class rdf:about="#address">`
	`  <rdfs:subClassOf rdf:resource="http://janus.orange.org/sdm#Concept"/>`
	`  <sdm:hasAbbreviation rdf:resource="#address.Abbreviation"/>`
	`</owl:Class>`

After the semantic and syntax relations, let's have a look at the mapping of the structural relations. The first relation is the "hasProperty", it defines that a concept is composed of another one (typically, the name of a person is composed of a first name and a last name - or maybe several of each). As we have seen previously, these properties are created when concepts are exported (they are object and datatype properties). However, during the concept export, we only create these properties without setting completely their range and domain; it is exactly what we want to do at this step. More concretely, to export that a name has first name and last name, we have to modify the domain of the **`has.first_name`** and **`has.last_name`**. We add the owl named class *name* to the domain of these properties. But as we have defined inverse properties, we also need to alter the range of those. We add *name* to the range of **`is.first_name.of`** and **`is.last_name.of`**. To export "property of" relations, we proceed in the same way as for "has property", except that "property of" are inverse properties. To export datatype relations, it is a little simpler, as we don't have inverse properties, we only have to specify the domain of the owl datatype property. In structural relations, we do not just have composition relations, we also have hierarchical relations, such as "is A" (hyperonym/hyponym) and "merged with" (equivalence relation). For "is A" relations, we rely on the hierarchical relations supported by owl. To render that an SDM class or an SDM property is a sub class of sub property, we reflect this hierarchy on the OWL model. More precisely, to represent that a class *employee* is a *person*, we add "*person*" as a super-class of "*employee*". We also reflect this relation on their respective properties: "**`has.person`**" is a super-property of **`"has.employee`**". For merged element relations, we use the annotation properties **`owl:equivalentClass`** and **`owl:equivalentProperty`**. To say that a class concept "1" is merged with a class concept "2", we define that concept "1" is an **`owl:equivalentClass`** of concept "2". For object and datatype properties, we use the **`owl:equivalentProperty`** relation.	
`Structural`	

hasProperty Concept1 hasProperty Concept2	[Concept1 => OWLNClass1] (ex: *person*) [Concept2 => OWLNClass2] (ex: *address*) [hasProperty => *OWLOProperty2*] (ex: *has.address*) [isPropertyOf => *OWLOP2Inv*] (ex: *is.address.Of*) We add *person* to the domain of *OWLOProperty2and* to the range of *OWLOP2Inv*: *OWLOProperty2* - domain : + *OWLNClass1 OWLOP2Inv* - range : + *OWLNClass1* Ex.: <owl:ObjectProperty rdf:ID="has.address"> <rdfs:domain rdf:resource="#person"/> <owl:inverseOf rdf:resource="#is.address.of"/> <rdfs:subPropertyOf rdf:resource="http://janus.orange.org/sdm#hasProperty"/> </owl:ObjectProperty> <owl:ObjectProperty rdf:ID="is.address.of"> <rdfs:domain rdf:resource="http://janus.orange.org/sdm#Concept"/> <rdfs:range rdf:resource="#person"/> <owl:inverseOf rdf:resource="#has.address"/> <rdfs:subPropertyOf rdf:resource="http://janus.orange.org/sdm#isPropertyOf"/> </owl:ObjectProperty>
PropertyOf Concept1 isPropertyOf Concept2	[Concept1 => *OWLNClass1*] (ex: *topping*) [Concept2 => *OWLNClass2*] (ex: *pizza*) [isPropertyOf => *OWLOProp2*] (ex: *is.topping.of*) [hasProperty => *OWLOP2Inv*] (ex: *has.topping*) Same process as "hasProperty"
hasDatatype Concept1 hasDatatype Datatype1	[Concept1 => *OWLNClass1*] [Datatype1 => *OWLDProperty1*] We add *OWLNClass1* as domain of *OWLDProperty1* Ex: <owl:Class rdf:ID="Enterprise"> <rdfs:subClassOf rdf:resource="http://janus.orange.org/sdm#Concept"/> </owl:Class> <owl:DatatypeProperty rdf:ID="EmployeeNumber"> <rdfs:domain rdf:resource="#Enterprise"/> </owl:DatatypeProperty>
"is A"	
A) Classes or properties *Concept1* is A *Concept2*	[Concept1 => *OWLNClass1* + *OWLOProperty1*] [Concept2 => *OWLNClass2* + *OWLOProperty2*] Adding *OWLNClass2* as superclass of *OWLNClass1* Ex.: <owl:Class rdf:ID="Person"> <rdfs:subClassOf rdf:resource="http://janus.orange.org/sdm#Concept"/> </owl :Class> <owl:Class rdf:ID="Employee"> <rdfs:subClassOf rdf:resource="#Person"/> </owl:Class> Adding *OWLOProperty2* as superproperty of *OWLOProperty1* Ex.: <owl:ObjectProperty rdf:ID="has.Person"> <owl:inverseOf rdf:resource="#is.Person.Of"/> <rdfs:subPropertyOf rdf:resource="http://janus.orange.org/sdm#hasProperty"/> </owl:ObjectProperty> <owl:ObjectProperty rdf:ID="has.Employee"> <owl:inverseOf rdf:resource="#is.Employee.Of"/> <rdfs:subPropertyOf rdf:resource="#has.Person"/> </owl:ObjectProperty> <owl:ObjectProperty rdf:ID="is.Person.Of"> <owl:inverseOf rdf:resource="#has.Person"/> <rdfs:subPropertyOf rdf:resource="http://janus.orange.org/sdm#isPropertyOf"/> </owl:ObjectProperty> <owl:ObjectProperty rdf:ID="is.Employee.Of"> <owl:inverseOf rdf:resource="#has.Employee"/> <rdfs:subPropertyOf rdf:resource="#is.Person.Of"/> </owl:ObjectProperty>
B) Datatypes	[Concept1 => *OWLDProp1*] [Concept2 => *OWLDProp2*]

	Add *OWLDProp2* as super property of *OWLDProp1*
	Ex :
	`<owl:DatatypeProperty rdf:ID="text">`
	  `<rdfs:subPropertyOf>`
	    `<owl:DatatypeProperty rdf:ID="string"/>`
	  `</rdfs:subPropertyOf>`
	`</owl:DatatypeProperty>`
	`<owl:DatatypeProperty rdf:ID="string">`
	  `<rdfs:range`
	`rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>`
	`</owl:DatatypeProperty>`

Merged elements	

A) Classes or properties *Concept1* merged with *Concept2*	[Concept1 => *OWLNClass1* + *OWLOProp1*]
	[Concept2 => *OWLNClass2* + *OWLOProp2*]
	OWLNClass1 owl:equivalentClass *OWLNClass2*
	OWLOProp1 owl:equivalentProperty *OWLOProp2*
	Ex :
	`<owl:Class rdf:ID="Human Being">`
	  `<owl:equivalentClass rdf:resource="#Person"/>`
	  `<rdfs:subClassOf`
	`rdf:resource="http://janus.orange.org/sdm#Concept"/>`
	`</owl:Class>`
	`<owl:Class rdf:ID="Person">`
	  `<owl:equivalentClass rdf:resource="#Human_Being"/>`
	  `<rdfs:subClassOf`
	`rdf:resource="http://janus.orange.org/sdm#Concept"/>`
	`</owl:Class>`
	`<owl:ObjectProperty rdf:ID="has.Human Being">`
	  `<owl:equivalentProperty>`
	    `<owl:ObjectProperty rdf:about="#has.Person"/>`
	  `</owl:equivalentProperty>`
	`</owl:ObjectProperty>`
	`<owl:ObjectProperty rdf:ID="has.Person">`
	  `<owl:equivalentProperty>`
	    `<owl:ObjectProperty rdf:about="#has.Human Being"/>`
	  `</owl:equivalentProperty>`
	`</owl:ObjectProperty>`

B) Datatypes	[Concept1 => *OWLDProp1*] [Concept2 => *OWLDProp2*] *OWLDProp1* owl:equivalentProperty *OWLDProp2*
Finally, we've got source relations. For "instance of" relations we proceed almost on the same way as for abbreviations. We need to create an enumerated class named *conceptname*.Instance subclass of sdm:Instance and linked to the concept via the sdm:instanceOf annotation property. For each instance, we create an OWL individual of this class.	
