Building an integrated Ontology within SEWASIE system*

Domenico Beneventano[1,2], Sonia Bergamaschi[1,2], Francesco Guerra[1], and Maurizio Vincini[1]

[1] Dipartimento di Ingegneria dell'Informazione
Università di Modena e Reggio Emilia
Via Vignolese 905 - Modena
{lastname.firstname}@unimo.it
[2] IEIIT-CNR Istituto di Elettronica e di Ingegneria
dell'Informazione e delle Telecomunicazioni
Viale Risorgimento 2 - Bologna

Abstract. The SEWASIE (SEmantic Webs and AgentS in Integrated Economies) project (IST-2001-34825) is an European research project that aims at designing and implementing an advanced search engine enabling intelligent access to heterogeneous data sources on the web.
In this paper we focus on the Ontology Builder component of the SEWASIE system, that is a framework for information extraction and integration of heterogeneous structured and semi-structured information sources, built upon the MOMIS (Mediator envirOnment for Multiple Information Sources) system. The result of the integration process is a Global Virtual View (in short GVV) which is a set of (global) classes that represent the information contained in the sources being used. In particular, we present the application of our integration concerning a specific type of source (i.e. web documents), and show the extension of a built-up GVV by the addition of another source.

Introduction

Nowadays the Web is a huge collection of data and its expansion rate is very high. Web users need new ways to exploit all this available information and possibilities. The problem is that Web information is meaningless for a computer and so it is very hard to find out what we are looking for. In this context, the need of *a new vision of the Web*, the **Semantic Web**[3], arises. Within the Semantic Web, resources could be annotated with machine-processable metadata providing them with background knowledge and meaning. This new scenario creates many expectations amongst the users and information providers but new issues have to be solved before achieving optimum results. One of the main components in this context is is the **ontology**; this "*explicit specification of a*

* This research has been partially supported by EU IST-SEWASIE
[3] http://www.w3.org/2001/sw/

conceptualization"[11] might allow information providers to give a shared meaning to their documents. Many studies are defining *languages* and *standards* that can help domain experts in the delicate task of expressing their knowledge in a formal way[4]. Another fundamental issue is the "*dynamics*". The web environment is very changeable, it is continuously updated, modified and the users need to rely on the data they retrieve from the net. Ontologies evolve, and therefore we have to address the problem of managing the dynamics with respect to the ontologies [13, 14].

SEWASIE (SEmantic Webs and AgentS in Integrated Economies) (IST-2001-34825) is a research project funded by EU on the action line "Semantic Web" (May 2002/April 2005 - http://www.sewasie.org/). The goal of the SEWASIE project is to design and implement an advanced search engine enabling intelligent access to heterogeneous data sources on the web via semantic enrichment to provide the basis of structured secure web-based communication. A SEWASIE user has at his disposal a search client with an easy-to-use query interface able to extract the required information from the Internet and to show it in an easily readable format.

In this paper we focus on the Ontology Builder component of the SEWASIE system, that is a framework for information extraction and integration of heterogeneous structured and semi-structured information sources, built upon the MOMIS (Mediator envirOnment for Multiple Information Sources) system [1, 2, 6].

The Ontology Builder implements a semi-automatic methodology for data integration that follows the *Global as View* (GAV) approach [15]. The result of the integration process is a global schema which provides a reconciled, integrated and virtual view of the underlying sources, the GVV (Global Virtual View). The GVV is composed of a set of (global) classes that represent the information contained in the sources being used and the mappings establishing the connection between the elements of the global schema and those of the source schemata. A GVV, thus, may be thought of as a domain ontology [12] for the integrated sources. We represent the ontology by means an object language, called ODL_{I^3}, which is an evolution of the OODBMS standard language ODL. Moreover, ODL_{I^3} permits the definition of integrity constraints (in the form of *if then* rules) that are translated, together with the schema properties, into a description logics OLCD (Object Language with Complements allowing Descriptive cycles) [4, 6]. In this way, inference tasks typical of Description Logics that are useful for the GVV creation process can be exploited. The Ontology Builder system relies on a logic layer, ODL_{I^3} is the language to represent the ontology properties and OLCD to perform reasoning over the data, like other approaches in the literature (DAML+OIL[5]).

