Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

INS

Information Systems

INformation Systems

Ontologies in information integration within multimedia presentation generation

J. Salas Enrech

REPORT INS-E0515 OCTOBER 2005

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-3681

Ontologies in information integration within multimedia presentation generation

ABSTRACT
Nowadays an enormous quantity of heterogeneous and distributed information is stored in the current World Wide Web. Sharing of different information sources is needed. Recently the word Semantic Web has become very popular. The Semantic Web provides a common framework that allows data to be shared and re-used through ontologies. Ontologies make information explicit and can be used in the integration information task. The relation of an ontology with information sources or other ontologies plays an essential role in information integration and multimedia presentation. Multimedia presentation generators use a set of media items. The challenge is to combine these items in a coherent presentation to the user. For this, a large amount of information about these media items and their relations is needed. The collection and maintenance of information is a time-consuming costly effort that leads to the requirement for using existing information whenever possible to re-use the input metadata from the databases. In this document, we study the different approaches for combining information and propose an ontology construction method for developing shared ontologies. This document further illustrates the example integration of two art-media ontologies applying this process.

1998 ACM Computing Classification System: I.2.4, H.5.1
Keywords and Phrases: ontology mapping, semantic web, multimedia
Note: Master's thesis

Ontologies in Information Integration within Multimedia Presentation Generation

Thesis by J. Salas Enrech

Facultat d'Informàtica de Barcelona
Enginyeria Informàtica

Supervisors:
Dr. Jacco van Ossenbruggen
Prof. Dr. Lynda Hardman

October-May 2004/2005

October 5, 2005

Contents

List of Figures

Acknowledgements

Many people have contributed in one way or the other to the development of this thesis. I would like to specially thank a few.

First of all, my supervisor Jacco van Ossenbruggen, without his help I could not have done this thesis. Thank you for showing me your knowledge about Semantic Web and giving me total freedom for developing my thesis. Also thanks for all your comments and reviews about this document.

I want to thank Lynda Hardman for giving me the opportunity to work at CWI and for making me feel at home and welcome. Your comments about my masters thesis and your English corrections are greatly appreciated. Also thanks for all this wonderful chocolate from all the world that I have tasted.

I would like to thank everyone of the INS-2 Group. Lloyd, for his useful comments and for showing me the power of the Sesame construct queries. Joost, for helping me with all my technical problems. Katya for her useful comments about Chime. Katharina, my office mate for the first months, for helping me with good LaTeX tricks and advice.

I want to thank all my colleagues Frank, Stefano, Yulia, Lynda, Jacco, Joost, Katya and Katherina for the great time I have had at CWI. For letting me be part of your group and for all your input during my research.

Thanks also to Georgina Ramírez who give me very good advice and has always tried to help me. Thanks for these coffee breaks, for making me feel at home and for improving my Spanish.

Special thanks to my parents who always encouraged me to study. Thanks to my mum for showing me the pleasure of reading from when I was very young. I would also like to thank my sister, Isabel and my brother-in-law, Pepe, for providing me home during five years in Barcelona.

I also would like to thank to my Erasmus colleagues Fernando and Marina. Thanks for satisfying me in my free time with laughter and good vibes. I will always remember these travels around Netherlands and the party in Weesperstraat.

My final thanks but not least to Javier Vizcaíno, who has always been encouraging me during this thesis and for visiting me. Thanks for bearing all my simulations of the presentations and for your computer advice. Therefore, now I must help you with your thesis.

Javier Salas Enrech

Amsterdam, May 2005

Abstract

Nowadays an enormous quantity of heterogeneous and distributed information is stored in the current World Wide Web. Sharing of different information sources is needed. Recently the word *Semantic Web* has become very popular. The Semantic Web provides a common framework that allows data to be shared and re-used through ontologies. Ontologies make information explicit and can be used in the integration information task. The relation of an ontology with information sources or other ontologies plays an essential role in information integration and multimedia presentation. Multimedia presentation generators use a set of media items. The challenge is to combine these items in a coherent presentation to the user. For this, a large amount of information about these media items and their relations is needed. The collection and maintenance of information is a time-consuming costly effort that leads to the requirement for using existing information whenever possible to re-use the input metadata from the databases. In this document, we study the different approaches for combining information and propose an ontology construction method for developing shared ontologies. This document further illustrates the example integration of two art-media ontologies applying this process.

Chapter 1

Introduction

In recent years, there have been great advances in the development of techniques related to the *Semantic Web*. This is the next step to achieve in the World Wide Web: that computers partly understand the information. Nowadays, the majority of the information stored on Internet is in human readable format only. The main aim of Semantic Web techniques is to make the information on the Web *machine-readable* [6]. These new Semantic Web techniques often need to use ontologies to represent the semantics. Ontologies are an important factor for enabling interoperability in the Semantic Web. These ontologies allows users to organize information into, e.g., taxonomies of concepts [22].

The collection and maintenance of information is a time-consuming and costly effort that leads to the requirement for using existing information whenever possible. Furthermore, the reuse of these ontologies may be very attractive and useful. However, there are various problems when we try to integrate or adapt two or more ontologies for new purposes.

Semantic integration is an interest area in several fields, such as databases, information-integration and ontologies. One of the main bottlenecks in semantic integration is mapping discovery. Most researchers agree that semantic integration is one of the most serious challenges for the Semantic Web today [25]. For this, firstly we need a framework about the different semantic problems that we can find in the integration process. Secondly, an approach to carry out the integration task is needed.

Multimedia presentation generators use a set of media items. The challenge is to combine these items in a coherent presentation to the user. For this, a large amount of information about these media items and their relations is needed. The collection and maintenance of information is a time-consuming costly task that leads to the requirement for using existing information whenever possible to re-use the input metadata from the databases. If we try to combine information belonging to different sources for carrying out a presentation we find different heterogeneity problems.

Multimedia presentations need knowledge about the domain to be able to convey the essential semantic relations in the presentations. It requires knowledge about how to order, group and prioritize this information effectively. For instance, organizing the

information in a coherent storyline with a clear introduction, main body and conclusion. It requires knowledge about media design. First, we need to be able to select the most appropriate medium, and second, we need to understand the characteristics of that medium to achieve the communication goal. However, this is not an easy task because it exists a great variety of data sources and semantic relations [21]. Therefore, information integration is important in multimedia presentation generation. For instance, we can be interested in the generation of a presentation about artworks placed in different museums.

1.1 Objectives

The main aim of this thesis is showing the *integration of two or more ontologies* for sharing information. To achieve this purpose we have to decompose it into intermediate goals.

- introduce a framework which can be used to solve the semantic problems that we can find when we try to integrate ontologies for new purposes,
- propose an ontology construction method for developing shared ontologies which can be used to define terms from different vocabularies and to automatically translate them from one vocabulary to another,
- apply this process to a real case: the integration of two ontologies within the broad field of art presentation for the construction of multimedia presentations.

1.2 Structure of the thesis

The first four chapters are based on literature search and give us an introduction to our research.

- **In chapter 2**, we give the background for our research. We discuss languages for representing models of information semantics. We introduce some basic concepts about ontologies in order to provide basic common concepts and understanding.
- **In chapter 3**, we provide general information about ontology-based integration approaches and address the types of possible conflicts.
- **In chapter 4**, we analyze the problem of ontology heterogeneity which is characterized by different kinds of mismatches. Mismatches are the problems which can appear when we try to integrate two or more ontologies. We study these mismatches at different levels and the way to resolve them.

- **In chapter 5**, we describe the different categories of mappings between ontologies.

The following chapters are the main contributions. A framework to solve the integration mismatches and an ontology construction method for developing shared ontologies are introduced.

- **In chapter 6**, we explain a set of concepts from the field of shared ontologies. We describe our method or process of exchanging meaning between different sources and how to build a shared vocabulary to combine and integrate information belonging to different sources. We also investigate and analyze the real case about the integration of two different art ontologies: *Aria* and *Chime*. These ontologies contain pictures and description of works of art in the Rijksmuseum in Amsterdam.
- **In chapter 7** we conclude the thesis with a summary of the main goals and present an outline of future work.

Chapter 2

Basic ontology concepts

This chapter introduces a set of basic concepts of ontology engineering. Its main aim is to provide a basic understanding of ontologies, which are the basis of this work.

2.1 The Semantic Web

The Semantic Web is an initiative to develop machine processable data on the World Wide Web. The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. Semantic Web is a form of Web content that is meaningful to computers. [6].

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

Tim Berners-Lee, James Hendler, Ora Lassila, The Semantic Web, Scientific American, May 2001

The Semantic Web is built on layers of enabling standards. Figure 2.1 illustrates a layered approach to the Semantic Web and describes the main layers of the Semantic Web design and vision. We begin explaining at the bottom of the figure and we finish at the top [3, 37].

- URIs or Uniform Resource Identifiers are an important component of the current World Wide Web, which provides a way of identifying resources and links among resources.
- XML or eXtensible Markup Language is a language that lets one write structured Web documents with a vocabulary defined by the user. It is a fundamental component for syntactical interoperability.
- RDF or Resource Description Framework is a basic data model for writing simple statements about resources.

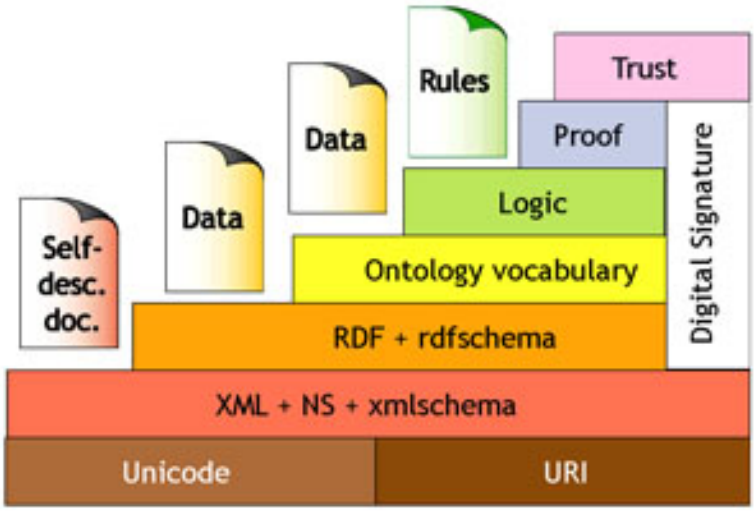

Figure 2.1: A layered approach to the Semantic Web from [5]

- RDFS or RDF Schema provides modeling primitives for organizing resources into hierarchies. Modeling primitives are: classes and properties, subclass and subproperty relationships, and domain and range restrictions. RDFS is based on RDF.
- The ontology vocabulary layer expands RDF Schema and allows the representations of more complex relationships between resources.
- The logic layer allows the writing of knowledge rules.
- The proof layers execute the use of rules and the representation of proofs in Web languages and proof validation.
- The trust layer will emerge through the use of digital signatures and other kinds of knowledge. The Web will be safer when users trust in its operations (security) and in the quality of information provided. Digital signatures and other kinds of knowledge can be used to authenticate the identity of the sender of a message, or of the signer of a document. It can also be used for detecting modifications in the original content of the message or document.

2.2 The role of ontologies

The concept *ontology* arises from philosophy. In this area, *ontology* (from the Greek $\omega\nu$ = being and $\lambda\grave{o}\gamma o\varsigma$ = word/speech) is the most fundamental branch of metaphysics. This branch of philosophy focuses the origins, essence and meaning of being.

In the nineties, the term *ontology* became embraced by computer science. In this context, *ontology* is a controlled vocabulary that describes objects and the relations between them in a formal way. *Ontology* comprises a grammar for using the vocabulary terms to express something meaningful within a specified domain of interest [19].

In general, every one has his/her individual view on the world and the things that he/she has to deal with every day. However, we have a common vocabulary to communicate this own conception of the world with another person. This vocabulary can be called a common *shared vocabulary*. The concepts involved in this shared vocabulary can be a little different but belong to a common understanding of the world. We call this concept a *conceptualization* of the world.

We know that a large number of languages and different viewpoints exist. We often find terms whose meaning can differ in different areas. For instance, the term *ontology* used in this section has a different use in philosophy than in computer science. This is the reason that we create a separation into different groups that share a common terminology and its conceptualization.

More formally, *an ontology is an explicit and formal specification of a shared conceptualization* [23]. A *conceptualization* refers to an abstract model of how people think about a real thing in the world. *Explicit* refers to the fact that the type of concepts used and the constraints on their use are defined. *Formal* means that the ontology should be machine comprehensible. *Shared* means that an ontology involves knowledge accepted by a group of people. Using ontologies we can describe the semantics of information sources and solve heterogeneity problems [18].

2.3 Standards and ontology languages

Various ontology languages have been developed, focusing on different aspects of ontology modeling. In this section, we show how ontologies should be represented and share information. We analyze some ontology languages that are compatible with existing Web technology. We will explain briefly RDF, RDFS and OWL because ontologies to integrate in Chapter 6 are written in these languages.

2.3.1 Semantic-oriented languages

- RDF

  RDF, *Resource Description Framework*, is a family of specifications which was developed by W3C. RDF provides a foundation for representing and processing metadata. An RDF graph is a set of triples, each triple consisting of a subject, a predicate and an object. The subject is the resource, the ”thing” being described. The predicate is what trait or aspect about that resource that is being described, and often expresses a relationship between the subject and the object. The object is the object of the relationship or value of that trait. This notion is useful because

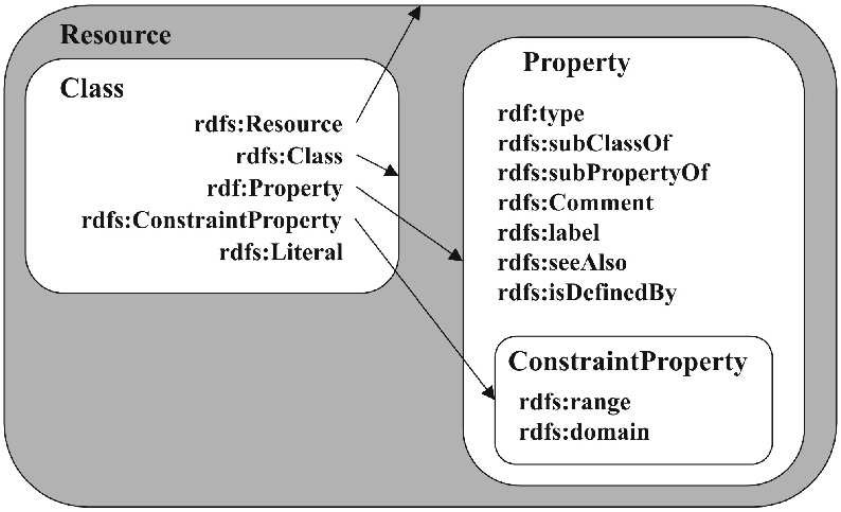

Figure 2.2: Modeling components of RDF Schema from [40]

RDF allows subjects and objects to be interchanged. For instance, any subject from one triple can play the role of an object in another triple [38].