Source	
InstanceOf	

A) For a new concept	[Concept1 => *OWLNClass1*]
	Enumerated Class : *OWLEClass1Inst*
	- name : *concept1name+".Instance"*
	- subclass of *sdm :Instance*
	We link *OWLNClass1* and *OWLEClass1Inst* with *sdm:hasInstanceOf*
	OWLNClass1 sdm:hasInstanceOf *OWLEClass1Inst*
	Ex:
	`<owl:Class rdf:ID="address">`
	  `<rdfs:subClassOf`
	`rdf:resource="http://janus.orange.org/sdm#Concept"/>`
	  `<sdm:has Abbreviation`
	`rdf:resource="#address.Abbreviation"/>`
	`</owl:Class>`
	`<owl:Class rdf:ID="address.Abbreviation">`
	  `<owl:equivalentClass>`
	    `<owl:Class>`
	      `<owl:oneOf rdf:parseType="Collection"/>`
	    `</owl:Class>`
	  `</owl:equivalentClass>`
	  `<rdfs:subClassOf`
	`rdf:resource="http://janus.orange.org/sdm#Abbreviation"/>`
	  `<sdm:isAbbreviationOf rdf:resource="#address"/>`
	`</owl:Class>`

B) Adding an instanceOf relation to an existing concept
Concept1 instanceOf *Instance1*
	Individual: *OWLIndividual1*
	- name: *concept1name+".Instance."+Instance1*
	- instance of *OWLEClass1Inst*
	OWLEClass1Inst
	- equivalentClass: + *OWLIndividual1*
	Ex :
	`<owl:Class rdf:ID="address.Abbreviation">`
	`  <owl:equivalentClass>`
	`    <owl:Class>`
	`      <owl:oneOf rdf:parseType="Collection">`
	`        <rdf:Description`
	`rdf:about="#address.Abbreviation.addr"/>`
	`      </owl:oneOf>`
	`    </owl:Class>`
	`  </owl:equivalentClass>`
	`  <rdfs:subClassOf`
	`rdf:resource="http://janus.orange.org/sdm#Abbreviation"/>`
	`  <sdm:isAbbreviationOf rdf:resource="#address"/>`
	`</owl:Class>`
	`<address.Abbreviation`
	`rdf:ID="address.Abbreviation.addr"/>`

Table C. 1 - SDM 2 OWL Mapping

A.2 Sdmo.owl – The Base Meta Model

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:sdm="http://janus.orange.org/sdm#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://janus.orange.org/sdm.owl#"
  xml:base="http://janus.orange.org/sdm.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:about="http://janus.orange.org/sdm#Concept"/>
  <owl:Class rdf:about="http://janus.orange.org/sdm#Instance"/>
  <owl:Class rdf:about="http://janus.orange.org/sdm#Abbreviation"/>
  <owl:ObjectProperty rdf:about="http://janus.orange.org/sdm#semantic"/>
  <owl:ObjectProperty rdf:about="http://janus.orange.org/sdm#structural"/>
  <owl:ObjectProperty rdf:about="http://janus.orange.org/sdm#isPropertyOf">
    <owl:inverseOf>
      <owl:ObjectProperty rdf:about="http://janus.orange.org/sdm#hasProperty"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="http://janus.orange.org/sdm#Concept"/>
    <rdfs:subPropertyOf rdf:resource="http://janus.orange.org/sdm#structural"/>
    <rdfs:range rdf:resource="http://janus.orange.org/sdm#Concept"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://janus.orange.org/sdm#syntax"/>
  <owl:ObjectProperty rdf:about="http://janus.orange.org/sdm#hasProperty">
    <rdfs:range rdf:resource="http://janus.orange.org/sdm#Concept"/>
    <rdfs:domain rdf:resource="http://janus.orange.org/sdm#Concept"/>
    <rdfs:subPropertyOf rdf:resource="http://janus.orange.org/sdm#structural"/>
    <owl:inverseOf rdf:resource="http://janus.orange.org/sdm#isPropertyOf"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://janus.orange.org/sdm#source"/>
  <owl:ObjectProperty rdf:ID="numberOfSources">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AnnotationProperty"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:about="http://janus.orange.org/sdm#hasDatatype">
    <rdfs:subPropertyOf rdf:resource="http://janus.orange.org/sdm#structural"/>
    <rdfs:domain rdf:resource="http://janus.orange.org/sdm#Concept"/>
  </owl:DatatypeProperty>
  <owl:AnnotationProperty rdf:about="http://janus.orange.org/sdm#hasAbbreviation">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:AnnotationProperty>
  <owl:AnnotationProperty rdf:about="http://janus.orange.org/sdm#isAbbreviationOf">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:AnnotationProperty>
  <owl:AnnotationProperty rdf:about="http://janus.orange.org/sdm#hasCommonStem">
```

```
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:AnnotationProperty>
  <owl:AnnotationProperty rdf:about="http://janus.orange.org/sdm#instanceOf">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:AnnotationProperty>
  <owl:AnnotationProperty
rdf:about="http://janus.orange.org/sdm#linguisticSimilarity">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:AnnotationProperty>
  <owl:AnnotationProperty rdf:about="http://janus.orange.org/sdm#trustAttendance">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:AnnotationProperty>
  <owl:AnnotationProperty rdf:about="http://janus.orange.org/sdm#trustCounter">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:AnnotationProperty>
  <owl:AnnotationProperty rdf:about="http://janus.orange.org/sdm#synonymOf">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:AnnotationProperty>
  <owl:AnnotationProperty rdf:about="http://janus.orange.org/sdm#nGramWith">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:AnnotationProperty>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.3.1, Build 430) -->
```

Glossary

ABIE	Aggregate Business Information Entity
ACC	Aggregate Core Component
API	Application Program Interface
ASBIE	Association Business Information Entity
ASCC	Association Core Component
BBIE	Basic Business Information Entity
BC	Business Context
BCC	Basic Core Component
BIE	Business Information Entity
BS	Business Specification
CC	Core Component
CCT	Core Component Type
CCTS	Core Component Technical Specification
DTD	Document Type Definition
EDI	Electronic Data Interchange
ebRIM	ebXML Registry Information Model
ebRS	ebXML Registry Services and Protocols
ERP	Enterprise Ressource Planning
JAXR	Java API for XML Registries
LCM	Life Cycle Manager
OWL	Web Ontology Language
OWL-S	Semantic Markup for Web Services
QM	Query Manager
RDF	Resource Description Framework
SaaS	Software as a service
SOA	Service Oriented Architecture
SME	Small and Medium Enterprise
SUMO	Suggested UpperMerged Ontology
SWIFT	Society for Worldwide Interbank Financial Telecommunication
UDDI	Universal Description Discovery and Integration
UML	Unified Modelling Language
WSMO	Web Service Modeling Ontology
WSML	Web Service Modeling Language
XML	eXtensible Markup Language
XMI	XML Metadata Interchange
XSD	XML Schema Definition

Personal Bibliography

Publications

International Conferences

[1] Mathieu Boussard, Vincent Hiribarren, Jean Pierre Le Rouzic, Stéphanie Fodor, Ivan Bedini, Noel Crespi, Gabor Marton, David Moro, Manuel Macias, Oscar Lorenzo Dueñas, Benjamin Molina. Servery: Web Telco Marketplace. Information and Communication Technologies – Mobile Summit 2009. 10 - 12 June 2009, Santander, Spain.