The outline of the paper is the following: section 1 describes the SEWASIE architecture, while in section 2 we depict the Ontology Builder and the approach

[4] OntoWeb - Ontology-based information exchange for knowledge management and electronic commerce, http://ontoweb.aifb.uni-karlsruhe.de/

[5] http://www.w3.org/TR/daml+oil-reference

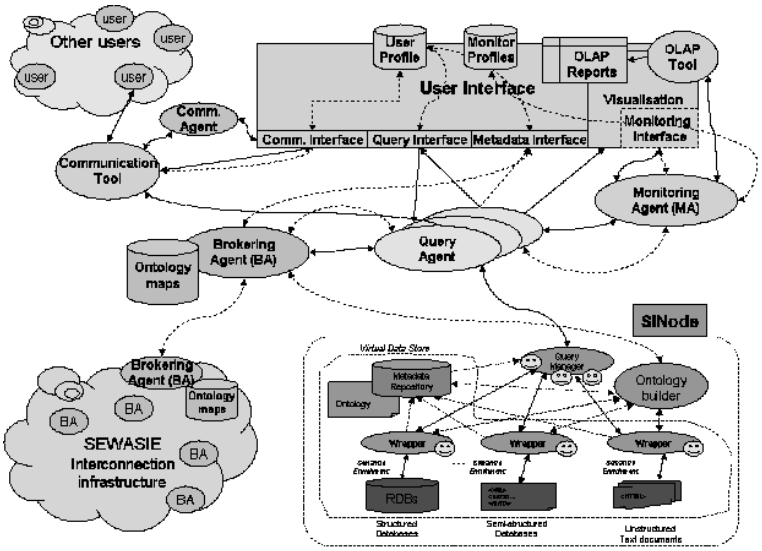

Fig. 1. The SEWASIE architecture

for creating a domain ontology from scratch and shows the result of the integration process (GVV). Section 3 describes the semi-automatic annotation process of the GVV. Section 4 presents the methodology to support the GVV extensions in the case of the addition of a new source. Finally, section 5 concludes the paper.

1 The SEWASIE Architecture

The first basic idea underlying the SEWASIE architecture is that *semantic enrichment of data sources is the next step towards building information systems that are really useful*. However, the addition of semantics to data sources is a formidable task and it may be achieved only if info seekers and info providers may reach each other across a middle ground. This requires a *common* language and strategy, and the tools that actually flesh them both out.

The second idea is that *we have to deal with two levels of knowledge*. We envision a multi-level architecture, composed of nodes (the SINodes) integrating information coming from communities with strong ties, and at a wider level the relationships among distinct SINodes are established by means of weaker semantics mappings. The latter is maintained by an infrastructure of *brokers*, which will provide the entry points to the system and some routing of the queries towards the relevant information nodes.

A search system architecture satisfying the aforementioned ideas and desiderata is shown in figure 1.

The **information nodes** (SINodes) are mediator-based systems, each including a Virtual Data Store, an Ontology Builder, and a Query Manager. A Virtual Data Store represents a virtual view of the overall information managed within any SINode and consists of the managed information sources, wrappers, and a metadata repository. The managed Information Sources are heterogeneous collections of structured, semi-structured, or unstructured data, e.g. relational databases, XML or HTML documents. A Wrapper implements common communication protocols and translates to and from local access languages. According to the metadata provided by the wrappers, the Ontology Builder performs semantic enrichment processes in order to create and maintain the Ontology of the SINode. The Metadata Repository holds the ontology and the knowledge required to establish semantic inter-relationships between the SINode itself and the neighboring ones. A Query Manager provides the functionalities for solving a query within an SINode and constitutes the SINode interface to the network.

The **brokering agents (BAs)** are the peers responsible for maintaining a view of the knowledge handled by the network, as well as the information on the specific content of SINodes which are under direct control (of each brokering agent). These agents are intermediaries which have direct control over a number of SINodes, and provide the means to publish a manifesto within the network of the locally held information with a semantic profile.