- RDFS

  RDF Schema expands RDF by adding a particular vocabulary for RDF data. In particular, RDF Schema defines class, subclass relations, property, subproperty relations, and domain and range restrictions. RDF and RDFS are different and complementary at the same time. The combination of both is sometimes denoted as RDF(S) [40].

  Figure 2.2 illustrates the different modeling components of RDF schema.

- OWL

  RDF and RDFS allow the representation of ontological knowledge. However, a number of other features are missing. For instance, we cannot define local scope of properties, disjointness of classes, boolean combinations of classes or cardinality restrictions.

  OWL, *Web Ontology Language*, is designed to process the content of information instead of just presenting information to humans. OWL facilitates greater interoperability of Web content than XML and RDF(S) by providing a richer vocabulary along with a formal semantics. OWL is composed of three sublanguages: OWL Lite (classification hierarchy and simple constraints), OWL DL (adding class axioms, boolean combinations of class expression and arbitrary cardinality) and OWL Full (permits also meta-modeling facilities in RDF(S)). Ontology developers need to select the most suitable sublanguage for their needs [3, 39].

2.3.2 Database and query languages

In this subsection we deal with databases and query languages which work with RDF and RDFS. We analyze and summarize briefly Sesame and SeRQL because this database and query language we use to integrate ontologies in Chapter 6.

- Sesame

  Sesame is an open source Java framework for storing, querying and reasoning with large quantities of metadata in RDF and RDF Schema. It can be used as a database for RDF and RDF Schema, or as a Java library to work with RDF internally. Sesame provides the necessary tools to parse, interpret, query and store all information in an RDF file [1, 9, 10, 30].

- SeRQL

  SeRQL, *"Sesame RDF Query Language"* pronounced *circle*, is an RDF/RDFS query language that addresses practical requirements from the Sesame user community. It is very similar to SPARQL ("Protocol And RDF Query Language"). [31]

  SeRQL combines the best features of other query languages such as RQL, RDQL, N-Triples or N3, and adds some of its own. Some of SeRQL's most important features are: graph transformation, RDF Schema support, XML Schema datatype support, expressive path expression syntax and optional path matching.

  SeRQL has a *construct* clause where you can specify which triples should be returned. However, construct queries can also be used to do *graph transformations* or to specify simple rules. Graph transformation is a powerful tool in domains where mappings between different vocabularies need to be defined [2, 8, 29].

2.4 Existing ontologies

We need a set of terms to build our shared vocabulary. There are several vocabularies and conceptualizations of the domain accepted as a standard. The selection of these sources is an important step when building a shared ontology. The majority of ontologies discussed in this section are not available in RDF. However, the number of ontologies available in RDF is increasing.

- **Upper-Level Ontologies**

  The use of an upper-level ontology provides us with a vocabulary. We can use these upper-level ontologies to find the bridge concept when we begin to create the shared vocabulary. Some examples are Dublin Core [15] and VRA Core [36].

  - **Dublin Core**

Dublin Core [15] has become an important part of the emerging infrastructure of the Internet. The Dublin Core metadata standard is a simple element set for describing a wide range of resource. The Web metadata might be used to infer semantic relationships. The Dublin Core Metadata Initiative (DCMI) is the organization dedicated to promoting the adoption of these metadata and developing specialized metadata vocabularies for describing resources.

- **VRA Core**

  The VRA Core Categories Version 3.0 [36] consists of a single element set with which we can create records to describe works of visual culture as well as the images that document them. As Dublin Core, only one object or resource may be described within a single metadata set. The elements are designed to facilitate the sharing of information among visual resources collections about works and images. These elements may not be sufficient to fully describe a local collection and additional fields can be added for that purpose. Moreover, every element may be repeated as many times as necessary within a given set to describe the image or work.

- **Scientific Classification**

  Scientific Classifications are another standard way to describe a domain. In order to effectively study plants and animals, all scientists need to use the same names. Taxonomy is a subject-based classification that arranges the terms in a controlled vocabulary into a hierarchy [32].

- **Thesauri**

  Thesauri are controlled vocabularies of terms in a particular domain with hierarchical, associative and equivalence relations between terms [34].

  Domain thesauri are appropriate sources for finding concept names for the shared vocabulary. These thesauri sometimes contain definitions of the term included which might provide guidance for the definition of concepts. Example thesauri are MeSH[1], Wordnet[2], AAT and ULAN.

  - **AAT**

    The Art and Architecture Thesaurus (AAT) [20] was developed as a vehicle for indexing catalogs of art objects. The AAT is used in cultural heritage institutions for cataloging art collections. The AAT was developed according to the ISO standard for the definition of monolingual (ISO2788) and multilingual thesauri (ISO5964). These standards consist of a record structure with a set of attributes and three relations: hierarchical relation

[1] Medical Subject Headings, http://www.nlm.nih.gov/mesh/
[2] Wordnet RDF Representation, http://www.semanticweb.org/library/

(broader/narrower term), equivalence of terms (synonyms and lexical variants) and an associative relation (related terms). This thesaurus is available in English, Spanish, Dutch and French.

– **ULAN**

The Union List of Artist Name (ULAN) [33] is a structured vocabulary which contains around 259,000 names and other information about artists. Names in ULAN may include given names, pseudonyms, variant spellings, names in multiple languages, etc. Among these names, one is the *preferred name*.

The focus of each ULAN record is an artist. Currently there are around 118,000 artists in ULAN. In the database, each artist record is identified by a unique numeric ID. Linked to each artist record are names, related artists, sources for the data, and notes.

– **Data catalogs**

Data catalogs provide metadata and terminology about their data including lists of variables with definitions and the time period for the data collection. Furthermore, knowledge catalogs provide online tools for collaborators and practitioners to describe and share their practice, research methods, and data [32].

Chapter 3

Ontologies in information integration

3.1 Introduction

Information is often heterogeneous and distributed. When we want to share different information sources, many technical problems have to be solved. The problem appears when we try to obtain information from different information sources. We can distinguish two main heterogeneity problems: *structural heterogeneity* and *semantic heterogeneity*. The *structural heterogeneity* means that different information sources store their data in different structures. The *Semantic heterogeneity* considers the contents of an information item and its intended meaning. Semantic conflicts appear when two sources do not use the same interpretation of the information. The use of ontologies for the description of a domain is a possible approach to overcome the problem of semantic heterogeneity.

Ontologies can be used in an integration task to describe the semantics of the information sources and to make the contents explicit. There are different ways for relating different ontologies: *single ontology approaches, multiple ontology approaches* and *hybrid approaches* [41]. The ideas of these approaches are depicted in figure 3.1

3.2 Single ontology approach

The principal aim is to develop a single global ontology that is the result of merging the related ontologies. This approach provides a shared vocabulary for the specification of the semantics. All information sources are related to the global ontology. We have to make semantic mappings between the global ontology and the information sources.

Mapping identifies semantically corresponding terms of different source ontologies whose terms are semantically equivalent or similar. For instance, in the example ontologies in Chapter 6, the term *Artist* in one ontology is semantically equivalent to the term *Artist* in the other one. We have to be careful that two terms with the same name does not mean that they are semantically equivalent. These semantic mappings are not a difficult task if all information sources involved have the same view on a

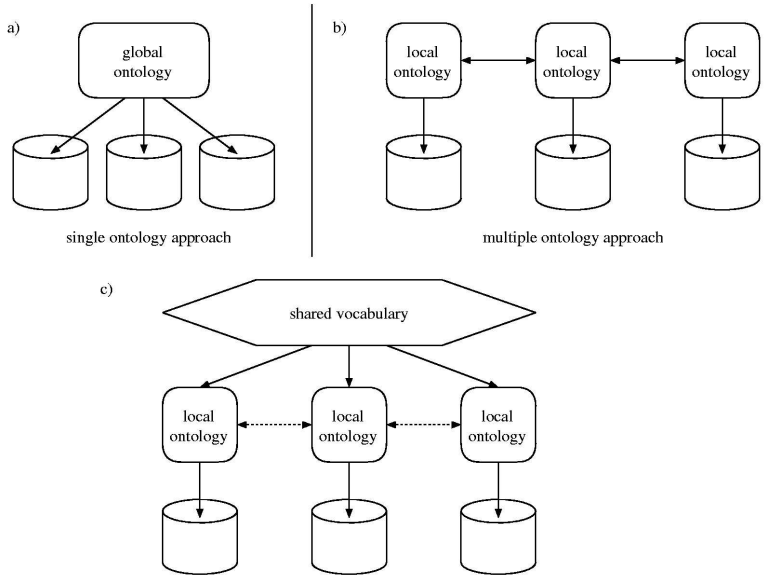

Figure 3.1: Three possible ways for using ontologies for content explication, from [41]

domain. But if one information source has a different point of view, for instance by providing another level of granularity, is better to try another approach.

The advantage of this approach is that the global ontology contains all information that is needed for information sharing. The drawbacks are that we will find many problems if we want to change something. Single ontology approaches are susceptible to changes in the information sources. These changes can imply others in the global ontology and indirectly, in the mappings with other information sources. These drawbacks lead us to the use and development of multiple ontology approaches.

3.3 Multiple ontology approach

The goal of this approach is avoid the creation of large global ontologies. In multiple ontologies, each information source is described by its own ontology. Modifications in one information source or the adding or removing of sources are possible.

Advantages are the absence of a global ontology and the support of heterogeneous points of view of a domain. A drawback of this approach is that it is extremely difficult to compare different source ontologies. We need to specify mappings between all ontologies to compare them. For instance, if we have to integrate ten ontologies, we have to create ten local ontologies and relate each of them with the nine others. We can find many semantic heterogeneity problems when we want to use this approach in practice.

	Single Ontology	Multiple Ontology	Hybrid Ontology
Implementation effort	straightforward	costly	reasonable
[illegible]	similar view of a domain	supports heterogeneous views	supports heterogeneous views
Adding/removing of sources	need for some adaptation in the global ontology	providing a new source ontology; relating to other ontologies	providing a new source ontology
Comparing multiple ontologies	not possible	difficult because of the lack of a common vocabulary	[illegible] use a common vocabulary

Figure 3.2: Advantages and drawbacks of the different ontology integration approaches, from [41]

3.4 Hybrid ontology approach

This approach takes the advantages of the *Single Ontology approach* and the *Multiple Ontology Approach*. *The Hybrid Approach* is similar to the *Multiple Approach Ontology*: each information source is described by its own ontology. The difference is that a shared vocabulary is built below the source ontologies to make them comparable. The shared vocabulary defines basic terms of the primitives of a domain. This shared vocabulary is useful to compare the source ontologies to each other and can also be an ontology.

The advantage is that new information sources can be added easily without the need of modification in the other mappings or in the shared vocabulary. We only have to relate the new information source with the terms and properties belonging to the shared vocabulary. The use of a shared vocabulary makes the source ontologies comparable and avoids the disadvantages of multiple ontology approaches. A drawback is that existing ontologies cannot be reused easily. We must carry out a deep analysis and study of the new ontology because it has to refer to the shared vocabulary. This new ontology added will be comparable with the other ontologies involved when all relations with the shared vocabulary are made.

Figure 3.2 gives a summary of the three approaches.

Chapter 4

Mismatches

4.1 Introduction

The goal of the Semantic Web is that computers understand the information on the World Wide Web. Ontologies are necessary to describe a domain of discourse formally. Typically, an ontology consists of a finite list of terms and the relationships between these terms. Nowadays, there are many ontologies, and the idea of reuse of these is very interesting and useful. We could find different problems when we try to adapt an ontology for new purposes or integrate two or more ontologies. Differences between ontologies are called mismatches in [22], and will be used throughout this work.

The aim of this chapter is to analyze the problem of ontology heterogeneity, which is characterized by different kinds of mismatches between ontologies. We need to solve this heterogeneity problem in order to combine multiple ontologies, which is needed in many. Mismatches can appear at two levels: *language level* and *ontology level*. The first is the level of the language primitives that are used to specify the ontology. The mismatches which appear at this level are the ways to define classes, relations, etc. The second happen when two or more ontologies are combined and these ontologies describe overlapping domains. A mismatch at this level is a difference in the way the domain is modeled. The main point in this chapter is *mismatches at the ontology level*. Figure 4.1 depicts a framework of issues, related to the integration of ontologies.

4.2 Language level mismatches

Mismatches at the language level occur when the ontologies are written in different languages. In [22], four types of mismatches are distinguished:

- **Syntax.** Each ontology has its own syntax. For instance, to define the class *Artist* in RDF Schema, we write *<rdfs:Class about="Artist">*. In LOOM, the expression *defconcept Artist* is used to define the same class. It is typical that an ontology language has several syntactical representations. The solutions to solve this

type of mismatch is syntax rewriting.

- **Logical representation** is the difference in the representation of logical notions. This type of problem is a little more difficult. For instance, there are ontology languages where it is possible to describe that two classes are disjoint. This means that an individual that is a member of one class cannot simultaneously be a member of another specified class. For instance in OWL:

  <owl:Class rdf:about="Painting">
  <owl:disjointWith rdf:resource="#Furniture"/>
  <owl:disjointWith rdf:resource="#Building"/>
  </owl:Class>

  In other ontologies this concept does not exist and it is necessary to use negation in subclass statements. For instance: *(A subclassOf NOT(B)) AND (B subclassOf NOT(A))*. One possible solution for this problem is to create translation rules.