[2] Ivan Bedini, Benjamin Nguyen and Georges Gardarin. B2B Automatic Taxonomy Construction. In Proceedings 10th International Conference on Enterprise Information Systems. 12 - 16, June 2008 Barcelona, Spain.

International Workshops

[3] Jérôme Le Moulec, Jacques Madelaine and Ivan Bedini. Discovery Services Interconnection. International Workshop on RFID Technology. Mai 2009, Milan, Italy.

International Demos

[4] Ivan Bedini, Benjamin Nguyen and Georges Gardarin. Janus: Automatic Ontology Construction Tool. Demo-Poster Session. 16th International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns. September 2008, Acitrezza, Italy.

[5] Ivan Bedini, Benjamin Nguyen and Georges Gardarin. Janus: Automatic Ontology Builder from XSD files. Developer track at 17th International World Wide Web Conference (WWW2008). Beijing, China, April 21 - 25, 2008

French Conferences and Demos

[6] Ivan Bedini, Georges Gardarin, Benjamin Nguyen. Deriving Ontologies from XML Schema. In Proceedings of the Entrepôts de Données et Analyse en Ligne (EDA), France, June 2008. RNTI, Vol. B-4, 3-17 (Invited Paper).

[7] Ivan Bedini, Fabrice Bourge, Benjamin Nguyen. RepXML: Experimenting an ebXML Registry to Store Semantics and Content of Business Messages. Developer Track at BDA 2006. Lille, France. October 2006.

Submitted

[8] Ivan Bedini, Georges Gardarin, Benjamin Nguyen. Semantic Web and e-business. Submitted Chapter for the book « Electronic Business Interoperability: Concepts, Opportunities and Challenges », IGI Global publisher. December 2009.

[9] Emmanuel Bertin, Ivan Bedini, Nassim Laga, Benoit Cristophe, Benjamin Molina. Selecting the best available service at runtime: the concept of abstract services. Submitted to IEEE transactions on Software engineering journal. November 2009.

Patents

[10] Ivan Bedini, Emmanuel Bertin, Nassim Laga. Dynamic selection of the best web service meeting user requirements process. Patent INPI number: 0954427 - 06/2009 *(Pending)*

[11] Ivan Bedini. Procedure for the automation of data sources matching combining semantic and structural properties. Request for patent INPI number: 08 58363 – 12/2008 *(Pending)*

Standardisation Activities

- UN/CEFACT:
  - Member of ICG (Information Content Group)
  - Contributor of TBG17 (Core Components Library Harmonization)
  - Contributor of TMG CCMA (Core Components Messaging Assembly)
- OASIS:
  - Member of ebXML Registry Technical Committee
  - Observer of UBL, eGov, SOA Technical Commitiees
- EDIFRANCE:
  - Member of HICC (Core Components Harmonization) working group
- W3C:
  - Observer of Semantic Web Interest Group.

Main contributions

[12] Fabrice Bourge, Ivan Bedini. UN/CEFACT Registry Implementation Specification. UN/CEFACT ICG Standard Draft

[13] ebXML Registry Profile for OWL-Lite. OASIS Standard Technical Note (Contributor)

[14] OASIS/ebXML Registry Information Model Specification V3.0. OASIS Standard Specification (Also ISO 15000, part 3 and 4 Standard) (Contributor)

[15] OASIS/ebXML Registry Services and Protocols v3.0. OASIS Standard Specification (Also ISO 15000, part 3 and 4 Standard) (Contributor)

[16] Ivan Bedini, Fabrice Bourge, Francis Berthomieu, Fabrice Jeanne, Sébastien Wafflart. EDIFRANCE RepXML Project Overview. UN/CEFACT ICG Deliverable. 2005.

Bibliography

[1] Hohpe, G., and Woolf, B. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley, October 2003. ISBN13:9780321200686 ISBN10: 0-321-20068-3.

[2] Berners-Lee, T., Hendler, J., and O. Lassila. The Semantic Web. Scientific American, 284(5), pp34-43. 2001.

[3] Motta, E., and Sabou, M. Next Generation Semantic Web Applications. In Proc. of the 1st Asian Semantic Web Conference (ASWC), Beijing, China, 3-7 September, 2006.

[4] Corcho, O., and Gomez-Perez A. Solving integration problems of e-commerce standards and initiatives through ontological mappings. In Proceedings of the Workshop on e-business and Intelligent Web. IJCAI 2001.

[5] Hepp, M. GoodRelations. An Ontology for Describing Products and Services Offers on the Web, Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW2008), Acitrezza, Italy, September 29 - October 3, 2008, Springer LNCS, Vol 5268, pp. 332-347.

[6] Giraldo, G., and Reynaud, C. Construction semi-automatique d'ontologies à partir de DTDs relatives à un même domaine, 13èmes journées francophones d'Ingénierie des Connaissances, Rouen, 28-30 Mai 2002.