The **query agents (QAs)** are the carriers of the user query from the user interface to the SINodes, and have the task of solving a query by interacting with the brokering agent network. Starting from a user- or task- specified brokering agent, they may access other BAs, connect with other information nodes, collect partial answers, and integrate them.

The **user interface** is the group of modules which work together to offer an integrated user interaction with the semantic search system. This interface needs to be personalized and configured with the specific user profile and a reference to the ontologies which are commonly used by this user.

2 The Ontology Builder

The process of semantic enrichment of the sources constituting a SINode is a crucial step towards building the overall SEWASIE structure. The process is human assisted and based on a tool, the Ontology Builder. The underlying strategy and framework are based on ODL_{I^3}, the ontology description language, and basic lexical ontologies to bootstrap. The final result is a Global Virtual View encompassing all the sources within the SINode.

In this section, we describe the information integration process for building the GVV of set of web pages (see Figure2 for the whole process representation).

2.1 ODL_{I^3} + OLCD

For a semantically rich representation of source schemas and object patterns, the Ontology Builder uses an object-oriented language called ODL_{I^3}[6]. ODL_{I^3} is

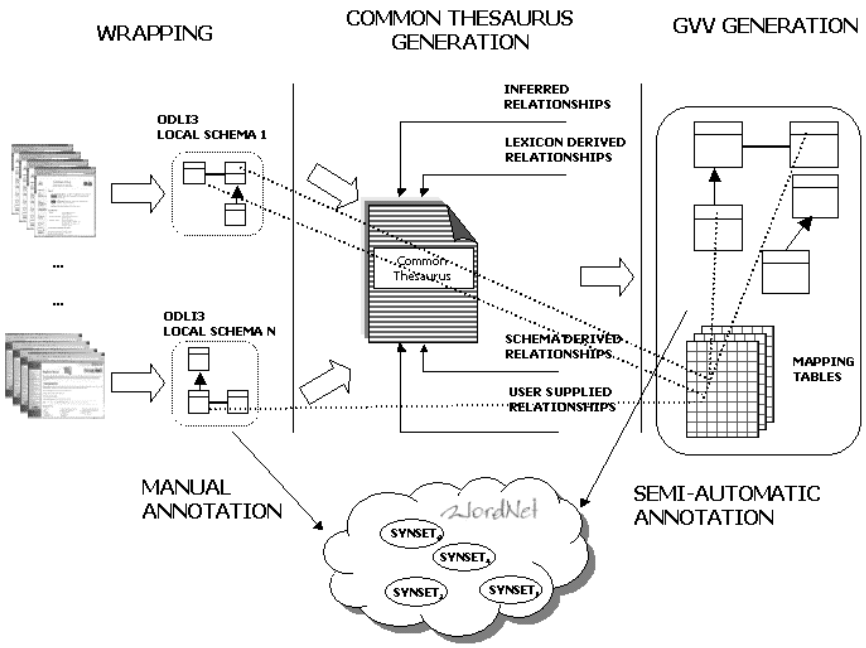

Fig. 2. An overview of the ontology integration process

an extension of the ODL language[6] and can be used to describe heterogeneous schemas of structured and semistructured data sources. In particular, ODL_{I^3} extends ODL with the following relationships expressing intra- and inter-schema knowledge for the source schemas:

- SYN (synonym of) is a relationship defined between two terms t_i and t_j that are synonyms in every involved source.
- BT (broader terms) is a relationship defined between two terms t_i and t_j, where ti has a broader, more general meaning than t_j. BT relationships are not symmetric. The opposite of BT is NT (narrower terms).
- RT (related terms) is a relationship defined between two terms t_i and t_j that are generally used together in the same context in the considered sources.

Other main additions are the *Integrity constraint rules*, introduced in ODL_{I^3} in order to express, in a declarative way, *if then* constraint rules at both intra- and inter-source level.

By means of ODL_{I^3} it is possible to describe both the sources (the input of the synthesis process) and the GVV (the result of the process) by using the same language.