- **Semantics of primitives.**

  This type of mismatch is the difference at the metamodel level. Sometimes the same name is used for more than one ontology but the semantics can differ. For instance, there are several interpretations of *A equalTo B*. A solution for solving to this problem is to rename the primitives.

- **Language expressiveness mismatches.**

  This is the most difficult mismatch at the metamodel level to resolve. It happens when one language can express something that in another one is inexpressible. For instance, some languages support expressions such as lists, sets, etc. that others do not support. One solution is to create translation rules.

4.3 Ontology level mismatches

This type of mismatch can appear when both ontologies are written in the same language or not. In [22], *ontology level mismatches* are divided into: *conceptualization mismatches* and *explication mismatches*.

4.3.1 Conceptualization mismatch

A conceptualization mismatch is a difference in the way a domain is interpreted, which implies different ontological concepts or different relationships between those concepts. We can find two mismatch types in this category. The mismatches described below cannot be solved automatically and require knowledge and decisions of a domain expert.

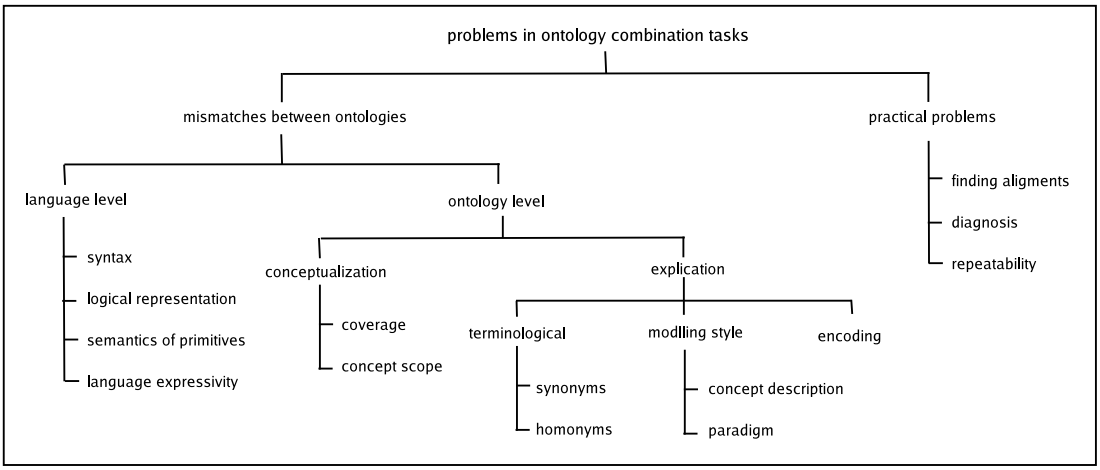

Figure 4.1: Framework of issues on ontology integration

- **Scope**

  The scope mismatch or class mismatch appear when two ontologies have two classes that apparently represent the same concept, but do not have the same instances of the object. The most standard example is the class *employee*, that several organizations use slightly different concepts of this class. For instance, *employee* can mean *"all people that have a room within a company"* or *"all people that get paid by a company"*.

- **Model coverage and granularity**

  These mismatches appear in the part of the domain that is covered by the ontology and is related with the ontology granularity. This is often the reason why ontologies are merged. For instance, we suppose that we have three different ontologies about cars: one ontology can represent cars but not trucks. Another one might model cars and trucks but only classify them into a few categories. The last one can do a detailed distinction between types of trucks based on their characteristics. This mismatch is often not a problem, but can be a motive to use two or more ontologies together. In this case, we will have to align the overlapping parts of the ontology.

4.3.2 Explication mismatch

Explication mismatch is a difference in the way the conceptualization is specified. There are three different subclassifications: *terminological mismatches*, *modeling style* and *encoding*.

Terminological mismatch

- **Synonym terms**

  This type of mismatch occurs when two or more concepts are represented by different names. For instance, in one ontology the term *Artefact* has the same meaning as the term *Artwork* in another ontology. Another example is the use of the term *car* in one ontology and the term *automobile* in another ontology.

  This mismatch occurs because of the natural language in which ontologies are described differ. To solve this problem we can use thesauri to integrate ontologies with synonyms in different languages. This mismatch takes a large amount of human effort and we have to be careful when we try to solve this type of problem because we can indirectly generate scope mismatches.

- **Homonym terms**

  Also called *concept mismatch*. The meaning of a term is different in another context or domain. For instance, the term *conductor* has a different meaning in a music domain from in an electrical engineering domain. To solve this mismatch, human intervention is required.

Modeling style

Modeling style is related to the paradigm and conventions taken by developers.

- **Paradigm**

  A paradigm refers to a pattern or model. A collection of assumptions, concepts, practices, and values that constitute a way of viewing reality, in particular for an intellectual community that shares them. Different paradigms can be used to represent concepts such as time, action, plans, causality, propositional attitudes, etc.

- **Concept description**

  Also called modeling conventions. Several choices can be made for the modeling of concepts in ontologies. We can make the distinctions by attributes or subclasses. A possible choice in concept descriptions is the way in which a hierarchy is built. For instance, we can model the distinction between paint and non-paint books as: $art < book < paint\ book$ or as $art < paint\ book < book$, or even as subclass of book and art at the same time. Another example: a motorbike can be for one ontology a bike with motor and for another one, a motor-vehicle with only two wheels.

Encoding

This is a trivial type of mismatch that appear when values in the ontologies may be coded in different formats. For instance, we can represent the distance in miles or kilometers, the height in centimetres or inches, the date may be appear as *dd/mm/yyyy* or as mm-dd-yy, etc. We can find many mismatches of this type, but these are all very easy to solve. It is only necessary to define a transformation step to eliminate all the differences.

4.4 Practical problems

We have studied the technical problems, but there are also practical problems that require serious human effort. These mismatches are done by hand. In [22] three important types are distinguished:

- it is difficult to **find** the terms that need to be aligned.
- when we make a mapping, its **consequences are difficult to see**.
- the **repeatability** of merges. Frequently, the sources that are used for the merging continue to evolve. The alignments that are created for the merging should be as reusable as possible for the merging of the revised ontologies. A solution can be an executable specification of the alignment.

4.5 Approaches and techniques to solve mismatches

The focus of this work is on ontology level mismatches, or in other words, semantic mismatches. We have seen different types of mismatches. We will briefly describe different techniques to solve language mismatches.

4.5.1 Solving language mismatches

We start looking at techniques for solving language mismatches. There are several approaches for solving the problem of integrating ontologies that are written in different knowledge representation languages. In [22] four approaches to enable interoperability between different ontologies at the language level have been identified.

- **Aligning the metamodel**

  In [7], an approach to transforming information from one representation to another is described. The paper focus is on model-based information where the information representation scheme provides structural modeling construction, analogous to a data model in a database. The semantic mismatches at the ontology level are not covered by this approach.

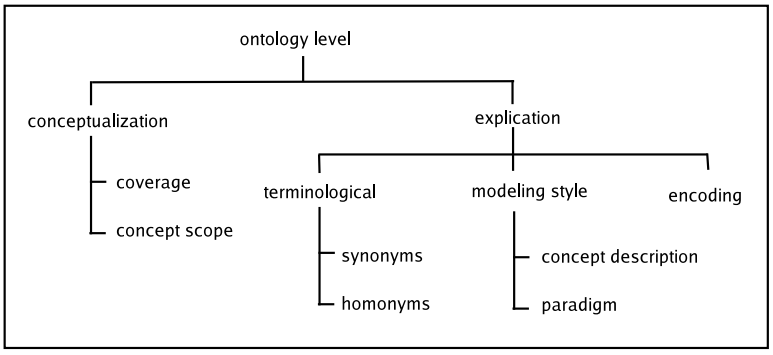

Figure 4.2: Semantic mismatches that are analyzed in this document

- **Layered interoperability**

  In [24], aspects of the language are divided in clearly defined layers and interoperability is resolved layer by layer. There are three main layers: the syntax layer, the object layer, and the semantic layer. It seems that a separation between different layers may facilitate interoperability. However, only some of the language mismatches are solved.

- **Transformation rules**

  The relation between two constructs in different ontology languages is handled by a unidirectional rule that specifies the transformation from one ontology to the other. In [12] this approach is described.

- **Mapping onto a common ontology**

  In this approach, the constructs of an ontology language are mapped onto a common ontology, for instance OKBC (The Open Knowledge Base Connectivity) [13].

4.6 Semantic Integration

4.6.1 Mappings

Mappings are the glue for integrating information sources. More formally, a mapping consists of relating similar concepts or relations from different sources to each other by an equivalence relation. Mappings are the connection of an ontology to other parts of the application system and preserve the *meaning* of the elements involved. There are many different kinds of models that we might want to connect with semantic mappings (thesauri, database schemes, ontologies...) and many languages exist for representing these models.

In the following, we briefly briefly mappings between different ontologies used in a system. This mapping division is taking from [22].

Inter-Ontology mapping

The majority of the existing information integration systems use more than one ontology to describe the information. We can find several problems when we try to map different ontologies. We only consider general approaches that are used in this area [22].

- **Defined mappings**

  The translations between ontologies are made by mediator agents. These provide to the user the possibility to define 1-1 mappings between classes and values or mappings between compound expressions. The advantage of this mapping is the great flexibility. There is, however, a large drawback: in this approach it is difficult to guarantee a preservation of semantics because the user is free to define the mappings even if they do not make sense or produce conflicts.

- **Lexical relations**

  This approach provide semantics for mappings between concepts in different ontologies. Relationships used are *synonym, hypernym, hyponym, overlap, covering* and *disjoint*.

- **Top-Level grounding**

  Top-Level Grounding relates all ontologies to a top-level ontology. This can be done by inheritance techniques. Using this type of mapping we can resolve conflicts and ambiguities. Another advantage is that we can establish connections between concepts from different ontologies in terms of common superclasses. For instance, in the following chapters we make semantic correspondences between terms or properties using *subClassOf* and *subPropertyOf*.

- **Semantic correspondences**

  This approach tries to find semantic correspondences using a shared vocabulary. In order to avoid arbitrary mappings between concepts, these approaches have to rely on a common vocabulary for defining concepts across different ontologies.

4.7 Conclusion

To solve ontology level mismatches, a complete understanding of the meaning of concepts involved is required. This task cannot be fully automated. However, there are many tools and approaches that concentrate on ontology level mismatches. These approaches suggest alignments and mappings. For instance, such tools help the user to find concepts in the different ontologies that will be good candidates for merging.

Conceptualization mismatches and most explication mismatches are hard to solve. Some mismatches can be solved automatically, but some will need our intervention to

solve them. In order to integrate ontologies, we need to know the different semantic level mismatches.

We are going to apply the top-level grounding approach in the next chapters. We establish connections between different ontologies in terms of common superclasses using *subClassOf* and *subPropertyOf*. When we carry out this approach in practice we find a group of mismatches which we have seen in this chapter.

Chapter 5

Aria and Chime integration

5.1 Introduction

We have seen many techniques and approaches for integrating information from different sources. In this chapter, we are going to apply the hybrid approach and our shared ontology construction method for integrating two ontologies: *Aria* and *Chime*. Both ontologies describe art-media domains. Each one comprises several files written in RDF(S) which contain information about artist, artworks, media-items, etc. We have chosen a hybrid approach because both ontologies have several files and it is more easier to add, remove and modify sources. It is also better for comparing ontologies.

Our approach relies on the use of shared ontologies. Shared ontologies as described in Chapter 3 are useful for identifying semantic correspondences. Our approach is based on the definition of a common terminology similar to an upper-level ontology. This common terminology consists of a set of elements that are used to describe classes from different ontologies in terms of formal concept expressions. The shared vocabulary must be sufficiently expressive and should be as small as possible for reducing the effort and time building it. Also, later it will be more easy to achieve the semantic mappings between the information sources and the shared vocabulary. Therefore, a process to build such shared terminologies is required. Firstly, we show this process for building shared ontologies. Secondly, we give an explanation about *Aria* and *Chime*. Finally, we apply our shared ontology construction method to our ontologies.

5.2 The ontology construction method

Our development process is based on stepwise-refinement. The method is a combination of the process presented in [11, 32] and my own contribution. These approaches are insufficient because they do not consider the different mismatches that be can found in the integration task. Therefore, they say nothing about how to resolve these mismatches. We carry out an analysis of mismatches and subsequently we solve them in our ontology construction method. We also define the connections with our top-

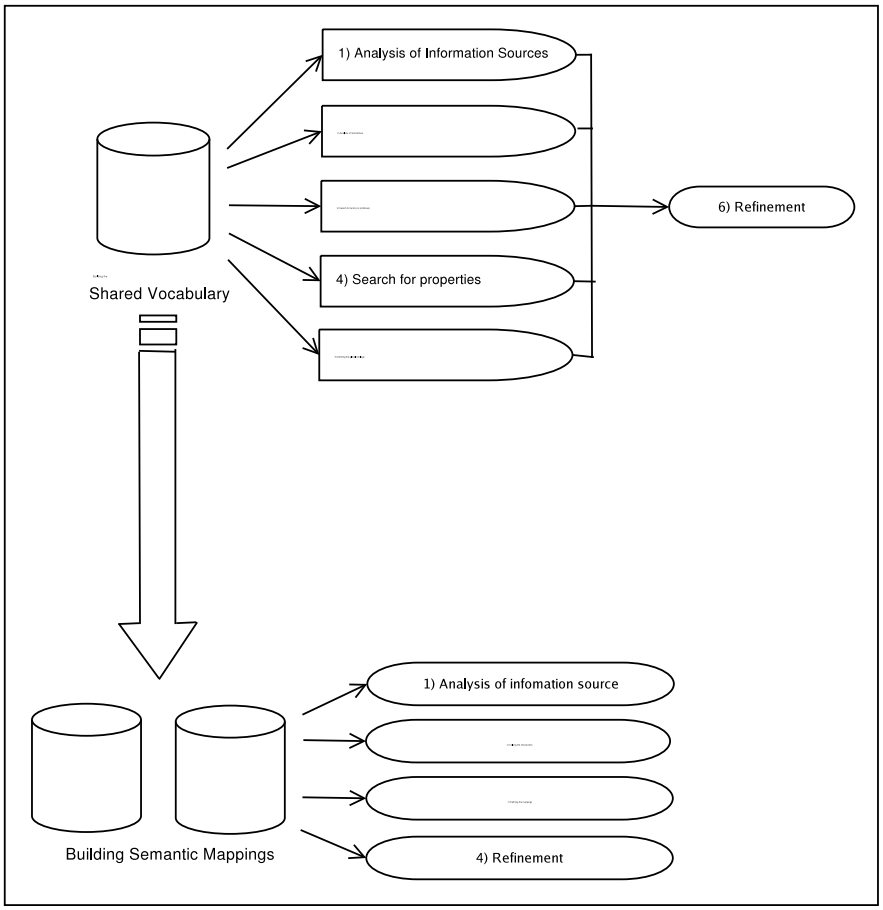

Figure 5.1: Ontology Construction Method

level ontology and we present a framework for solving ontology level mismatches. Figure 5.1 illustrates the procedure designed.