[7] Caracciolo, C., Euzenat, J., Hollink, L., Ichise, R., Isaac, A., Malaisé, V., Meilicke, C., Pane, J., Shvaiko, P., Stuckenschmidt, H., Šváb-Zamazal, O., and Svátek, V. Results of the Ontology Alignment Evaluation Initiative 2008. Proceedings of the ISWC workshop on Ontology Matching (OM-2008).

[8] Euzenat, J., Mochol, M., Shvaiko, P., Stuckenschmidt, H., Svab, O., Svatek, V., Van Hage, W., and Yatskevich, M. Results of the ontology alignment evaluation initiative 2006. In Proceedings of the ISWC workshop on Ontology Matching, pages 73–95, 2006.

[9] Euzenat, J., Isaac, A., Meilicke, C., Shvaiko, P., Stuckenschmidt, H., Šváb, O., Svátek, V., van Hage, W., and Yatskevich, M. Results of the Ontology Alignment Evaluation Initiative 2007. Proceedings of the ISWC+ASWC workshop on Ontology Matching (OM-2007)

[10] Aleksovski, Z., Ten-Kate, W., and Van-Harmelen, F. Exploiting the Structure of Background Knowledge Used in Ontology Matching. Ontology Matching 2006

[11] Aleksovski, Z., Klein, M. C. A., Ten-Kate, W., and Van-Harmelen, F. Matching Unstructured Vocabularies Using a Background Ontology. EKAW 2006: 182-197

[12] Giunchiglia, F., Shvaiko, P., and Yatskevich, M. Discovering Missing Background Knowledge in Ontology Matching. ECAI 2006: 382-386

[13] Euzenat, J., and Shvaiko, P. Ontology matching. Springer-Verlag, Heidelberg (DE), 2007.

[14] Bedini, I., Gardarin, G., and Nguyen, B. Deriving Ontologies from XML Schema. In Proceedings 4émes Journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 2008). Invited paper. 5 - 6, June 2008 Toulouse, France

[15] Bedini, I., Nguyen, B., and Gardarin, G. B2B Automatic Taxonomy Construction. In Proceedings 10th International Conference on Enterprise Information Systems (ICEIS 2008). 12 - 16, June 2008 Barcelona, Spain

[16] Bedini, I. Procédé de recherche de correspondances entres différentes sources de données. Request for patent INPI number: 08 58363

[17] Bedini, I., Nguyen, B., and Gardarin, G. Janus: Automatic Ontology Builder from XSD files. Developer track at 17th International World Wide Web Conference (WWW2008). Beijing, China, April 21 - 25, 2008

[18] Bedini, I., Nguyen, B., and Gardarin, G. Janus: Automatic Ontology Construction Tool. Demo-Poster Session. 16th International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns (EKAW 2008). September 2008, Acitrezza, Italy

[19] Booch, G. Jacobson, I., and Rumbaugh, J. OMG Unified Modeling Language Specification, Version 1.3 First Edition: March 2000.

[20] Fallside, D. C., and Walmsley, P. XML Schema Part 0: Primer Second Edition. W3C Recommendation 28 October 2004.

[21] Charlet, J., Bachimont, B., and Troncy, R. Ontologies pour le Web sémantique. In Revue I3, numéro Hors Série «Web sémantique», 2004.

[22] Welty, C. Ontology Research. AI Magazine, 24(3), 2003.

[23] Fensel, D. Ontologies: Silver bullet for knowledge management and electronic commerce. Springer-Verlag, Berlin, 2001.

[24] Gomez-Perez, A., and Benjamins, V. R. Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods. IJCAI-1999, Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends.

[25] Gómez-Pérez, A., Fernández-López, M., Corcho, O. Ontological Engineering. November 2003.

[26] Staab, S., and Studer, R., (eds.). Handbook on Ontologies. International Handbooks on Information Systems, Springer Verlag, 2004.

[27] Gruber, T. Encyclopedia of Database Systems, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag, 2008.

[28] Gruber, T. A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220, 1993.

[29] Borst, W. Construction of Engineering Ontologies for Knowledge Sharing and Reuse: Ph.D. Dissertation, University of Twente. 1997.

[30] Fensel, D. Ontologies: Dynamic Networks of Formally Represented Meaning, 2001.

[31] Guarino, N. Formal Ontology and Information Systems. Proceedings of FOIS’98, Trento, Italy, 6-8 June 1998. Amsterdam, IOS Press, pp. 3-15.

[32] Guarino, N. Understanding, Building, and Using Ontologies: A Commentary to "Using Explicit Ontologies in KBS Development", by van Heijst, Schreiber, and Wielinga. International Journal of Human and Computer Studies (46): 293-310. 1997.

[33] Baader, F., and Nutt W. Basic description logics. In The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2003.

[34] Brickley, D., and Guha, R. V. Rdf vocabulary description language 1.0: Rdf schema. W3C recommendation, World Wide Web Consortium, February 2004.

[35] Brickley, D., and Guha, R. V. Resource description framework (rdf) model and syntax specification. W3c recommendation, World Wide Web Consortium, February 1999.

[36] Patel-Schneider, P. F., Hayes, P., and Horrocks, I. OWL Web ontology language semantics and abstract syntax. W3C recommendation, World Wide Web Consortium, February 2004.

[37] Dean, M., and Shreiber, G. OWL Web Ontology Language Reference. W3C Recommendation 10 February 2004. (http://www.w3.org/TR/owl-ref/)

[38] Anicic, N, Ivezic, N., and Jones, A. Semantic Web Technologies for Enterprise Application Integration. In proceedings of the International Journal ComSIS Vol.2, No.1, June 2005

[39] Sure, Y., and Studer, R. On-To-Knowledge Methodology Final Version Project Deliverable D18, 2002.

[40] Lopez, M. F. Overview of the methodologies for building ontologies. Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden, August 1999.

[41] Vrandecic, D., Pinto, H. S., Sure, Y., and Tempich, C. The DILIGENT Knowledge Processes. Journal of Knowledge Management 9 (5): 85-96. October 2005. ISSN: 1367-3270.

[42] Suárez-Figueroa, M. C., Dellschaft, K., Montiel-Ponsoda, E., Villazon-Terrazas, B., Yufei, Z., Aguado de Cea, G., García, A., Fernández-López, M., Gómez-Pérez, A., Espinoza, M., and Sabou, M. NeOn D5.4.1. NeOn Methodology for Building Contextualized Ontology Networks. NeOn project. http://www.neon-project.org. February 2008.

[43] Mehrnoush, S., and Abdollahzadeh, B. The State of the Art in Ontology Learning: A Framework for Comparison. The Knowledge Engineering Review, Volume 18, Issue 4. December 2003.