[6] http://www.service-architecture.com/database/articles/odmg_3_0.html

Due to the fact that the ontology is composed of concepts (represented in ODL_{I^3} with Global Classes) and simple binary relationships, the translation of ODL_{I^3} descriptions into one of the Semantic Web standards such as RDF, DAML+OIL, OWL is a straightforward process. In fact, from a general perspective an ODL_{I^3} concept corresponds to a *Class* of a the Semantic Web standard, and ODL_{I^3} relationships are translated into *properties* (in particular the BT/NT ODL_{I^3} relationships are *subclassof* in the Semantic Web standards). Analyzing syntax and semantics of each standard, further specific correspondences may be established. For example, there is the correspondence with the DAML+OIL *Class*, the simple domain attributes correspond to DAML+OIL *DataTypeProperty* concept and complex domain attributes correspond to DAML+OIL *ObjectProperty* concept. Moreover, classes are wrapped in both the approches into description logics. For a more detailed description of ODL_{I^3} /OLCD translation see [6]. For a description of the OLCD description logics see [4, 3]

2.2 Wrapping: extracting data structure for sources

The first step of the ontology development process is the construction of a semantic representation of the information sources, i.e. the conceptual schema of the sources, by means of the common data language ODL_{I^3}. To accomplish this task, we encapsulate each source with a wrapper that logically converts the underlying data structure into the ODL_{I^3} information model. Therefore, the wrapper architecture and interfaces are crucial, because wrappers are the focal point for managing the diversity of data sources.

For conventional structured information sources (e.g. relational databases, object-oriented databases), a schema description is always available and can be directly translated.

For semistructured information sources, a schema description is in general not directly available at the sources. In fact, a basic characteristic of semistructured data is that they are "self-describing", hence the information associated with the schema is specified within data. Thus, in order to manage a semi-structured source a specific wrapper has to implement a (semi-) automatic methodology to extract and explicitly represent the conceptual schema of the source. We developed a wrapper for XML/DTDs files.

Information is available on the Web mainly in HTML format that is human-readable but cannot easily be automatically accessed and manipulated. In particular, HTML language does not separate data structure from layout. Thus, in order to manage these kind of sources, we need a further preliminary step of extraction: by means of a commercial tool we translate the content of a web page (data and data structure) into a XML file, then we exploit the previously developed wrapper XML/DTD to acquire the source descriptions.

We have tested many research and commercial tools, such as Lixto [10], RoadRunner [8], Andes [16], and we select Lixto as the most suitable for our approach. By providing a fully visual and interactive user interface, Lixto assists the user to create a wrapper program in a semi-automatic way. Once the wrapper is built, it can be applied automatically to continually extract relevant

information from a permanently changing web page and translate it into a XML file to be exploited by the XML/DTD wrapper.

2.3 Running example

We consider the creation of an ontology of two web sources related to the University domain. By means of a Lixto generated wrapper, the source content is translated into XML files according to the DTDs sketched in Table 1.

University Site (UNI)	Computer Science Site (CS)
<!ELEMENT UNI(People*)> <!ELEMENT People(Research_Staff*| School_Member*)> ... <!ELEMENT Research_Staff(name, e-mail, Section*, Article*)> <!ELEMENT Section(name, year. period)> <!ELEMENT Article(title, year, journal, conference)> <!ELEMENT School_Member(name, e-mail)> <!ELEMENT name (#pcdata)> ...	<!ELEMENT CS(Person*)> ... <!ELEMENT Person(Professor*| Student*)> <!ELEMENT Professor(first_name, last_name, e-mail, Publication*)> <!ELEMENT Student(name, e-mail)> <!ELEMENT Course(denomination, Professor)> <!ELEMENT Publication(title, year, journal, editor)> <!ELEMENT School_Member(name, e-mail)> <!ELEMENT name (#pcdata)>...

Table 1. A fragment of the University (UNI) and Computer Science (CS) DTDs

By means of the XML/DTD wrapper, the obtained DTDs are translated into ODL_{I^3} descriptions. An example of the classes obtained in this step is shown in Table 2.

2.4 Annotation of a local source with WordNet

With reference to the Semantic Web area, where generally the annotation process consists of providing a web page with semantic markups w.r.t. an ontology, in our approach we markup the metadata descriptions extracted by the wrappers, i.e. the ODL_{I^3} schemata, and the reference lexical ontology is WordNet.