As we can see, the method has two main stages: *building the shared vocabulary* and *building semantic mappings*. Each stage includes a set of steps that must be carried out.

5.2.1 Building the shared vocabulary

As Figure 5.1 shows, this stage contains six main steps: *analysis of information sources, analysis of mismatches, search for terms or primitives, search for properties, definition of the global ontology* and *refinement of definitions*. We describe the steps as follows.

1. **Analysis of Information sources**

The first step is to decide what we want to translate. It implies a complete analysis of the information sources: what information is stored, how it is stored, the semantic meaning of this information, etc.

2. **Analysis of Mismatches**

   In this step we must localize the different types of mismatches between the ontologies involved in the integration process. We have seen in Chapter 4 the different types of mismatches that we can find.

3. **Search for Terms or Primitives**

   In this step we have to select the terms or primitives that are going to belong to our shared vocabulary. The shared vocabulary must be as expressive as possible to later reduce the effort and time of making the mapping between the information sources and the shared vocabulary.

   We have to find a concept which subsumes all classes from the sources. This concept is called a *bridge concept* [32] because it makes a semantic translation from one source to another. We define the properties and attributes of this concept. The most general *bridge concept* is a concept that includes every other possible concept. It is recommended to select a concrete *bridge concept* for an exact classification. We can define more than one if we need it. We decide how many terms are going to be a member of our shared vocabulary. It depends on the level of description of our shared vocabulary.

4. **Search for Properties**

   Each concept has the properties that describe it. In this step we have to find suitable properties for the *bridge concept* and the concepts found in the last step. For instance, an artwork can be described using concepts such as size, material, creator, etc.

5. **Defining the Global Ontology**

   In this step we use the terms and its properties chosen in the previous steps to build the global ontology.

6. **Refine Definitions**

   This process is based on the life cycle. It allows us to step back at any time to add, remove and modify ontologies, terms, properties, etc. It is important to achieve a vocabulary as simple and as expressive as possible to later make the semantic mappings between the information sources and the shared vocabulary.

   Sometimes we have to adapt our ontology because our shared vocabulary will not express all the terms that we want to translate. In this case, we have to refine our ontology in order to resolve the problem.

ONTOLOGY LEVEL			GENERAL SOLUTION TECHNIQUES
CONCEPTUALIZATION MISMATCH	CONCEPT SCOPE		- Study and analysis of the semantics involved in this mismatch - Join and align the overlappings parts of the ontology
	MODEL COVERAGE & GRANULARITY		- This mismatch is often not a problem - Align the overlapping parts of the ontology - Conversion from subclass to attribute or vice versa
EXPLICATION MISMATCH	TERMINOLOGICAL MISMATCH	SYNONYM TERMS	- Use of thesauri - Rename concepts. Be careful because we can generate indirect scope mismatches - Semantic connections betweeen information source and the shared vocabulary.
		HOMONYM TERMS	- Use of thesauri - Rename concepts
	MODELLING STYLE	PARADIGM	- To make a translation from one paradigm to the other
		CONCEPT DESCRIPTION	- Semantic connectios between information source and the shared vocabulary. - Conversion from subclasses to atrtribute or vice versa
	ENCODING		- Easy to solve - Wrapper or transformation step to eliminate all the differences

Figure 5.2: Framework to solve ontology level mismatches

5.2.2 Building semantic mappings

As Figure 5.1 shows, this stage contains four steps: *analysis of information source, solving the mismatches, defining mappings* and *refinement*.

1. **Analysis of information source**

   This step is similar to the first step with the same name in the previous stage with one exception: we must do this analysis without taking into account the other information sources. With our shared vocabulary as a reference our aim is to get a semantic connection between one information source and the shared vocabulary.

2. **Solving the mismatches**

   The next step for building the semantic mappings is to resolve the mismatches that have been found in the analysis of mismatches belonging to the last stage. In order to integrate ontologies, it is better to separate mismatches that are hard to solve and those that are not. We will begin resolving the easy mismatches to proceed with the hard ones. Conceptualization mismatches often need human efforts to be solved. Most explication mismatches can be solved automatically, but the terminological mismatches may be difficult.

   Figure 5.2 illustrates a framework to solve ontology level mismatches.

3. **Defining the mappings**

   Each information source defines and describes its own terms and properties independently of the other sources of information. In this step we relate a source ontology to our shared vocabulary. We implement the semantic connection between the terms or properties that are semantically equivalent or similar in a separate file. Through these semantic connections we can resolve a good proportion of the mismatches found in the last steps.

4. **Refinement**

   In the last step we defined the relations between the ontologies and the shared vocabulary. Subsequently, we must solve the mismatches that we cannot resolve in the last step.

5.3 Aria and Chime integration

5.3.1 Topia project

The Topia project, based on Aria (Amsterdam Rijksmuseum Interactive) repository, comprises a system which generates a media presentation structure around media objects. The Aria repository contains pictures and descriptions of the full Rijksmuseum collection. This project aims to show how a semantically rich multimedia database can be the basis for a ”storyline” that conveys the underlying relationships in the data. Aria is built with Semantic Web technologies to provide computable descriptions of archived media.

Through semantic queries the user can retrieve media items from the repository. Aria has a keyword search interface based on a list of relevant URIs. The search is applied by a SeRQL [2] query that searches for literal values and returns all resources that appear in triples as subjects with the literal as the object.

In this section, we give only a brief summary of the Aria project. See [27] for a more complete overview. See [4] for an on-line demo. Figure 5.3 illustrates the Aria project’s implementation.

5.3.2 Chime project

The aim of the Chime project (Cultural Heritage in an Interactive Multimedia Environment) is to investigate the use of semantic models for tailoring the presentation of cultural information extracted from existing repositories to different types of users.

A part of the Chime project is SampLe (Semi-Automatic Presentation generation Environment) [17], that is a framework where system support is provided at any stage of the presentation building process. The support is managed by incorporating explicit knowledge about the domain, narrative structures, media modalities and tasks involved in the process of multimedia presentation creation. SampLe consists of five layers

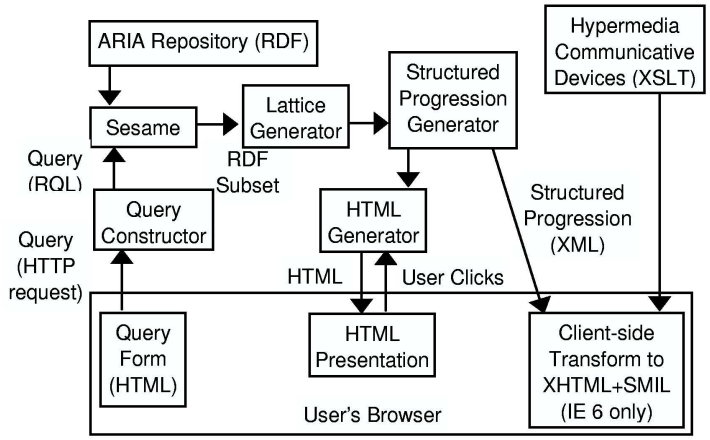

Figure 5.3: Aria's implementation design from [27]

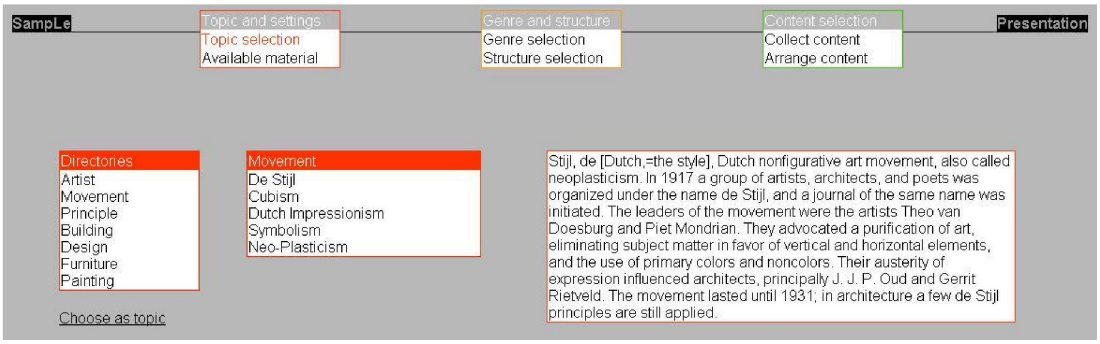

Figure 5.4: SampLe topic selection phase and the overall view on the interface from [16]

each dealing with a certain user task. These tasks are: theme identification, building the logical structure of the presentation, selecting the material, organizing the material and a suitable material presentation. Figure 5.4 illustrates a screenshot from SampLe.

The aim of the system is to facilitate time-consuming actions (exploration and search) and to support a user in performing creative actions. Users communicate with SampLe using a web browser. All application functionality is realized on the server side, using standard tools. The three main components of the SampLe system are the Exploration, Discourse structure and Material collection support mechanism. These three components are implemented using XSLT and XSP. The RDF repositories are accessed by using an XSLT extension that allows SeRQL to be combined with XSLT. Figure 5.5 illustrates the SampLe architecture. See [14] [16] and [17] for a more complete overview.

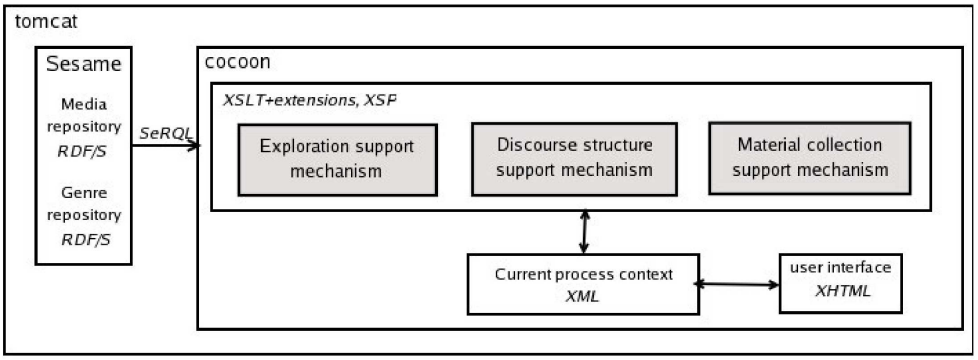

Figure 5.5: SampLe architecture from [16]

5.4 Applying the ontology construction method

Our proposal is based on a hybrid approach. We have seen in Chapter 3 the main advantages of this approach:

- New information sources can be added without need of modification. We only have to define the semantic mappings between the information source and the shared vocabulary
- The different information sources are easily comparable.

  Figure 5.6 illustrates the hybrid approach applied to our integration task.

We have seen a method for building shared ontologies. We are going to apply this method to our two ontologies: *Aria* and *Chime*.

Building the shared vocabulary

The first stage consists in building the shared vocabulary which will contain a set of terms and properties to define it. We describe this process step by step.

1. **Analysis of information sources**

   Before we begin with the construction of our shared vocabulary, we have to carry out a complete and exhaustive analysis of the information sources. In other words, we have to check all information that is stored in each information source, how this information is stored, the explicit and implicit semantic meaning, etc. However, analyzing the information and knowing the structure from the ontologies is not an easy task. We remind that *Aria* and *Chime* are encoded in RDF and RDFS and they are not a small ontologies. Each one has several files that describe them. We known that it is difficult to make an idea about the structure of an ontology written in RDF(S). In RDF, we have many small chunks of

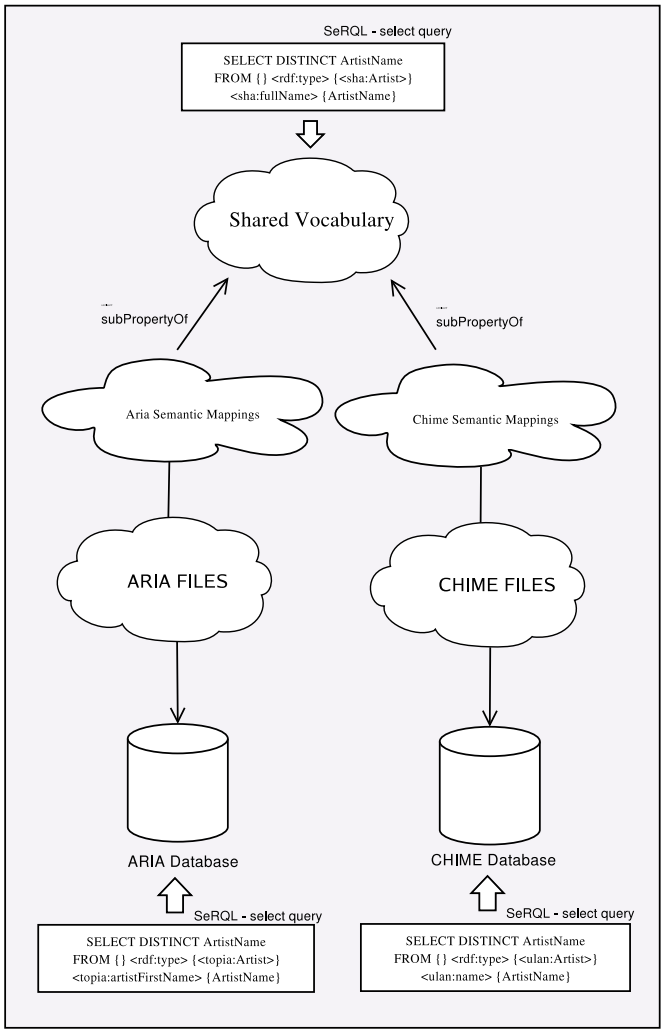

Figure 5.6: Hybrid approach: semantic mappings and query structure

information with many explicit relations among them. We need different tools and techniques to get this objective. Different approaches to displaying RDF are explained in [28].