[44] Aussenac-Gilles, N., and Maedche, A. OLT 2002, Workshop on Machine Learning and Natural Language Processing for Ontology Engineering, held in conjunction with the ECAI'02 conference, Lyon, France, July 22-23, 2002.

[45] Buitelaar, P., Cimiano, P., Grobelnik, M., and Sintek, M. Ontology Learning from Text Tutorial. ECML/PKDD 2005 Porto, Portugal; 3rd October - 7th October, 2005. In conjunction with the Workshop on Knowledge Discovery and Ontologies (KDO-2005).

[46] Rahm, E., and Bernstein, P.A. A survey of approaches to automatic schema matching. The VLDB Journal 10: 334–350. November 2001.

[47] Euzenat J., Le Bach T., Barrasa J., Bouquet P., De Bo J., Dieng R., Ehrig M., Hauswirth M., Jarrar M., Lara R., Maynard D., Napoli A., Stamou G., Stuckenschmidt H., Shvaiko P., Tessaris S., Van Acker S., and Zaihrayeu, I. State of the Art on Ontology Alignment. Knowledge Web Deliverable #D2.2.3, INRIA, Saint Ismier, 2004.

[48] Castano, S., Ferrar, A., Montanelli, S., Hess, G. N., and Bruno, S. State of the Art on Ontology Coorination and Matching. Deliverable 4.4 Version 1.0 Final, March 2007. BOEMIE Project.

[49] Giunchiglia, F., and Shvaiko, P. Semantic Matching. Knowledge engineering review 18 (2003) 265–280.

[50] Noy, N. F., and Musen, M. A. The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping. International Journal of Human-Computer Studies, 2003.

[51] Noy, N. F., and Musen, M. A. Anchor-PROMPT: Using Non-Local Context for Semantic Matching. In Proceedings of workshop on OIS at IJCAI, 2001

[52] Stumme, G., and Maedche, A. FCA-MERGE: Bottom-Up Merging of Ontologies. In Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI), Seattle, WA, 2001.

[53] Doan, A., Madhavan, J., Domingos, P., and Halevy, A. Learning to Map between Ontologies on the Semantic Web. In Proceedings of the 11th International World Wide Web Conference (WWW 2002), Honolulu, Hawaii, USA (2002) 662–673

[54] Castano, S., Ferrara, A., and Montanelli, S. H-MATCH: an Algorithm for Dynamically Matching Ontologies in Peer-based Systems. In Proceedings SWDB 2003.

[55] Bohring H, and Auer S. Mapping XML to OWL Ontologies, Leipziger Informatik-Tage. 2005: 147-156.

[56] Auer, S., Dietzold, S., and Riechert, T. OntoWiki – A Tool for Social, Semantic Collaboration. 5th International Semantic Web Conference, Nov 5th-9th, Athens, GA, USA. In I. Cruz et al. (Eds.): ISWC 2006, LNCS 4273, pp. 736–749, 2006.

[57] Ferdinand, M., Zirpins, C., and Trastour, D. Lifting XML Schema to OWL. In Proceedings of Web Engineering, 4th International Conference, ICWE 2004, Munich, Germany, July 26-30, 2004, pages 354–358. Springer Heidelberg, 2004.

[58] Gasevic, D., Djuric, D., Devedzic, V., and Damjanovic, V. From UML to ready-to-use OWL ontologies. In Intelligent Systems, 2004. Proceedings. 2nd International IEEE Conference, volume 2, pages 485-490. June 2004.

[59] Miller, G.A. WORDNET: A lexical database for English. Communications of ACM (11), 39-41. 1995.

[60] Lonsdale, D., Ding, Y., Embley, D., and Melby, A. Peppering knowledge sources with SALT: Boosting conceptual content for ontology generation, 2002.

[61] Hu, H., and Liu, D.Y. Learning OWL ontologies from free texts. In Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on, volume 2, pages 1233_1237, 2004.

[62] Kong, H., Hwang, M., and Kim, P. Design of the automatic ontology building system about the specific domain knowledge. In Proceedings of the 8th International Conference Advanced Communication Technology, volume 2, February 2006.

[63] Moldovan, D.I., and Girju, R. Domain-specific knowledge acquisition and classification using wordnet. In Proceedings of the 13th International Florida Artificial Intelligence Research Society Conference, pages 224/228. AAAI Press, 2000.

[64] Agirre, E., Ansa, O., Hovy, E., and Martinez, D. Enriching Very Large Ontologies Using the WWW. In Proc. of the Ontology Learning Workshop, ECAI, Berlin, Germany, 2000.

[65] Cho, M., Kim, H., and Kim, P. A new method for ontology merging based on concept using wordnet. In Advanced Communication Technology, the 8th International Conference, volume 3, pages 1573/1576, February 2006.

[66] Kietz, J., Maedche, A., and Volz, R. A Method for Semi-Automatic Ontology Acquisition from a Corporate Intranet. In Proceedings of EKAW-2000 Workshop Ontologies and Text, Juan-Les-Pins, France, October 2000.

[67] Biebow, B., and Szulman, S. TERMINAE: A linguistics-based tool for the building of a domain ontology. In Proc. of EKAW’99, (1999)

[68] Bourigault, D. Lexter, a natural laguage processing tool for terminology extraction. In Procedings of the 7th EURALEX InternationalCongress, Goteborg, 1996.

[69] Nobécourt J. A method to build formal ontologies from texts. In Workshop on ontologies and text, Juan-Les-Pins, France, 2000.

[70] Khan, L., and Luo, F. Ontology Construction for Information Selection. In Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02), Washington, DC, USA, 2002.

[71] Cimiano, P., Hotho, A., and Staab, S. Learning concept hierarchies from text corpora using formal concept analysis. J. of Artificial Intelligence Research, 24:305–339, 2005.

[72] Bertrand, F., Faucher, C., Lafaye, M.C., Lafaye, J.Y., and Alain Bouju. Génération d'une ontologie à partir d'un modèle métier annoté. In proceedings IDM 06, Jun 2006

[73] Dos Santos Mello, R., Heuser, C.A. A Bottom-Up Approach for Integration of XML Sources. Workshop on Information Integration on the Web 2001: 118-124

[74] Dos Santos Mello, R., Heuser, C.A. A Rule-Based Conversion of a DTD to a Conceptual Schema. ER 2001: 133-148

[75] Dos Santos Mello, R., Castano, S., and Heuser, C.A. A Method for the Unification of XML Schemata. Information & Software Technology 44(4): 241-249 (2002)

[76] Halpin, T. Object-Role Modeling (ORM/NIAM). Handbook on Architectures of Information Systems. Chapter 4. Spring-Verlag Berlin/Heidelberg, 1998.

[77] Batini, C., Ceri, S., and Navathe, S.B. Conceptual Database Design: An Entity-Relationship Approach. The Benjamin/Cummings Publishing Company, 1992.