The WordNet database contains 146,350 lemma organized in 111,223 synonym sets. WordNet's starting point for lexical semantics comes from a conventional association between the forms of the words – that is, the way in which words are pronounced or written – and the concept or meaning they express. These associations give rise to several properties, including synonymy, polysemy, and so forth. The correspondence between the words form and their meaning is

University Site (UNI)	Computer Science Site (CS)
...	...
Interface `Research_Staff`	Interface `Professor`
`(Source Un_site.dtd)`	`(Source Sc_site.dtd)`
`{ attribute string name;`	`{ attribute string first_name;`
`attribute string email;`	`attribute string last_name;`
`attribute set<Section> section;`	`attribute string email;`
`attribute set<Article> article;}`	`attribute set<Publication>`
	`publication;}`
Interface `Article`	Interface `Publication`
`(Source Un_site.dtd)`	`(Source Sc_site.dtd)`
`{ attribute string title;`	`{ attribute string title;`
`attribute string journal;`	`attribute string year;`
`attribute string conference;`	`attribute string journal;}`
`attribute string year; }`	
...	...

Table 2. A piece of the University (UNI) and Computer Science (CS) sources in ODL_{I^3}

represented in the so-called Lexical Matrix M (see table 3), in which the words meaning are reported in rows (hence each row represents a synset) and columns represent the words form (form/base lemma).

	WF1	WF2	WF3	...	WFn
M1	E1,1	E1,2			
M2		E2,2			
M3			E3,3		
...				...	
Mm					Em,n

Table 3. WordNet word form and meanings

Thus, entry E1,1 implies that word form F1 can be used to express word meaning M1. If there are at least two entries in the same column then the corresponding word form is polysemous (i.e. it can be used to represent more than one meaning, exactly two in this case); if there are at least two entries in the same row then two word forms are synonyms relative to a context.

Given a word form `F`, its i-th meaning will be denoted by `F♯i`. For example, the word form `course` has 8 meanings in WordNet; the first one is `course♯1` = `"education imparted in a series of lessons or class meetings"`.

In the phase of a local source annotation, the integration designer has to manually choose the appropriate WordNet meaning for each element of the conceptual schema provided by the wrappers. The annotation phase is composed of two different steps:

1. **Word Form choice.** In this step, the WordNet morphologic processor aids the designer by suggesting a word form corresponding to the given term. More precisely, the morphologic processor stems (i.e. converts to a common root form) the term and checks if it exists as word form.
2. **Meaning choice.** The designer can choose to map an element on zero, one or more senses. Notice that the user can only choose a sense among the existing ones in WordNet, and that is he is not allowed to extend it with his new meanings.

Notice that, for a compound descriptive term, our tool extracts the component terms and all these terms are processed by the WordNet morphologic processor. For example if the attribute name is `shipment_received_date` then the terms `shipment`, `received`, and `date` are proposed to the designer. If a term is not available as word form (this can happen, for example, for an abbreviation), if there is an ambiguity, or the selected word form is not satisfactory, the designer can choose another word form of WordNet or manually search for a meaning of the term. A term that doesn't find a meaning within WordNet is considered as unknown term and no lexicon relationship will be derived for it (see next section).

This phase assigns a name, LEN (this name can be the original one or a word form chosen from the designer), and a set (that might be empty) of meanings, LEM_i (a class or attribute meaning is given by the disjunction of its set of meanings), to each local element (class or attribute) LE of the local schema:

$$LE = \langle LEN, \{LEM_1, \ldots, LEM_k\}\rangle,\ k \geq 0$$

For example:

`CS.Course` = < `course`, {`course`♯1} >
`UNI.Professor` = < `professor`, {`professor`♯1} >
`UNI.School_Member` = < `student`, {`student`♯1} >
`UNI.School_Member.name` = < `name`, {`name`♯1} >

where

`course`♯1 = 'education imparted in a series of lessons or class meetings'
`professor`♯1 = 'someone who is a member of the faculty at a college or university'
`student`♯1 = 'a learner who is enrolled in an educational institution'
`name`♯1 = 'a language unit by which a person or thing is known'

2.5 Common Thesaurus Generation

The Ontology Builder constructs a Common Thesaurus describing intra and inter-schema knowledge in the form of relationships SYN, BT, NT, and RT.