Several systems for automatic generation of hypermedia from RDF repositories have made some progress in displaying RDF: Haystack [26], Hera [35], Disc [21] and Noadster [28]. We have used Noadster for exploring our ontology repositories: *Aria* and *Chime*. *Noadster* is a demonstrator system which illustrates ways of structuring information and conveying the structure allowing the user to explore a particular view of the repository. Noadster uses a single XSLT extension that allows the querying of a Sesame RDF repository [9, 10]. Noadster does not need any knowledge of the RDF vocabulary used and can be applied to

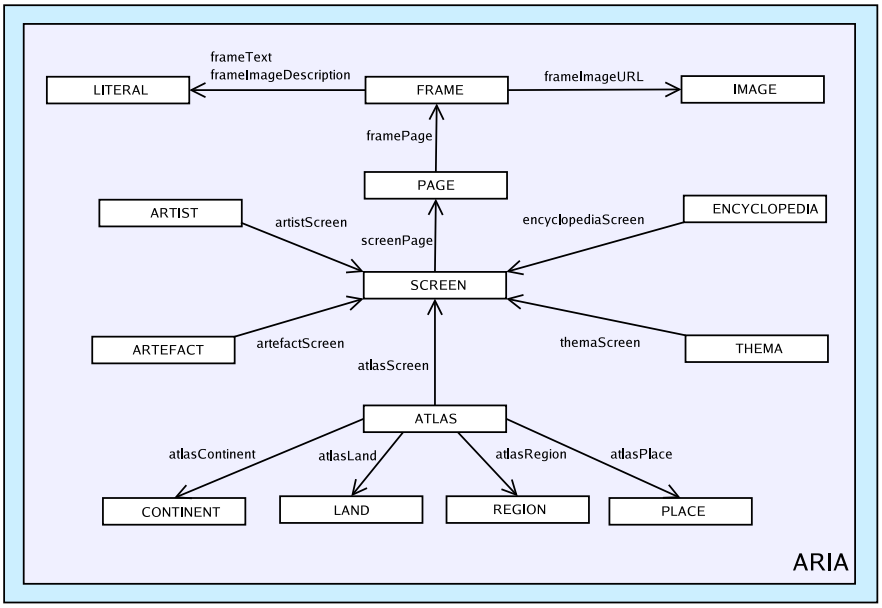

Figure 5.7: Aria Classes

different domains.

We can find different techniques to explore an ontology repository to understand its structure. Therefore, users can select the best approach for their repositories in the literature. In our case, for a deep study of the *Aria* and *Chime* structure we have chosen the combination of Sesame explore mode with Noadster system. We have made an analysis of our ontologies with the help of these tools. This analysis is not trivial. However, it is necessary in order to carry out the integration task. If the structure of both ontologies was already known, this step would not be so difficult. Figure 5.7 illustrates a global and relevant part of the classes and relationships in Aria. In the same way, figures 5.8 and 5.9 represent the *Chime* structure. Figure 5.10 shows the different concept descriptions of the term *Artwork* in each ontology. In the same way, figure 5.11 illustrates the term Artist. All these figures, which display different parts of the ontology structures, have been made after a deep analysis of all information systems involved in the integration task.

2. **Analysis of mismatches**

   Aria and *Chime* are written in the same language: RDF and RDFS. Therefore, we do not find language level mismatches when we try to integrate both ontologies. We move on classify and analyze the different types of ontology level mismatches between both ontologies.

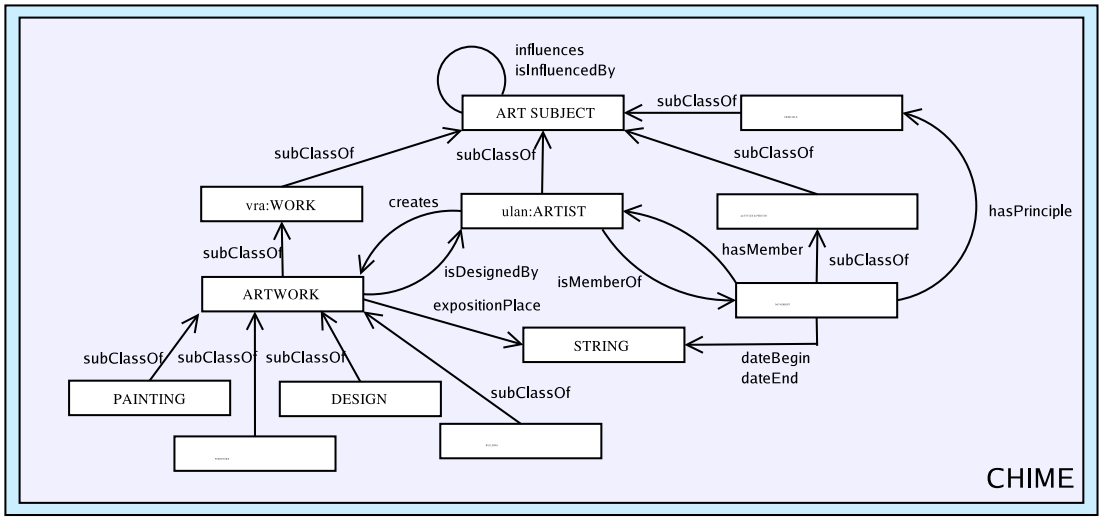

Figure 5.8: Chime Art Subject

- **Scope**

  Scope mismatch appears when two or more ontologies have two classes that apparently represent the same concept but they do not have the same instances of the concept. At first sight seems that the term *Artefact* in *Aria* is equal to the term *Artwork* in *Chime*. However, in *Chime*, the concept *Artwork* subsumes paintings, furniture, designs and buildings. In *Aria*, however, the class *Artefact* includes paintings and sculptures.

- **Model coverage and granularity**

  It is related with the level of detail to which a domain is modeled. According to the scope mismatch, *Chime* can represent furniture, designs and buildings and *Aria* cannot. This type of mismatch is very common in the integration of ontologies. We have some concepts that only appear in one ontology. For instance, we can find in *Chime* the concept *Discourse* that is used to build the storyline presentation. In Aria this term do as not appear. On the otherhand, *Aria* can represent the continent, region and place of an artefact. Whereas, *Chime* does not offer this possibility. This type of mismatches appear because Aria is a system that generates presentation structure around media objects and Chime is a system that creates a storyline presentation around different art themes. The basis is the same, *media art objects*, but the aim of each project is different.

- **Synonym terms**

  This mismatch appears when we have two or more concepts semantically equivalent which are represented by different names. We can find several synonym mismatches between both ontologies. Sometimes, inside one on-

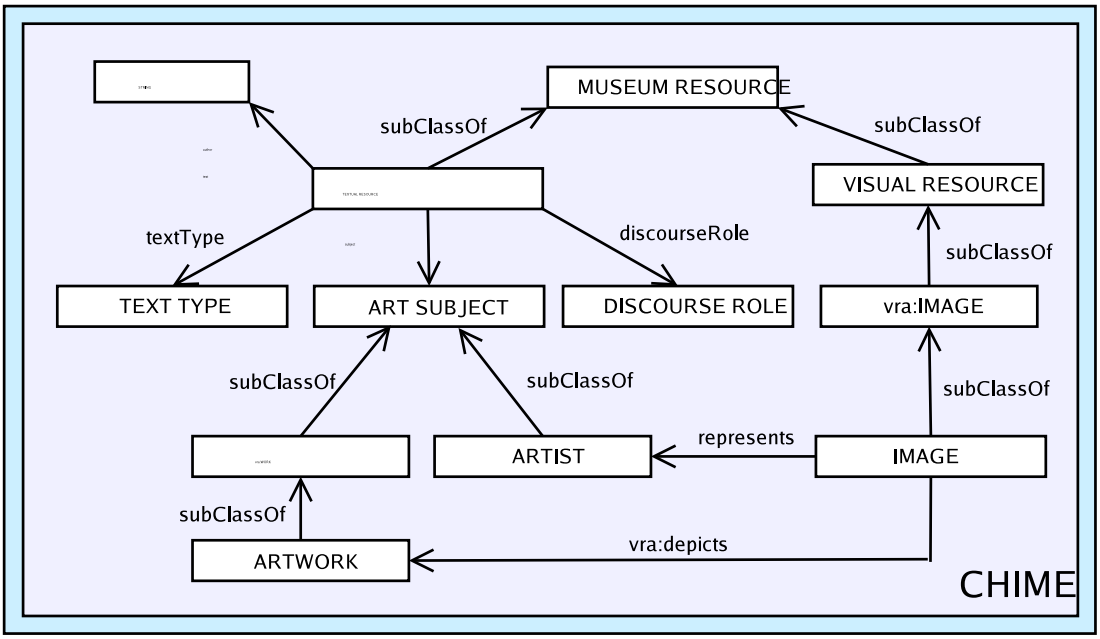

Figure 5.9: Chime Media and Text

tology there are synonym concepts such as *isDesignedBy* and *creator* or *location* and *expositionPlace*. Figure 5.12 summarises all the mismatches of this type that we find in the analysis. These concepts that both ontologies have in common will be good candidates to be a member of our shared vocabulary. We only have to search for a suitable name which will represent the concept or property in the shared vocabulary. Finally, we will have to carry out the semantic mapping between each concept or property in the information sources and the new concept in the shared vocabulary.

- **Homonym terms**

  We have two or more concepts which are not semantically equivalent but are represented by the same name. Both our ontologies have the same domain: *media art*. This type of mismatch does not occur.

- **Paradigm**

  Different paradigms can be used to represent concepts. Both ontologies share the same point of view in the art media domain. For instance, there are a set of artists who create artworks and these artworks are shown in different places. We do not find this type of mismatch in this analysis.

- **Concept description**

  This is the way in which we have modeled a concept. We would like to denote some concept description mismatches that we find in our analysis:

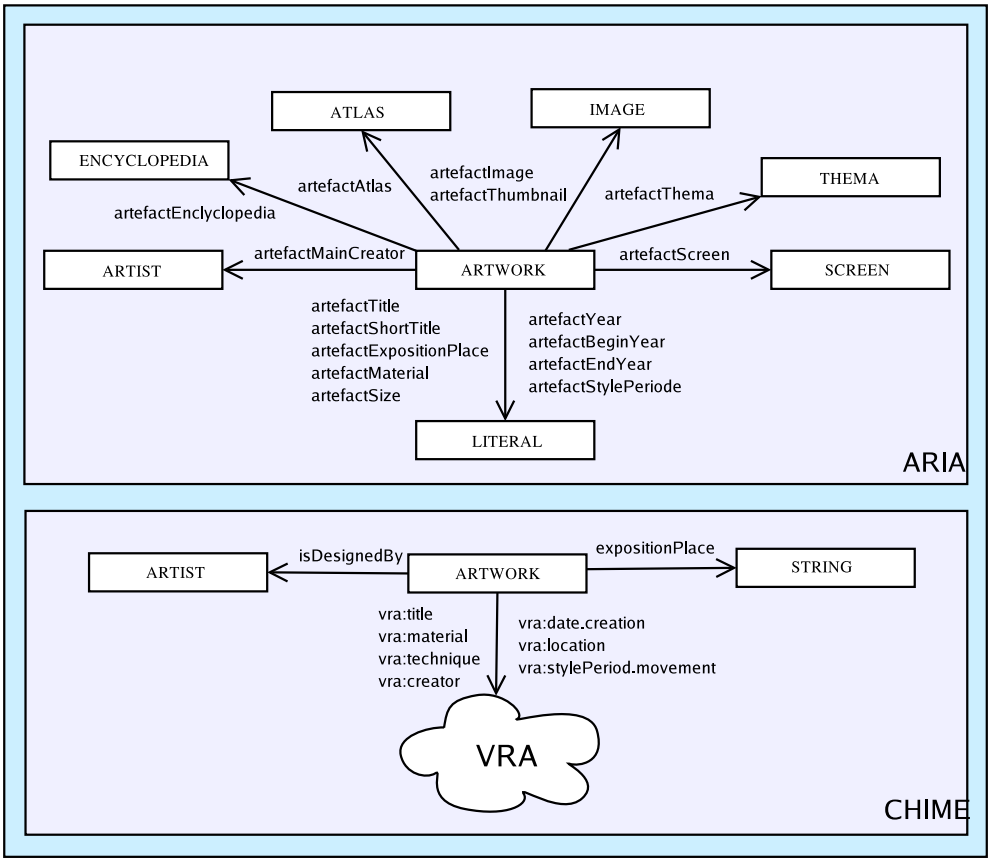

Figure 5.10: Term 'Artwork' and its properties in Aria and Chime

– **Style period movement**
Each artwork belongs to a style period movement. The term *Movement* in *Chime* is *subClassOf* the term *Style and Periods* that belong to AAT. Movement name is represented as *rdfs:label* in AAT. We cannot make a semantic connection between *rdfs:label* and the property *stylePeriodMovement* belonging to the shared vocabulary. Property *rdfs:label* is a common and general property. The effects can be dreadful if we denote it as a *rdfs:subPropertyOf* of the property *stylePeriodMovement*.
How can we achieve this translation in our integration task? Our ontologies are written in RDF/RDFS and we use SeRQL as the query language to access the information stored in the repository. One possible solution using these tools is to make use of the *construct query* belonging to the SeRQL query (see section 2.3.3). Construct queries can also be used to specify simple rules or to do graph transformations. Graph transformation is a powerful tool in application scenarios