[78] Missikoff, M., and Taglino, F. Symontox: a web-ontology tool for ebusiness domains. In Web Information Systems Engineering (WISE 2003). Proceedings of the Fourth International Conference on, pages 343/346, 2003.

[79] Noy, N. F., Fergerson, R. W., and Musen, M. A. The knowledge model of Protege-2000: Combining interoperability and flexibility. 2th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France, 2000.

[80] Madche, A., Motik, B., Silva, N., and Volz, R. MAFRA – a mapping framework for distributed ontologies. In Proc. ECAI workshop on Knowledge Transformation for the Semantic web, Lyon (FR), pages 60–68, 2002.

[81] Maedche, A. and Staab, S. The Text-To-Onto Ontology Learning Environment. Software Demonstration at ICCS-2000 - Eight International Conference on Conceptual Structures. August, 14-18, 2000, Darmstadt, Germany.

[82] Bozsak, E., Ehrig, M., Handschuh, S., Hotho, A., Mädche, A., Motik, B., Oberle, D., Schmitz, C., Staab, S., Stojanovic, L., Stojanovic, N., Studer, R., Stumme, G., Sure, Y., Tane, J., Volz, R., and Zacharias, V. KAON - Towards a large scale Semantic Web. In: Proceedings of EC-Web 2002. Aix-en-Provence, France, September 2-6, 2002. LNCS, Springer, 2002.

[83] Maedche, A., and Staab, S. Ontology Learning for the Semantic Web. IEEE Intelligent Systems, 16(2): 72/79, 2001.

[84] Raghunathan, P. Fast semi-automatic generation of ontologies and their exploitation. Department of Computer Science, Technical Report, Kansas State University, 2003

[85] Ehrig M., and Staab S. QOM - Quick Ontology Mapping. In Proceeding of ISWC, 2004, pages 683-697.

[86] Sabou, M., d'Aquin, M., and Motta, E. Using the Semantic Web as Background Knowledge for Ontology Mapping. In Proceedings of the International Workshop on Ontology Matching, collocated with ISWC'06.

[87] Do, H., and Rahm, E. COMA - A System for Flexible Combination of Schema Matching Approaches. In Proceedings of 28th International Conference on Very Large Databases (VLDB 2002), Hong Kong, China (2002)

[88] Thor, A., and Rahm, E. MOMA - A Mapping-based Object Matching System. CIDR 2007: 247-258.

[89] Ehrig, M., and Sure, Y. Ontology Mapping - An Integrated Approach. In Proceedings Of the 1st European Semantic Web Symposium, Heraklion, Greece, Springer Verlag (2004) 76–91

[90] Noy, N. F. Semantic Integration: a Survey of Ontology-based Approaches. SIGMOD Record Special Issue on Semantic Integration, 2004.

[91] Shvaiko, P., Euzenat, J. A Survey of Schema-based Matching Approaches. Journal on Data Semantics (JoDS) (2005).

[92] Gracia, J., Lopez, V., d'Aquin, M., Sabou, M., Motta, E., and Mena, E. Solving Semantic Ambiguity to Improve Semantic Web based Ontology Matching. In Proceedings of The Second International Workshop on Ontology Matching November 11, 2007. Busan, Korea.

[93] Bergamaschi, S., Castano, S., Vincini, M., and Beneventano, D. Semantic Integration of Heterogeneous Information Sources. Journal of Data and Knowledge Engineering, vol. 36, n. 3, 2001.

[94] Madhavan, J., Bernstein, P.A., Domingos, P., and Halevy, A. Representing and reasoning about mappings between domain models. In Proceedings of the 18th National Conference on Artificial Intelligence (AAAI'02), Edmonton, Alberta, Canada, August 2002.

[95] Kalfoglou, Y., and Schorlemmer, M. Information-flow-based ontology mapping. LNCS 2519, pages 1132–1151. Springer, 2002.

[96] Castano, S., De Antonellis V., De Capitani di Vimercati S., and Melchiori M. An XML-based framework for information integration over the web. In Proceedings of International Workshop on Information Integration and Webbased Applications & Services, Yogyakarta, Indonesia, September 2000.

[97] Castano, S., De Antonellis, V., De Capitani di Vimercati S., and Melchiori M. Semi-automated extraction of ontological knowledge from XML datasources. In Proceedings of 13th International Workshop on Publication Date. September 2002.

[98] Hill, N.C., and Ferguson, D.M. Electronic Data Interchange: A Definition and Perspective. EDI Forum: The Journal of Electronic Data Interchange: 5 – 12 (1989).

[99] Kantor, M., and Burrows, J.H. Electronic Data Interchange (EDI). Federal Information Processing Standards Publication 161-2. National Institute of Standards and Technology. 1996 April 29. Available from: http://www.itl.nist.gov/fipspubs/fip161-2.htm

[100] ISO 20022 – UNIFI: UNIversal Financial Industry message scheme – http://www.iso20022.org/

[101] ISO/IEC 14662:2004 – Information technology - Open-edi Reference Model

[102] Bray, T., Paoli, J., and Sperberg-McQueen, C.M. Extensible Markup Language (XML) 1.0. W3C Recommendation 10 February 1998.

[103] Gable, J. Enterprise application integration. Information Management Journal, April 2002.

[104] Wenzel, P. OASIS ebXML Messaging Services Version 3.0: Part 1, Core Features. OASIS Committee Specification 02, 12 July 2007.

[105] Workflow Process Definition Interface. XML Process Definition Language. Document Number WFMC-TC-1025-Oct-10-08-A (Final XPDL 2.1 Specification), 2008.

[106] BPML 1.0 Specification, www.bpmi.org, June 2002

[107] Dubray, J.J. A novel approach for modeling business process definitions. http://www.ebpml.org/ebpml2.2.doc.

[108] Dubray, J.J., St. Amand, S., and Martin, J.M. ebXML Business Process Specification Schema Technical Specification v2.0.4. OASIS Standard, 21 December 2006.

[109] Barreto, C. Web Services Business Process Execution Language Version 2.0 Primer. OASIS Standard. 9 May 2007.

[110] Curbera, F. Business Process Execution Language for Web Services (Version 1.1), May 2003, http://www.ibm.com/developerworks/library/specification/ws-bpel/

[111] Shapiro, R. A Technical Comparison of XPDL, BPML and BPEL4WS. http://xml.coverpages.org/Shapiro-XPDL.pdf, 2002

[112] Staab, S., Van der Aalst, W., Benjamins, V.R., Sheth, A., Miller, J.A., Bussler, C., Maedche, A., Fensel, D., and Gannon, D. Web services: been there, done that?. Intelligent Systems, IEEE. Volume 18, Issue 1, Jan-Feb 2003 Page(s): 72 – 85

[113] Dogac, A., and Kabak, Y. Semantic Representations of the UN/CEFACT CCTS-based Electronic Business Document Artifacts. Draft OASIS Profile, September 24, 2008.