The Common Thesaurus is constructed through an incremental process in which relationships are added in the following order:

1. *schema-derived relationships*: relationships holding at intra-schema level extracted by analyzing each schema separately;

2. *lexicon-derived relationships*: These originate from the annotation of the schemas respect the lexical ontology. WordNet defines a large variety of semantic relations between its meanings. A lexicon relationship between terms for the common thesaurus is derived from a semantic relation in WordNet between the annotated meanings of the terms according to the following correspondences:

   Synonymy: corresponds to a SYN relation
   Hypernymy: corresponds to a BT relation
   Hyponymy: corresponds to a NT relation
   Holonomy: corresponds to a RT relation
   Meronymy: corresponds to a RT relation
   Correlation: corresponds to a RT relation

3. *designer-supplied relationships*: new relationships can be supplied directly by the designer, to capture specific domain knowledge. This is a crucial operation, because the new relationships are forced to belong to the Common Thesaurus. This means that, if a nonsense or wrong relationship is inserted, the subsequent integration process can produce a wrong global schema;
4. *inferred relationships*: Description Logics techniques of ODB-Tools [5] are exploited to infer new relationships, by means of subsumption computation applied to a "virtual schema" obtained by interpreting BT/NT as subclass relationships and RT as domain attibutes.

In our running example, some of the relationships automatically obtained and proposed at the integration designer are the following:

schema derived: `CS.Professor NT CS.Person`
schema derived: `CS.Student NT CS.Person`
lexicon derived: `UNI.School_Member NT CS.Person`
lexicon derived: `UNI.Article NT CS.Publication`
designer-supplied: `UNI.Research_Staff SYN CS.Professor`
inferred: `UNI.Research_Staff NT CS.Person`
inferred: `UNI.Research_Staff RT UNI.Article`

If the designer accepts and confirms the above relationships, they are included in the Common Thesaurus.

2.6 Global Virtual View generation

The proposed methodology allows us to identify similar ODL_{I^3} classes, that is, classes that describe the same or semantically related concept in different sources. To this end, *affinity coefficients* (i.e., numerical values in the range $[0, 1]$) are evaluated for all possible pairs of ODL_{I^3} classes, based on the relationships in the Common Thesaurus properly strengthened. Affinity coefficients determine the degree of matching of two classes based on their names (*Name*

Affinity coefficient) and their attributes (*Structural Affinity* coefficient) and are fused into the *Global Affinity* coefficient, calculated by means of the linear combination of the two coefficients. For a detailed description of the affinity coefficient evaluation, the reader can refer to [7]. Global affinity coefficients are then used by a hierarchical clustering algorithm [9], to classify ODL_{I^3} classes according to their degree of affinity. The output of the clustering procedure is an affinity tree, where ODL_{I^3} classes are the leaves and intermediate nodes have an associated affinity value, holding for the classes in the corresponding cluster. Clusters for integration (candidate clusters) are interactively selected from the affinity tree using a threshold based mechanism whose parameter are set by the designer. Regarding the quality of our clustering results the reader can refer to [6] where a deep discussion of the experimentations results of the use of the strengthened terminological relationships and affinity-based clustering is reported.

The generation of **Global Classes** out of selected clusters is a synthesis activity performed interactively with the designer: a Global Class GC_i definition is built for each cluster Cl_i. The GVV generation consists of two phases. First, the system automatically associates a set of global attributes with GC_i, corresponding to the union of local attributes of the classes belonging to Cl_i. Then, the system proposes to the designer the restriction of the global attributes set by exploiting the Common Thesaurus lattice that contains SYN relationships and BT/NT relationships among local attributes.
For each global class, a persistent **Mapping Table** MT storing all the mappings is generated; it is a table whose columns represent the set of local classes which belong to the cluster and whose rows represent the global attributes. An element $MT[GA][LC]$ represents the set of attributes of the local class LC which are mapped into the global attribute GA: the value of the GA attribute is a function of the values assumed by the set of attributes $MT[GA][LC]$. Some simple and frequent cases of such function are the following:

- *identity*: the GA value is equal to the LA value; we denote this case as $MT[GA][LC] = LA$
- conjunction: the GA value is obtained as a conjunction of the values assumed by a set of local attributes LA_i of the local class LC; we denote this case as $MT[GA][LC] = LA_1$ `and` $\ldots$ `and` LA_n
- *constant*: GA assumes into the local class LC a constant value set by the designer; we denote this case by $MT[GA][L] =$ `const`
- *undefined*: GA is a set undefined into the local class LC; we denote this case as $MT[GA][L] =$ `null`.