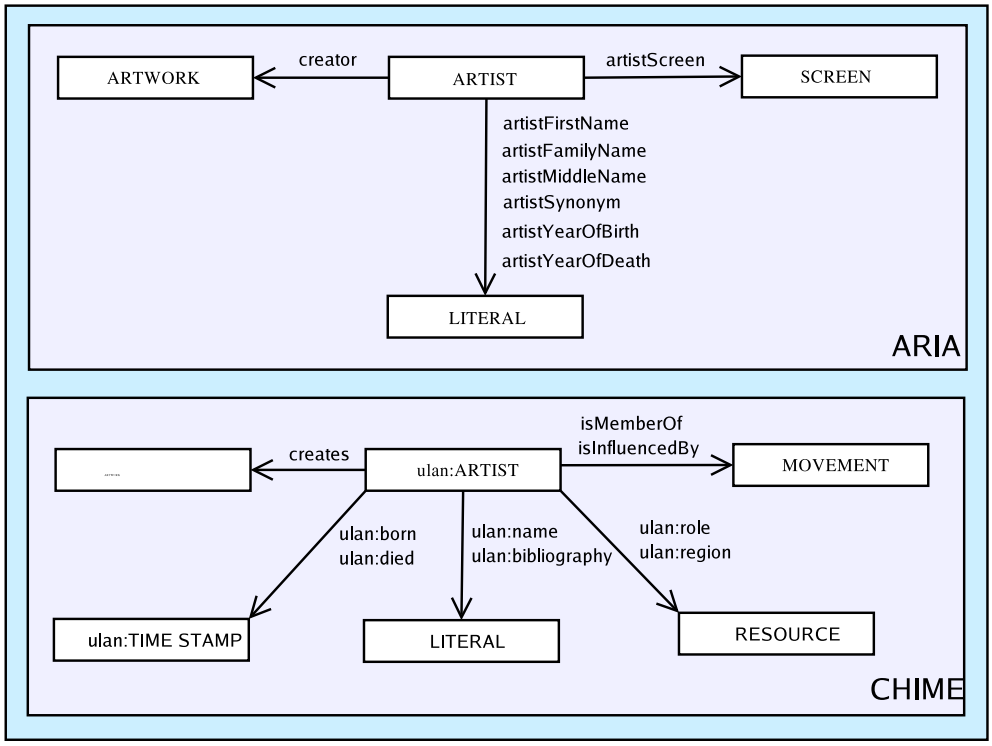

Figure 5.11: Term 'Artist' and its properties in Aria and Chime

where semantic mappings between different vocabularies need to be defined.

– **Year of birth and death**
In the same way, year of birth and death are represented by *year* property in Chime. The property *year* is also a general property. We cannot express this property as a *rdfs:subPropertyOf* of its correspondent property in the shared vocabulary. For instance, if we make a query for getting *Mondrian* year of birth, we will retrieve all years stored in the repository. We can solve this problem using a Sesame construct query.

– **Media item**
Class *Media item* does not exist in Aria. However, we can find properties as *artefactImage* or *frameImageURL* which describe images. Using Sesame construct query we can create this new class whose values are the set of URLs about images. Later, we can achieve a semantic mapping between the new class Media Item and its corresponding class in the shared vocabulary.

– **Represents**
We have defined the class Media Item. Therefore, we have to create the property *represents* between the classes *Media Item* and *Artefact*.

CHIME	ARIA	DESCRIPTION
born	artistYearOfBirth	Artist year of birth
died	artistYearOfDeath	Artist year of death
creates	creator	Artist artefacts or artworks
title	artefactTitle	Artwork name
isDesignedBy, creator	artefactMainCreator	Artefact creator
measurements	artefactSize	Artefact size
material	artefactMaterial	Artefact material
date.creation	artefactYear	Artefact year of creation
location, expositionPlace	artefactExpositionPlace	Artefact exposition place
stylePeriod.movement	artefactStylePeriode	Artefact style or period movement

Figure 5.12: Synonym mismatches between *Aria* and *Chime*

  - **isRepresentedBy**
    The Sesame construct query gives us the opportunity of creating on the inverse property. Using this technique we get more expressiveness in our shared vocabulary without much effort.

  Figure 5.13 shows all concept description mismatches that we have found in the integration task.

- **Encoding**

  Encoding appears when values in the ontologies may be encoded in different formats. We find two types of encoding mismatches. The first one is about artwork size representation. For instance, in *Chime* there are artworks measured in inches and others in centimetres.
  We also can find in both ontologies artworks modeled in three dimensions and others only in two.
  The reason for this encoding mismatch is because some paintings are in English museums and their size is represented in inches and other paintings are in Amsterdam and their size is in centimetres.

  Another mismatch is the representation of the artist names in the different ontologies. For intance, in *Aria* we have the following properties related with artist name: *artistFirstName, artistFamilyName* and *artistMiddleName*. In *Chime*, however, the value is represented as a full name with the following format: *artistFamilyName, artistFirstName*. We are going to see how to solve these mismatches in the next stage.

3. **Search for terms or primitives**

   The next step is to select the terms or primitives that are going to be a member of our shared vocabulary. The set of terms and property from the *synonym* and

Shared vocabulary term or property	Information Source	Domain	...	Description
stylePeriodMovement	Chime	Artwork	Literal	Movement name is represented as 'rdfs:label'. We cannot make a semantic connection with this generic property
yearOfBirth	Chime	Artist	Literal	Artist year of birth and death are represented by the 'year' property. We cannot make a semantic mapping with this generic property
yearOfDeath	Chime	Artist	Literal	
MediaItem	Aria	Media Item		Class 'Media Item' does not exist in Aria. However, we have a set of images and properties as 'frameImageURL' or 'artefactImage'
represents	Aria	Media Item	Artwork	We define the relationship between the new class 'Media Item' and the Artefacts that are represented by this' Media Item'
isRepresented	Aria and Chime	Art Museum Concept	Media Item	We define 'represents' inverse to get more expressiveness in our shared vocabulary
isDescribedBy	Aria	Artist	TextItem	
isDescribedBy	Chime	Artist	TextItem	
isDescribedBy	Aria	Artwork	TextItem	We relate 'Text Item' with 'Artwork'
isDescribedBy	Chime	Artwork	TextItem	
describes	Aria and Chime	TextItem	Art Museum Concept	We define 'isDescribedBy' inverse
text	Aria	TextItem	Literal	We only select text about 'Artists' and 'Artworks'
text	Chime	TextItem	Literal	

Figure 5.13: Concept description mismatches between *Aria* and *Chime*

concept description mismatches are good candidates to be a term or a properties of our shared vocabulary.

A suitable candidate for our *general bridge concept* can be a new class called *MuseumResource*. As following, we have to decide the group of terms belonging to our shared vocabulary. This task is not very difficult. We only have to think in the concepts included in the information sources domains. For instance, we clearly distinguish in both ontology domains: artists, artworks, media items and descriptions or text about these artists and artworks.

4. **Search for properties**

   The next step is to search for properties which define each term found in the previous step. For instance, figures 5.10 and 5.11 can be useful for selecting the properties about artworks and artists and the group of properties which we obtained in the analysis of *synonym* and *concept description* mismatches. We only define the properties that appear in both ontologies because we want the shared vocabulary to be as small as possible. Later, if we add new information sources, the semantic connections with the shared vocabulary will be more easy to carry out.

   Figure 5.14 illustrates the shared vocabulary structure with all its classes and properties.

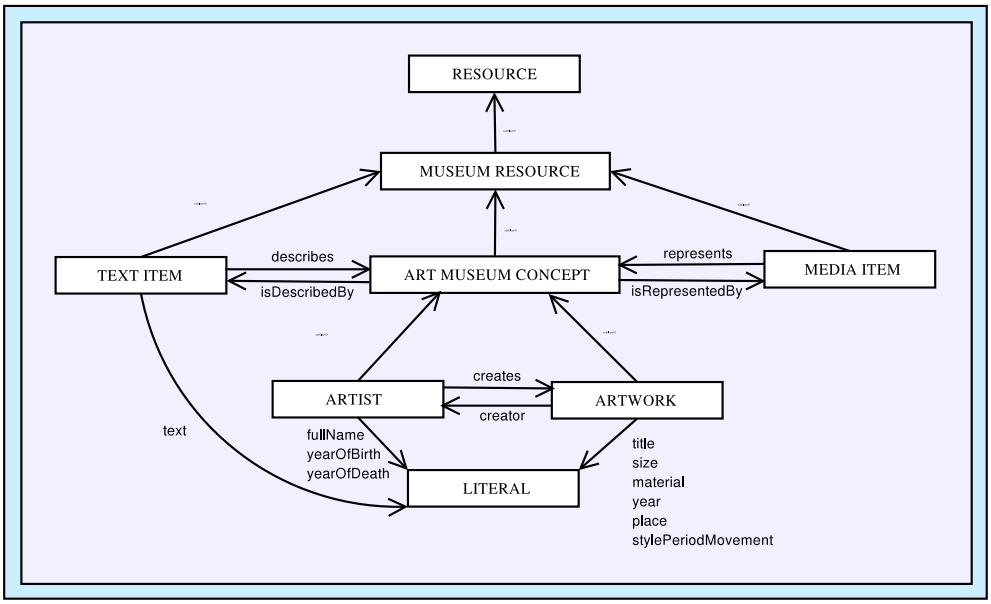

Figure 5.14: Shared Vocabulary Structure

5. **Defining the global ontology**

   Finally, we have to implement our shared ontology. In our case the implementation is in RDF(S). We have to define all concepts and properties found in the previous steps. We can use standard top level ontologies such as Dublin Core [15] in our shared vocabulary. For instance, the title of an artwork in our shared ontology is *subPropertyOf* the resource title belonging to Dublin Core. We can see the final implementation of our shared ontology structure in Appendix A. It is summarized in figure 5.14.

6. **Refinement**

   This process is not direct. When we carried it out, we had to step back many times to add, remove and modify ontology concepts, properties, etc. We do not carry out a formal evaluation of our shared vocabulary. Therefore, we have only tested our shared vocabulary doing different experiments. We have concluded that sometimes it is necessary to step back for adding, removing or modifying terms and properties. It is important to get a suitable shared vocabulary to avoid other difficulties in the posterior semantic mappings.

Building semantic mappings

The next stage is to achieve all semantic mappings for connecting our ontologies. We have to carry out four steps.

1. **Analysis of information source**

   Now, we must do the analysis between our new shared ontology and each information source without taking in account the other information sources into account. We must have our shared vocabulary as reference all the time. This new analysis help us to achieve the semantic mappings in the correct way.

2. **Solving the mismatches**

   In this step we have to resolve the different mismatches that we have found in the previous steps.

   - **Scope**

     Although the concept *Artefact* in Aria and the term *Artwork* in Chime differs, we decide to map both to the concept *Artwork* in the shared vocabulary. The only difference is that we cannot ask for buildings or furniture in Chime with the query terminology of the shared vocabulary. One possible solution is to combine in a query the syntax of the information sources with the syntax of the shared vocabulary for accessing specific information that is not mapped.

   - **Model coverage and granularity**

     The shared vocabulary only represents terms and properties which are described in both ontologies. Therefore, this mismatch is not a problem.

   - **Synonym terms**

     Synonym mismatches that appear in figure 5.12 are a member of our shared vocabulary with a different name. We only have to map the concept or property belonging to each information source with its corresponding term or property in the shared vocabulary. We have seen how to do this in an example in section 4.3.2.

   - **Concept description**

     We have explained that the way to resolve this type of mismatch is through Sesame construct queries. We can find all these queries in Appendix D.

   - **Encoding**

     There are a number of solutions for resolving encoding mismatches. One possibility to resolve the size encoding mismatch is to change all sizes and dimensions to a common one. To resolve the concatenation name encoding mismatch, we have implemented a script in Prolog which concatenates the *artistFamilyName* and *artistFirstName* in Aria in the same format as Chime.

3. **Defining the mappings**

   In this step we relate the source ontology to our shared vocabulary. Therefore, we implement all the semantic mappings in a separate file which we must add to our repository. We can see the Aria semantic mappings in Appendix B. In the same way, in Appendix C we find the Chime semantic mappings.

4. **Refinement**

   In this last step we must solve the mismatches which we cannot resolve in the previous steps.

5.5 Conclusion

In this chapter we have created a useful and practical method for the construction of a shared ontology used in hybrid approaches. The method has two main stages: *building the shared vocabulary* and *building semantic mappings*. Each stage consists of a number of steps that must be followed. We have also presented a framework to solve different ontology level mismatches. Through this framework we can explore different techniques to solve different problems.

We have also seen a real case about the integration of two different art ontologies: *Aria* and *Chime*. The goal of this chapter was to present an ontology construction method for building shared ontologies and apply this process to a real case. We have concluded that this process can become a difficult task if the size of the information sources is extensive.

Chapter 6

Conclusion

6.1 Summary

In this thesis we have seen:

- different approaches for integrating information. These approaches are: *single ontology, multiple ontology* and *hybrid ontology*.
- different types of mismatches that we can find when we integrate two or more ontologies.
- approaches and techniques for solving these mismatches.
- an ontology construction method for building shared ontologies.
- a framework for solving ontology level mismatches.
- a real case of integration of two art-media ontologies, *Aria* and *Chime*, belonging to the Rijksmuseum of Amsterdam. Into this integration we have applied our construction method to build the shared vocabulary.

6.2 Conclusion

In this thesis we focused on the integration of heterogeneous information belonging to different information sources.

The goal of this thesis was to present an ontology construction method for building shared ontologies and apply this process to a real case: the integration of two ontologies which contain images and information about the artworks in Rijksmuseum in Amsterdam. We also presented a framework for resolving different type of semantic mismatches that we can find in the integration task.

Our main experience with the shared ontology construction method shows that the process can become a difficult task if the size of the information sources is extensive.

A possible solution in this case is to divide the information sources into small modules selected by topic. After this, we have to apply our construction method to each module for making a large shared vocabulary as descriptive as possible.

Another drawback is that human intervention is required several times. However, we have seen that a construction method for a shared vocabulary is possible. This can be the basis for developing automatic tools that can help to integrate fully or partially different ontologies.

One possible application of our ontology construction method can be in a cultural heritage domain. For instance, we suppose that we are interested in the generation of a presentation about artworks placed in different museums. We will find several problems. For instance, heterogeneity problems. We also find several mismatches between the different sources. We need to know different techniques and approaches for solving these mismatches.