[114] Ganter, B., and Wille, R. Formal Concept Analysis. Mathematical Foundations. Berlin: Springer. 1999.

[115] OASIS ebCPPA Technical Committee. Collaboration-Protocol Profile and Agreement Specification Version 2.0. ISO/TC 15000. ISO 15000-1 international standard. 29 March 2004.

[116] Blantz, M.K., Kulvatunyou, S., Reinprecht, N., Stuhec, G., and Walther, J. UN/CEFACT Core Components Message Assembly Technical Specification. Working Draft Revision 1.8 28 October 2007.

[117] Clement, L., Hately, A., Von Riegen, C., and Rogers, T. OASIS Universal Description, Discovery and Integration (UDDI) Specification. Version 3.0.2. OASIS Standard. February 2005.

[118] Breininger, K., Najmi, F., and Stojanovic, N. OASIS ebXML Registry Information Model (RIM) v3.0. OASIS Standard. May 2005.

[119] Breininger, K., Najmi, F., and Stojanovic, N. OASIS ebXML Registry Services Specification (RS) v3.0. OASIS Standard. May 2005.

[120] Pankraz, A., and Weinhart, J. Semic Repository System Architecture. SEMIC.EU Project deliverable. October 2008.

[121] Bedini, I., and Bourge, F. UN/CEFACT Registry Implementation Specification. UN/CEFACT ICG Standard Draft Specification. 2008.

[122] Bedini, I., Bourge, F., and Nguyen, B. RepXML: Experimenting an ebXML Registry to Store Semantics and Content of Business Messages. Developer Track at BDA 2006. Lille, France. October 2006.

[123] E-Business W@tch observatory, 2007. The European e-Business Report, 2006/07 edition. 5th Synthesis Report of the e-Business W@tch, on behalf of the European Commission's Directorate General for Enterprise and Industry. January 2007. (http://www.ebusiness-watch.org)

[124] eClass e.V. eCl@ss: Standardized Material and Service Classification. 2007. (http://www.eclass-online.com)

[125] United Nations Development Programme. United Nations Standard Products and Services Code (UNSPSC). 2007. (http://www.unspsc.org)

[126] Kabak Y., and Dogac A. A Survey and Analysis of Electronic Business Document Standards Under revision in ACM Computing Surveys. 2008.

[127] Léger, A., ed. OntoWeb: ontology-based information exchange for knowledge management and electronic commerce. OntoWeb D2.2 final. 2002.

[128] Fensel, D., Ding, Y., Omelayenko, B., Schulten, E., Botquin, G., Brown, M., and Flett, A. Product Data Integration in B2B E-Commerce. IEEE Intelligent Systems, vol. 16, 2001, pp. 54-59.

[129] Hepp, M. Possible Ontologies: How Reality Constrains the Development of Relevant Ontologies. IEEE Internet Computing 11(1): 90-96. 2007.

[130] Zhao, Y., and Sandahl, K. Potential Advantages of Semantic Web for Internet Commerce. Proceedings of International Conference on Enterprise Information Systems (ICEIS), Vol 4, pp151-158, Angers, France, April 23-26, 2003

[131] Zhao, Y., and Lövdahl, J. A Reuse-Based Method of Developing the Ontology for E-Procurement. Proc Second Nordic Conference on Web Services (NCWS'2003), ISBN 91-7636-392-9, Växjö, Sweden, Nov 20-21, 2003

[132] Coates, A.B. Semantic data models and business context modeling. Invited speaker at XML2007. Boston, Massachusetts, USA. 3-5 December 2007.

[133] Smith, B. Against Idiosyncrasy in Ontology Development. International Conference on Formal Ontology in Information Systems (FOIS 2006). Baltimore, Maryland (USA), November 9-11, 2006.

[134] Batres, R., West, M., Leal, D., Price, D., and Naka, Y. An Upper Ontology based on ISO 15926. In proceedings of European Symposium on Computer Aided Process Engineering - ESCAPE 15. Barcelona, Spain. June 2005.

[135] Hepp, M. E-Business Vocabularies as a Moving Target: Quantifying the Conceptual Dynamics in Domains. EKAW 2008, LNCS 5268, pp. 388–403, 2008

[136] D'Aquin, M., Haase, P., and Gómez-Pérez, J.M. NeOn - Lifecycle Support for Networked Ontologies: Case studies in the pharmaceutical industry. In proceedings of ESTC'08

[137] Tran, D.C., Haase, P., Lewen, H., Munoz-Garcia, O., Gómez-Pérez, A., and Studer R. Lifecycle-Support in Architectures for Ontology-Based Information Systems. In Proc. of the International Semantic Web Conference, ISWC 2007.

[138] Noy, N. F., and Klein, M. Ontology Evolution: Not the Same as Schema Evolution. Knowledge and Information Systems 6(4), 428–440 (2004)

[139] UN/CEFACT Techniques and Methodologies Group. UN/CEFACT Core Components Technical Specification. Part 8 of the ebXML Framework, ISO\TS 15000-5. Version 2.01, 15 November 2003.

[140] Noy, N.F., and McGuinness, D.L. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.

[141] Niles, I., and Pease, A. Towards a standard upper ontology. In Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS), pages 2–9, 2001.

[142] IEEE SUO Working Group. Standard Upper Ontology Knowledge Interchange Format. IEEE P1600.1 Standard Draft. 2003. Available from: http://suo.ieee.org/SUO/KIF/index.html

[143] Yarimagan, Y., and Dogac, A. A Semantic based Solution for the Interoperability of UBL Schemas. To appear in IEEE Internet Computing Magazine, 2009.

[144] Lara, R., Cantador, I., and Castells, P. XBRL taxonomies and OWL ontologies for investment funds, First International Workshop on Ontologizing Industrial Standards at the 25th International Conference on Conceptual Modeling (ER2006), Tucson, Arizona, USA, November 6-9, 2006

[145] Haller, A., Gontarczyk, J., and Kotinurmi, P. Towards a complete SCM ontology: the case of ontologising RosettaNet. SAC 2008: 1467-1473

[146] Haller, A., Kotinurmi, P., Vitvar, T., and Oren, E. Handling heterogeneity in RosettaNet messages. SAC 2007: 1368-1374

[147] Lausen, H., Polleres, A., and Roman, D. Web Service Modeling Ontology (WSMO). Member submission, W3C, 2005. Available from: http://www.w3.org/Submission/WSMO/.

[148] De Bruijn, J., and Lausen, H. Web Service Modeling Language (WSML). W3C Member Submission 3 June 2005. Available from: http://www.w3.org/Submission/WSML/

[149] Riehle, D., Tilman, M., and Johnson, R. Dynamic Object Model. In Pattern Languages of Program Design 5. Edited by Dragos Manolescu, Markus Völter, James Noble. Reading, MA: Addison-Wesley, 2005.