In our running example the integration process gives rise to three global classes:

```
Global1: (UNI.Section, CS.Course)
Global2: (UNI.Article, CS.Publication)
Global3: (UNI.Research_Staff, UNI.School_Member, CS.Professor, CS.Student)
```

For each global class a Mapping Table is generated. For example the Mapping Table for Global2 is:

	UNI.Article	**CS.Publication**
Title	Title	Title
Year	Year	Year
Journal	Journal	Journal
Conference	Conference	null
Editor	null	Editor

Table 4. Mapping Table of the global class Global2 (Publication)

3 Global Virtual View Annotation

In this section, we propose a semi-automatic methodology to annotate a GVV, i.e. to assign a name, GEN, and a set (that might be empty) of meanings, GEM_i (a class or attribute meaning is given by the disjunction of its set of meanings) to each global element (class or attribute) GE:

$$GE = \langle GEN, \{GEM_1, \ldots, GEM_p\}\rangle,\ p \geq 0$$

3.1 Global Class Annotation

In order to semi-automatically associate an annotation to each global class, we consider the set of all its "broadest" local classes, w.r.t. the relationships included in the Common Thesaurus, denoted by GC_B:

$$GC_B = \{LC \in GC | \neg\exists y \in GC, (LC\ \texttt{NT}\ y) \vee (y\ \texttt{BT}\ LC)\}$$

In our example:

	GC	GC_B
GC_1	`CS.Course, UNI.Section`	`CS.Course, UNI.Section`
GC_2	`CS.Publication, UNI.Article`	`CS.Publication`
GC_3	`CS.Professor, CS.Person,UNI.School_Member, UNI.Research_Staff, CS.Student`	`CS.Person`

On the basis of GC_B, the designer will annotate the global class GC as follows:

- **name choice**: the integration designer is responsible for the choice of the GC name: the system only suggests a list of possible names. The designer may select a name within the proposed list or select another name not inside the list. In particular, concerning the *name* and according to the role of the global class name (to allow the designer to identify the Global Class and its contents), we consider the name as a label. Therefore, a name might not be a word form of WordNet. For example, regarding Global Class GC1 (see Table 5), the designer selected the name *course* between the suggested *Course* and *Section*. Regarding GC3 the designer chose a more significative name (*University_Member*) instead of the proposed generic *person*.

– **meaning choice**: the union of the meanings of the local class names in GCB are proposed to the designer as meanings of the Global Class. The designer may change this set, by removing some meanings or by adding other ones.

With respect to our example, the proposed annotations are the following:

GC	Names	Meanings
GC1	`course or section`	`course♯1`
GC2	`publication`	`publication♯1`
GC3	`University_Member`	`person♯1`

Table 5. University GVV annotation

3.2 Global Attributes Annotation

We extend the previously used approach for names and meanings of the attributes. Given a global attribute GA of the global class GC, we consider the set LGA of local attributes, which are mapped into GA:

$$LGA = \{LA | \exists LC \in GC, LA \in LC \wedge MT[GA][LA] \neq \texttt{null}\}$$

and the set of all its "broadest" local attributes, denoted by LGA_B:

$$LGA_B = \{LA \in LGA | \neg \exists y \in LGA, (LA \ \texttt{NT} \ y) \vee (y \ \texttt{BT} \ LA)\}$$

On the basis of LGA_B, the designer will annotate the global attribute as described for global classes. Moreover, according to mapping function, we may develop some specific policy to automatically select meanings.