Furthermore, it can be interesting to apply our ontology construction method to different domains, not only cultural heritage ones. Our method is general enough to be applied to other domains.

We have learned that integrating different ontologies belonging to the same domain is not an easy task. Specifically, multimedia presentation generators use a set of media items, which are combined in a coherent presentation to the user. For this, a large amount of information about these media items and their relations is needed. When we try to integrate this amount of information we find several heterogeneity problems. We have learned through our study how to solve this problems. In other words, we have shown different techniques to carry out a integration task with success.

6.3 Evaluation

We have used the following techniques for carrying out our integration task: RDF(S) for doing the semantic connections, Sesame construct queries for resolving the more difficult mismatches and Prolog for solving encoding mismatches. We do not carry out a formal evaluation of our shared vocabulary. We have only tested the shared vocabulary doing different experiments. For instance, we have made a test implementation based on Noadster [28] where we retrieve all the artists and their artworks stored in different sources: *Aria* and *Chime*. We generate a presentation through this artist or artworks using Noadster. The goal of this application is to show how the integration of different ontologies can be used in presentation generation.

To tie together a set of component ontologies as part of a third is frequently useful to be able to indicate that a particular class or property in one ontology is equivalent to a class or property in a second ontology. OWL [3, 39] is useful for doing these connections. The OWL property *owl:equivalentClass* is used to indicate that two classes have the same instances. In the same way, the OWL property *owl:equivalentProperty* is used to indicate that two properties have the same instances. We can also use *owl:sameAs* that indicates that two URI references actually refer to the same thing. For instance,

we could state that two URI references actually refer to the same person. This property is often used for defining mappings between ontologies. We do not carry out our integration task with OWL language because Sesame does not support it. OWL allows us to we can define local scope of properties, disjointness of classes, boolean combinations of classes or cardinality restrictions. A possible upgrading of our integration task would be the addition of OWL language.

6.4 Future work

Further research is required for automating different steps belonging to our ontology construction method presented in this document. For instance, the analysis of mismatches and their resolution is a task that takes a great deal of human effort. We can include finding similar terms within two or more ontologies or for finding other types of mismatches.

Nowadays, when you are building an ontology you try to relate it with other standard shared vocabularies to make your ontologies as standard as possible. For instance, *Chime* makes semantics connections with *Ulan, AAT, Dublin Core* and *VRA Core*. Another future work can be an automatic process for building semantic mappings between terms and properties belonging to an information source and its corresponding definitions in the shared vocabulary.

Another possible way is to refine our construction process by testing it with diverse ontologies. For instance, we can apply our method to other domains.

Appendix A

Shared vocabulary in RDF(S)

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE rdf:RDF [
        <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
        <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
        <!ENTITY sha 'http://www.cwi.nl/ media/ns/sha#'>]>

<rdf:RDF xmlns:rdf="&rdf;"
            xmlns:rdfs="&rdfs;"
            xmlns:sha="&sha;">

<!– Terms –>

<rdfs:Class rdf:about="&sha;MuseumResource">
        <rdfs:label>Museum Resource</rdfs:label>
        <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>

<rdfs:Class rdf:about="&sha;ArtMuseumConcept">
        <rdfs:label>Art Museum Concept</rdfs:label>
        <rdfs:subClassOf rdf:resource="&sha;MuseumResource"/>
</rdfs:Class>

<rdfs:Class rdf:about="&sha;Artist">
        <rdfs:label>Artist</rdfs:label>
        <rdfs:subClassOf rdf:resource="&sha;ArtMuseumConcept"/>
</rdfs:Class>

<rdfs:Class rdf:about="&sha;Artwork">
        <rdfs:label>Artwork</rdfs:label>
        <rdfs:subClassOf rdf:resource="&sha;ArtMuseumConcept"/>
</rdfs:Class>

<rdfs:Class rdf:about="&sha;MediaItem">
        <rdfs:label>Media Item</rdfs:label>
        <rdfs:subClassOf rdf:resource="&sha;MuseumResource"/>
</rdfs:Class>
```

```
<rdfs:Class rdf:about="&sha;TextItem">
        <rdfs:label>Text Item</rdfs:label>
        <rdfs:subClassOf rdf:resource="&sha;MuseumResource"/>
</rdfs:Class>

<!– Artist properties –>

<rdf:Property rdf:about="&sha;fullName">
        <rdfs:label>Artist Full Name</rdfs:label>
        <rdfs:domain rdf:resource="&sha;Artist"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>

<rdf:Property rdf:about="&sha;yearOfBirth">
        <rdfs:label>Artist Year of Birth</rdfs:label>
        <rdfs:domain rdf:resource="&sha;Artist"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/date"/>
</rdf:Property>

<rdf:Property rdf:about="&sha;yearOfDeath">
        <rdfs:label>Artist Year of Death</rdfs:label>
        <rdfs:domain rdf:resource="&sha;Artist"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/date"/>
</rdf:Property>

<rdf:Property rdf:about="&sha;creates">
        <rdfs:label>Artist Creates</rdfs:label>
        <rdfs:domain rdf:resource="&sha;Artist"/>
        <rdfs:range rdf:resource="&sha;Artwork"/>
</rdf:Property>

<!– Arwork properties –>

<rdf:Property rdf:about="&sha;title">
        <rdfs:label>Title of Artwork</rdfs:label>
        <rdfs:domain rdf:resource="&sha;Artwork"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/title"/>
</rdf:Property>

<rdf:Property rdf:about="&sha;creator">
        <rdfs:label>Creator of Artwork</rdfs:label>
        <rdfs:domain rdf:resource="&sha;Artwork"/>
        <rdfs:range rdf:resource="&sha;Artist"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/creator"/>
</rdf:Property>

<rdf:Property rdf:about="&sha;size">
```

```
        <rdfs:label>Size of Artwork</rdfs:label>
        <rdfs:domain rdf:resource="&sha;Artwork"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/format"/>
</rdf:Property>

<rdf:Property rdf:about="&sha;material">
        <rdfs:label>Material of Artwork</rdfs:label>
        <rdfs:domain rdf:resource="&sha;Artwork"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/format"/>
</rdf:Property>

<rdf:Property rdf:about="&sha;year">
        <rdfs:label>Data of Artwork </rdfs:label>
        <rdfs:domain rdf:resource="&sha;Artwork"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/date"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/coverage"/>
</rdf:Property>

<rdf:Property rdf:about="&sha;place">
        <rdfs:label>Place of Artwork</rdfs:label>
        <rdfs:domain rdf:resource="&sha;Artwork"/>
        <rdfs:range rdf:resource="&sha;Museum"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/contributor"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/coverage"/>
</rdf:Property>

<rdf:Property rdf:about="&sha;stylePeriodMovement">
        <rdfs:label>Style Period Movement</rdfs:label>
        <rdfs:domain rdf:resource="&sha;Artwork"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/coverage"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/subject"/>
</rdf:Property>

<!– Media Item properties –>

<rdf:Property rdf:about="&sha;represents">
        <rdfs:label>Represents</rdfs:label>
        <rdfs:domain rdf:resource="&sha;MediaItem"/>
        <rdfs:range rdf:resource="&sha;ArtMuseumConcept"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/relation"/>
</rdf:Property>

<rdf:Property rdf:about="&sha;isRepresentedBy">
        <rdfs:label>is Represented By</rdfs:label>
        <rdfs:domain rdf:resource="&sha;ArtMuseumConcept"/>
        <rdfs:range rdf:resource="&sha;MediaItem"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/relation"/>
```

```
</rdf:Property>

<!– Text Item properties –>

<rdf:Property rdf:about="&sha;describes">
        <rdfs:label>Describes</rdfs:label>
        <rdfs:domain rdf:resource="&sha;TextItem"/>
        <rdfs:range rdf:resource="&sha;ArtMuseumConcept"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/relation"/>
</rdf:Property>

<rdf:Property rdf:about="&sha;isDescribedBy">
        <rdfs:label>is Described by</rdfs:label>
        <rdfs:domain rdf:resource="&sha;ArtMuseumConcept"/>
        <rdfs:range rdf:resource="&sha;TextItem"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/relation"/>
</rdf:Property>

<rdf:Property rdf:about="&sha;text">
        <rdfs:label>Text</rdfs:label>
        <rdfs:domain rdf:resource="&sha;TextItem"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
        <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/description"/>
        </rdf:Property>
</rdf:RDF>
```

Appendix B

Aria Semantic Mappings in RDF(S)

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE rdf:RDF [
        <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
        <!ENTITY topia 'http://www.telin.nl/rdf/topia#'>
        <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
        <!ENTITY sha 'http://www.cwi.nl/ media/ns/sha#'>]>

<rdf:RDF xmlns:rdf="&rdf;;"
            xmlns:topia="&topia;;"
            xmlns:rdfs="&rdfs;;"
            xmlns:sha="&sha;;">

<!– ARIA –>
<!– Terms or primitives –>

<rdf:Description rdf:about="&topia;Artist">
        <rdfs:subClassOf rdf:resource="&sha;Artist"/>
</rdf:Description>

<rdf:Description rdf:about="&topia;Artefact">
        <rdfs:subClassOf rdf:resource="&sha;Artwork"/>
</rdf:Description>

<rdf:Description rdf:about="&topia;artefactImage">
        <rdfs:subClassOf rdf:resource="&sha;MediaItem"/>
</rdf:Description>

<rdf:Description rdf:about="&topia;Frame">
        <rdfs:subClassOf rdf:resource="&sha;TextItem"/>
</rdf:Description>

<!– Artist properties –>

<rdf:Description rdf:about="&topia;artistFullName">
        <rdfs:subPropertyOf rdf:resource="&sha;fullName"/>
</rdf:Description>
```

```
<rdf:Description rdf:about="&topia;artistYearOfBirth">
        <rdfs:subPropertyOf rdf:resource="&sha;yearOfBirth"/>
</rdf:Description>

<rdf:Description rdf:about="&topia;artistYearOfDeath">
        <rdfs:subPropertyOf rdf:resource="&sha;yearOfDeath"/>
</rdf:Description>

<rdf:Description rdf:about="&topia;creator">
        <rdfs:subPropertyOf rdf:resource="&sha;creates"/>
</rdf:Description>

<!– Artwork properties –>

<rdf:Description rdf:about="&topia;artefactTitle">
        <rdfs:subPropertyOf rdf:resource="&sha;title"/>
</rdf:Description>

<rdf:Description rdf:about="&topia;artefactMainCreator">
        <rdfs:subPropertyOf rdf:resource="&sha;creator"/>
</rdf:Description>

<rdf:Description rdf:about="&topia;artefactSize">
        <rdfs:subPropertyOf rdf:resource="&sha;size"/>
</rdf:Description>

<rdf:Description rdf:about="&topia;artefactMaterial">
        <rdfs:subPropertyOf rdf:resource="&sha;material"/>
</rdf:Description>

<rdf:Description rdf:about="&topia;artefactYear">
        <rdfs:subPropertyOf rdf:resource="&sha;year"/>
</rdf:Description>

<rdf:Description rdf:about="&topia;artefactExpositionPlace">
        <rdfs:subPropertyOf rdf:resource="&sha;place"/>
</rdf:Description>

<rdf:Description rdf:about="&topia;artefactStylePeriode">
        <rdfs:subPropertyOf rdf:resource="&sha;stylePeriodMovement"/>
</rdf:Description>

<!– Media Item properties –>

<rdf:Description rdf:about="&topia;artefactImage">
        <rdfs:subPropertyOf rdf:resource="&sha;represents"/>
        <rdfs:domain rdf:resource="&sha;MediaItem"/>
</rdf:Description>

</rdf:RDF>
```

Appendix C

Chime Semantic Mappings in RDF(S)

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE rdf:RDF [
        <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
        <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
        <!ENTITY sha 'http://www.cwi.nl/ media/ns/sha#'>
        <!ENTITY art 'http://www.cs.vu.nl/bi/ns/art#'>
        <!ENTITY museum 'http://www.cs.vu.nl/bi/ns/museum#'>
        <!ENTITY vra 'http://www.swi.psy.uva.nl/mia/vra#'>
        <!ENTITY aat 'http://www.swi.psy.uva.nl/mia/aat#'>
        <!ENTITY ulan 'http://www.swi.psy.uva.nl/mia/ulan#'>
        <!ENTITY txt 'http://www.cs.vu.nl/bi/ns/text#'>]>

<rdf:RDF xmlns:rdf="&rdf;"
            xmlns:rdfs="&rdfs;"
            xmlns:sha="&sha;"
            xmlns:vra="&vra;"
            xmlns:art="&art;"
            xmlns:museum="&museum;"
            xmlns:aat="&aat;"
            xmlns:ulan="&ulan;"
            xmlns:txt="&txt;">

<!– CHIME2–>

<!– Terms or primitives –>

<rdf:Description rdf:about="&art;Artist">
        <rdfs:subClassOf rdf:resource="&sha;Artist"/>
</rdf:Description>

<rdf:Description rdf:about="&art;Artwork">
        <rdfs:subClassOf rdf:resource="&sha;Artwork"/>
</rdf:Description>

<rdf:Description rdf:about="&museum;VisualResource">
        <rdfs:subClassOf rdf:resource="&sha;MediaItem"/>
```

```
</rdf:Description>

<rdf:Description rdf:about="&txt;TextualResource">
        <rdfs:subClassOf rdf:resource="&sha;TextItem"/>
</rdf:Description>

<!– Artist properties –>

<rdf:Description rdf:about="&ulan;name">
        <rdfs:subPropertyOf rdf:resource="&sha;fullName"/>
</rdf:Description>

<rdf:Description rdf:about="&art;creates">
        <rdfs:subPropertyOf rdf:resource="&sha;creates"/>
</rdf:Description>

<!– Artwork properties –>

<rdf:Description rdf:about="&vra;title">
        <rdfs:subPropertyOf rdf:resource="&sha;title"/>
</rdf:Description>

<rdf:Description rdf:about="&vra;creator">
        <rdfs:subPropertyOf rdf:resource="&sha;creator"/>
</rdf:Description>

<rdf:Description rdf:about="&art;isDesignedBy">
        <rdfs:subPropertyOf rdf:resource="&sha;creator"/>
</rdf:Description>