[150] Hammer, M., and McLeod D. Database Description with SDM: A Semantic Database Model. ACM ‘Transactions on Database Systems, Vol. 6, No. 3, September 1981, Pages 351-386.

[151] Abiteboul, S., and Hull, R. IFO: A Formal Semantic Database Model. ACM Transactions on Database Systems, Vol. 12, No. 4, December 1987, Pages 525-565.

[152] Codd, E. F. A relational model of data for large shared data banks. Commun. ACM 13,6. 1970, 377-387.

[153] Chen, P. P. The entity-relationship model-toward a unified view of data. ACM Trans. Database Syst. 1, 1 (1976), 9-36.

[154] Giunchiglia F., Shvaiko P., and Yatskevich M. S-Match: an algorithm and an implementation of semantic matching. In Proceedings of ESWS 2004, Heraklion (GR), pages 61–75, 2004.

[155] Biron, P.V., and Malhotra, A. XML Schema Part 2: Datatypes Second Edition. W3C Recommendation 28 October 2004.

[156] Klein, M.C.A., Broekstra, J., Fensel, D., Van Harmelen, F., Horrocks, I. Ontologies and Schema Languages on the Web. Spinning the Semantic Web 2003: 95-139.

[157] Salton, G., and Buckley, C. Term-Weighting Approaches in Automatic Text Retrieval. Information Processing and Management Journal. 24(5): 513-523. 1988.

[158] Brin, S., and Page, L. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, 1998.

[159] Jeh, G., and Widom, J. SimRank: a measure of structural-context similarity. In KDD'02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 538-543. ACM Press, 2002.

[160] Van Assem, M., Gangemi, A., and Schreiber, G. RDF/OWL Representation of WordNet. W3C Working Draft 19 June 2006.

[161] Ding, Z. and Peng, Y. A Probabilistic Extension to Ontology Language OWL. In Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 4, IEEE Computer Society, 2004.

[162] Da Costa, P.C.G., Laskey, K.B., and Laskey, K.J. PR-OWL: A Bayesian Ontology Language for the Semantic Web, ISWCURSW, 2005, 23-33.

[163] Pool, M., Fung, F., Cannon, S., and Aikin, J. Is It Worth a Hoot? Qualms about OWL for Uncertainty Reasoning. ISWCURSW, 2005, 1-11.

[164] Sowa J. F. Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading, MA, 1984.

[165] Bouzeghoub, M., and Métais, E. Semantic Modelling and Object-Oriented Modelling: Two Complementary Paradigms. ER 1991: 325-348

[166] Ganter, B., Stumme, G., and Wille, R. Formal Concept Analysis: Foundations and Applications. LNAI, no. 3626, Springer-Verlag.

[167] Priss, U. Formal Concept Analysis in Information Science. In Cronin, Blaise (Ed.), Annual Review of Information Science and Technology, ASIST, Vol. 40. 2005.

[168] Quan, T.T., Hui, S.C., Fong, A.C.M., and Cao, T.H. Automatic Fuzzy Ontology Generation for Semantic Web. IEEE Trans. Knowl. Data Eng. 18(6): 842-856. 2006.

[169] Beneventano, D., Dahlem, N., El Haoum, S., Hahn, A., Montanari, D., and Reinelt, M. Ontology-driven Semantic Mapping. In proceedings of I-ESA '08.

[170] STASIS Project Deliverable 2.3.2. Semantic and ontology language specification. Version 10, January 4 2008.

[171] Berners-Lee, T. Linked Data. Web architecture note. (http://www.w3.org/DesignIssues/LinkedData.html)

[172] Jaffri, A., Glaser, H., and Millard, I. URI Disambiguation in the Context of Linked Data. In Proc. of the Linked Data on the Web Workshop, WWW2008, Beijing, China, 2008.

[173] Bergman M. K., and Giasson F. UMBEL - Upper Mapping and Binding Exchange Layer. UMBEL Project Proposal 12 July 2007. (http://www.umbel.org).

[174] Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., and Yergeau, F. Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation 26 November 2008.

[175] Thompson, H.S., Beech, D., Maloney, M., and Mendelsohn, N. XML Schema Part 1: Structures Second Edition. W3C Recommendation 28 October 2004.

[176] Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., and Wenke, D. OntoEdit: Collaborative Ontology Engineering for the Semantic Web. In: Proc. ISWC, Italy, 2002

[177] Euzenat, J., Loup, D., Touzani, M., and Valtchev, P. Ontology Alignment with OLA. Proceedings of the 3rd EON Workshop, 3rd Intl. Semantic Web Conference, Hiroshima (JP), November 2004.

[178] Maedche, A., Motik, B., Silva, N., and Volz, R. MAFRA - Mapping Distributed Ontologies in the Semantic Web. Proc. 13th European Conf. Knowledge Eng. and Management (EKAW 2002), Springer- Verlag, 2002, pp. 235–250.

[179] Obiedkov, S.A., and Kuznetsov, S.O. Algorithms for the construction of concept lattices and their diagram graphs. In Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD). Pages 289-300. 2001.

[180] Wang, D., Li, P., Hu, X., and Wei, X. A parallel algorithm to construct concept lattice. In Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007) Vol.2 Pages 119-213, 2007.

[181] Chen, B., Tan, H., and Lambrix, P. Structure-based filtering for ontology alignment. WETICE '06. 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, pages 364-369, June 2006.

[182] Euzenat, J. An API for ontology alignment. In Proc. 3rd conference on International Semantic Web Conference (ISWC), Hiroshima (JP), (Frank van Harmelen, Sheila McIlraith, Dimitris Plexousakis (eds). LNCS 3298, 2004), pp698-712, 2004

[183] Boussard, M., Hiribarren, V., Le Rouzic, J.P., Fodor, S., Bedini, I., Crespi, N., Marton, G., Moro, D., Macias, M., Dueñas, O.L., and Molina, B. Servery: Web Telco Marketplace. ICT-MobileSummit 2009. 10 - 12 June 2009, Santander, Spain.

[184] Bedini, I., Bertin, E., and Laga, N. Method for performing run time service composition and orchestration in a Web environment. Patent INPI number: 0954427 - 06/2009

[185] D'Aquin, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Sabou, M., and Motta, E. Watson: A Gateway for Next Generation Semantic Web Applications. Poster session of the International Semantic Web Conference, ISWC 2007.

[186] Tummarello, G., Delbru, R., and Oren, E. Sindice.com: Weaving the open linked data. In 6th International Semantic Web Conference, pages 552–565, 2007.

[187] Boris Motik, Peter F. Patel-Schneider, Bijan Parsia. OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Working Draft 08 October 2008.

[188] Kent, W. Data and Reality. 1stBooks Library, rev. 3/28/2000. ISBN-13: 978-1585009701.