<rdf:Description rdf:about="&vra;measurements">
        <rdfs:subPropertyOf rdf:resource="&sha;size"/>
</rdf:Description>

<rdf:Description rdf:about="&vra;material">
        <rdfs:subPropertyOf rdf:resource="&sha;material"/>
</rdf:Description>

<rdf:Description rdf:about="&vra;date.creation">
        <rdfs:subPropertyOf rdf:resource="&sha;year"/>
</rdf:Description>

<rdf:Description rdf:about="&art;expositionPlace">
        <rdfs:subPropertyOf rdf:resource="&sha;place"/>
</rdf:Description>

<rdf:Description rdf:about="&vra;location">
        <rdfs:subPropertyOf rdf:resource="&sha;place"/>
</rdf:Description>

<!– Media Item properties –>
```

```
<rdf:Description rdf:about="&museum;represents">
        <rdfs:subPropertyOf rdf:resource="&sha;represents"/>
</rdf:Description>

<rdf:Description rdf:about="&vra;depicts">
        <rdfs:subPropertyOf rdf:resource="&sha;represents"/>
</rdf:Description>

</rdf:RDF>
```

Appendix D

Sesame construct queries

```
— STYLE PERIOD MOVEMENT (CHIME) —

construct distinct {Artwork} sha:stylePeriodMovement {stylePeriodMovement}
from {Artwork} rdf:type {art:Artwork};
                    vra:stylePeriod.movement {movementName},
       {movementName} rdfs:label {stylePeriodMovement}
using namespace
    sha = <http://www.cwi.nl/ media/ns/sha#>,
    art = <http://www.cs.vu.nl/bi/ns/art#>,
    vra = <http://www.swi.psy.uva.nl/mia/vra#>

— YEAR OF BIRTH (CHIME) —

construct distinct {Artist} sha:yearOfBirth {birthYear}
from {Artist} rdf:type {art:Artist};
                ulan:born {yearOfBirth},
       {yearOfBirth} ulan:year {birthYear}
using namespace
    sha = <http://www.cwi.nl/ media/ns/sha#>,
    art = <http://www.cs.vu.nl/bi/ns/art#>,
    ulan = <http://www.swi.psy.uva.nl/mia/ulan#>

— YEAR OF DEAT (CHIME) —

construct distinct {Artist} sha:yearOfDeath {deathYear}
from {Artist} rdf:type {art:Artist};
                ulan:died {yearOfDeath},
       {yearOfDeath} ulan:year {deathYear}
using namespace
    sha = <http://www.cwi.nl/ media/ns/sha#>,
    art = <http://www.cs.vu.nl/bi/ns/art#>,
    ulan = <http://www.swi.psy.uva.nl/mia/ulan#>

— MEDIA ITEM (ARIA) —

construct distinct {MediaItem} rdf:type {sha:MediaItem}
```

```
from {} rdf:type {topia:Artefact};
        topia:artefactImage {MediaItem}
using namespace
   sha = <http://www.cwi.nl/ media/ns/sha#>,
   topia = <http://www.telin.nl/rdf/topia#>

— REPRESENTS (ARIA) —

construct distinct {MediaItem} sha:represents {Artwork}
from {Artwork} rdf:type {topia:Artefact};
                topia:artefactImage {MediaItem}
using namespace
   sha = <http://www.cwi.nl/ media/ns/sha#>,
   topia = <http://www.telin.nl/rdf/topia#>

— IS REPRESENTED BY ('REPRESENTS' INVERSE) —
IMPORTANT: YOU MUST ADD THE RESULTS OF CONSTRUCT 5

construct distinct {ArtMuseumConcept} sha:isRepresentedBy {MediaItem}
from {MediaItem} rdf:type {sha:MediaItem};
                    sha:represents {ArtMuseumConcept}
using namespace
   sha = <http://www.cwi.nl/ media/ns/sha#>

— IS DESCRIBED BY (CHIME. ARTIST) —

construct distinct {Artist} sha:isDescribedBy {TextItem}
from {TextItem} rdf:type {sha:TextItem};
                txt:subject {Artist},
      {Artist} rdf:type {sha:Artist}
using namespace
   sha = <http://www.cwi.nl/ media/ns/sha#>,
   txt = <http://www.cs.vu.nl/bi/ns/text#>

— IS DESCRIBED BY (CHIME. ARTWORK) —

construct distinct {Artwork} sha:isDescribedBy {TextItem}
from {TextItem} rdf:type {sha:TextItem};
                txt:subject {Artwork},
      {Artwork} rdf:type {sha:Artwork}
using namespace
   sha = <http://www.cwi.nl/ media/ns/sha#>,
   txt = <http://www.cs.vu.nl/bi/ns/text#>

— IS DESCRIBED BY (ARIA. ARTIST) —

construct distinct {Artist} sha:isDescribedBy {TextItem}
from {Artist} rdf:type {topia:Artist};
              topia:artistScreen {artistScreens},
      {artistScreens} rdf:type {topia:Screen};
                      topia:screenPage {artistPages},
```

```
      {artistPages} rdf:type {topia:Page};
                    topia:pageFrame {TextItem}
using namespace
   topia = <http://www.telin.nl/rdf/topia#>,
   sha = <http://www.cwi.nl/ media/ns/sha#>

— IS DESCRIBED BY (CHIME. ARTWORK) —

construct {Artwork} sha:isDescribedBy {TextItem}
from {Artwork} rdf:type {topia:Artefact};
               topia:artefactScreen {artworkScreens},
      {artworkScreens} rdf:type {topia:Screen};
                       topia:screenPage {artworkPages},
      {artworkPages} rdf:type {topia:Page};
                     topia:pageFrame {TextItem}
using namespace
   topia = <http://www.telin.nl/rdf/topia#>,
   sha = <http://www.cwi.nl/ media/ns/sha#>

— DESCRIBES ('IS DESCRIBE BY' INVERSE) —
IMPORTANT: YOU MUST ADD BEFORE THE RDF RESULTS FROM CONSTRUCTS 7,8,9 AND 10

construct distinct {TextItem} sha:describes {ArtMuseumConcept}
from {ArtMuseumConcept} rdf:type {sha:ArtMuseumConcept};
                        sha:isDescribedBy {TextItem}
using namespace
   sha = <http://www.cwi.nl/ media/ns/sha#>

— TEXT (CHIME) —
IMPORTANT: YOU MUST ADD BEFORE THE RDF RESULTS FROM CONSTRUCT 11

construct distinct {TextItem} sha:text {text}
from {TextItem} rdf:type {sha:TextItem};
                sha:describes {ArtMuseumConcept};
                txt:text {text}
using namespace
   sha = <http://www.cwi.nl/ media/ns/sha#>,
   txt = <http://www.cs.vu.nl/bi/ns/text#>

— TEXT (ARIA) —
IMPORTANT: YOU MUST ADD BEFORE THE RDF RESULTS FROM CONSTRUCT 11

construct distinct {TextItem} sha:text {text}
from {TextItem} rdf:type {sha:TextItem};
                sha:describes {ArtMuseumConcept};
                topia:frameText {text}
using namespace
   sha = <http://www.cwi.nl/ media/ns/sha#>,
   topia = <http://www.telin.nl/rdf/topia#>
```

Bibliography

[1] Aidministrator Nederland B.V. Sesame, 2002.

[2] Aidministrator Nederland B.V. *SeRQL user manual*, April 4, 2003.

[3] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer*. The MIT Press Cambridge, Massachusetts, London England, 2004.

[4] ARIA. http://topia.demo.telin.nl.

[5] Tim Berners-Lee. *Weaving the Web*. Orion Business, 1999.

[6] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific Amercan*, 284(5):35–43, 2001.

[7] S. Bowers and L. Delcambre. Representing and transforming model-based information. *In Proceedings of the FirstWorkshop on the Semantic Web at the Fourth European Conference on Digital Libraries, Lisbon, Portugal*, 2000.

[8] J. Broekstra and Kampman A. Serql: An rdf query and transformation language. *Submitted to the International Semantic Web Conference, ISWC 2004*, August 2004. Available at http://www.cs.vu.nl/˜jbroeks/papers/SeRQL.pdf.

[9] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: An Architecture for Storing and Querying RDF Data and Schema Information. In D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, editors, *Semantics for the WWW*. MIT Press, 2001.

[10] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In Ian Horrocks and Jim Hendler, editors, *The Semantic Web - ISWC 2002*, number 2342 in Lecture Notes in Computer Science, pages 54–68, Berlin Heidelberg, 2002. Springer.

[11] Agustina Buccella and Alejandra Cechich. An ontology approach to data integration. *JCS&T*, Vol. 3 (No.2), October2003.

[12] Hans Chalupsky. OntoMorph: A translation system for symbolic logic. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *KR2000: Principles of Knowledge Representation and Reasoning*, pages 471–482, San Francisco, March 2000. Morgan Kaufmann.

[13] V. Chaudhri, A. Farquhar, R. Fikes, P. Karp, and J. Rice. Okbc: A programmatic foundation for knowledge base interoperability. *In Proceedings of AAAI-98*, pages 600–607, 1998.

[14] CHIME. http://homepages.cwi.nl/media/projects/chime/index.html.

[15] Dublin Core Community. Dublin Core Element Set, Version 1.1, 2003. ISO Standard 15836-2003 (February 2003), http://www.niso.org/international/SC4/n515.pdf; NISO Standard Z39.85-2001 (September 2001), http://www.niso.org/standards/resources/Z39-85.pdf; CEN Workshop Agreement CWA 13874 (March 2000), http://www.cenorm.be/isss/cwa_download_area/cwa13874.pdf.

[16] Kateryna Falkovych and Frank Nack. Context aware guidance for multimedia authoring: harmonizing domain and discourse knowledge. *INS-E0502*, 2005.

[17] Kateryna Falkovych, Frank Nack, Jacco van Ossenbruggen, and Lloyd Rutledge. SampLe: Towards a Framework for System-supported Multimedia Authoring. Technical Report INS-E0302, CWI, August 2003.

[18] D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster. Spinning the semantic web. *MIT Press*, pages 1–8, 2003.

[19] Dieter Fensel, Michael Erdmann, and Rudi Studer. Ontology Groups: Semantically Enriched Subnets of the WWW. In *Proceedings of the International Workshop Intelligent Information Integration during the 21st German Annual Conference on Artificial Intelligence*, Freiburg, Germany, September 9-12, 1997.

[20] Getty Research Institute. Art & Architecture Thesaurus (Online). http://www.getty.edu/research/tools/vocabulary/aat/, 2000. Version 2.0.

[21] Joost Geurts, Stefano Bocconi, Jacco van Ossenbruggen, and Lynda Hardman. Towards Ontology-driven Discourse: From Semantic Graphs to Multimedia Presentations. Technical Report INS-R0305, CWI, May 2003.

[22] A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt, and M. Uschold. Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing. Technical Report 47, CEUR workshop proceedings, Seattle, USA, August 2001.

[23] T.R. Gruber. A Translation Approach Portable Ontology Specification. *Knowledge Acquisition*, 1993.

[24] S. Melnik and S. Decker. A layered approach to information modeling and interoperability on the web. *In Proceedings of the ECDL 2000 Workshop on the Semantic Web, Lisbon, Portugal*, 2000.

[25] Natalya F. Noy. Semantic integration: a survey of ontology-based approaches. *SIGMOD Rec.*, 33(4):65–70, December 2004.

[26] Dennis Quan and David R. Karger. How to Make a Semantic Web Browser. In *The Thirteenth International World Wide Web Conference*, New York City, May 17-22, 2004. IW3C2, ACM Press.

[27] Lloyd Rutledge, Martin Alberink, Rogier Brussee, Stanislav Pokraev, William van Dieten, and Mettina Veenstra. Finding the Story — Broader Applicability of Semantics and Discourse for Hypermedia Generation. In *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia*, pages 67–76, Nottingham, UK, August 23-27, 2003. ACM.

[28] Lloyd Rutledge, Jacco van Ossenbruggen, and Lynda Hardman. Making RDF Presentable – Integrated Global and Local Semantic Web Browsing. In *The Fourteenth International World Wide Web Conference*, pages 199–206, Chiba, Japan, May 2005. IW3C2, ACM Press.

[29] SeRQL. Available at http://www.openrdf.org/doc/users/ch06.html.

[30] Sesame. http://sesame.aduna.biz/sesame.

[31] SPARQL. Available at http://www.w3.org/tr/2005/wd-rdf-sparql-query-20050419/.

[32] Heiner Stuckenschmidt. *Ontology-Based Information Sharing in Weakly Structured Environments*. PhD thesis, Vrije Universiteit, Amsterdam, January 23, 2003.

[33] ULAN. http://www.getty.edu/research/conducting_research/vocabularies/ulan/.

[34] Mark van Assem, Maarten R. Menken, Guus Schreiber, Jan Wielemaker, and Bob Wielinga. A method for converting thesauri to rdf/owl. *Proceedings of the 3rd International Semantic Web Conference (ISWC'04), Hiroshima, Japan*, 2004. Available at http://thesauri.cs.vu.nl, accompanying this paper.

[35] R. Vdovjak, F. Frasincar, G.J. Houben, and P. Barna. Engineering Semantic Web Information Systems in Hera. *Journal of Web Engineering*, 2(1 and 2):3–26, 2003.

[36] Visual Resources Association. Visual Resources Association Website.

[37] W3C. Semantic Web Activity.

[38] W3C. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendations are available at http://www.w3.org/TR, February 22, 1999. Edited by Ora Lassila and Ralph R. Swick.

[39] W3C. Web Ontology Language (OWL) Reference Version 1.0. Work in progress. W3C Working Drafts are available at http://www.w3.org/TR, 12 November 2002. Edited by Mike Dean and Dan Connolly and Frank van Harmelen and James Hendler and Ian Horrocks and Deborah L. McGuinness and Peter F. Patel-Schneider Lynn Andrea Stein.

[40] W3C. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, 10 February 2004. Edited by Dan Brickley and R.V. Guha.

[41] H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, and H. Neumannand S.Huebner. Ontology-based integration of information - a survey of existing approaches. In *Ontologies and Information Sharing* [22], pages 108–117.