4 Adding a new source

Supporting the evolution of an ontology represents a challenging issue (to be faced). Many interesting solutions have been developed with regard to this topic [13, 14] and an oustanding idea is to exploit multiple variants of the same ontology to cope with changes. This approach, called *Ontology Versioning*, is different from our proposal where a single ontology is kept consistent with the sources which refer to.

Within Ontology Builder if new sources are added/deleted, or if some changes occur in the sources, the corresponding GVV has to change. The integration process is expensive both for the designer and for the system. For this reason, we propose a methodology for integrating a new source, which exploits the previous integration work, i.e., a built-up GVV, without restarting the integration process from scratch.

In the GVV building approach all the sources to be integrated contribute with the same weight to the process. Therefore, if we consider an already built GVV and we have to insert a new source which refers to the same ontology, we can assume that this source brings less semantics than the GVV itself. For this reason, we devise an integration process of a new source that starts from the obtained GVV and tries to integrate a new source in the GVV.

In the following, we show how the evolution of a GVV caused by the insertion of a new source can be strongly simplified by having available the lexicon-based knowledge of the GVV annotation.

4.1 Integration of a new source in a GVV

The insertion of a new source is managed as an integration process between two schemata: the GVV and the new source schema; in other words, the global classes of GVV are considered as local classes and are integrated with the local classes of the new source.
We show the approach analyzing all the integration phases of the GVV with the new source. We introduce the following notation:

`gcNew` the global class of the new integrated schema has a name, `gcNewName` and a set of global attributes $\texttt{gcNewAtt}_i$,

`gcOld` the global class of the old integrated schema has a name, `gcOldName` and a set of global attributes $\texttt{gcOldAtt}_j$,

`lcNew` the local class of the new source has a name, `lcNewName` and a set of local attributes $\texttt{lcNewAtt}_k$.

According to the integration methodology, we have to create a Common Thesaurus of the involved sources. In this case, the Common Thesaurus will contain schema-derived relationships extracted from the analysis of the new source and intra-schema lexicon-derived relationships obtained by the annotation of the new source. Further, the GVV global classes have to be semantically enriched according to the semi-automatic annotation method shown in section 3. The interesting point is that the annotation of GVV allows us to discover inter-schema lexical relationships which enrich the Common Thesaurus.

The next step is the cluster generation followed by Global Classes and mapping tables generation. This phase has to provide mapping rules among Global Classes and new or old local classes. In order to achieve this result, we substitute here old Global Classes with the respective Local Classes. In this way, new Global Classes that represent old Local Classes and new Local Classes as are built. Thus we have:

$$\texttt{gcNew} = \{\texttt{gcOld}_1, \ldots, \texttt{gcOld}_p, \texttt{lcNew}_1, \ldots, \texttt{lcNew}_n\}$$

the resulting rewriter step is:

$$\texttt{gcNew} = \{\texttt{lcOld}_{11}, \ldots, \texttt{lcOld}_{1z}, \ldots, \texttt{lcOld}_{p1}, \ldots, \texttt{lcOld}_{pn}, \texttt{lcNew}_1, \ldots, \texttt{lcNew}_n\}$$

With Global Class generation, we observe that, using the same clustering parameters, an old Global Class $\texttt{lc}_1,\ldots,\texttt{lc}_i,\ldots,\texttt{lc}_n$ changes only if the integration process inserts one or more new local classes ($\texttt{lcNew}_i$) into the Global Class. Therefore, we observe that the following cases are possible:

a) A new global class `gcNew` is composed of only one old global class (`gcOld`) and one or more new local classes ($\texttt{lcNew}_i$):

$$\texttt{gcNew} = \{\texttt{gcOld},\texttt{lcNew}_1,\ldots,\texttt{lcNew}_i,\ldots,\texttt{lcNew}_n\}$$

The new global class (`gcNew`) may have new global attributes generated from the semantic contribution of new local classes. New mapping rules are defined among a global attribute and its corresponding local attribute(s). In this case, global attributes belonging to the `gcOld` ($\texttt{gcOldAtt}_i$) may map both local classes of the old Global Class and new local classes (see the columns associated to $\texttt{lcNew}_t$, for example). New global attributes can only map new local Classes (null mappings in the following table).

So we can say that meanings associated to each global attribute are:

- The meaning of old global attributes have to be enriched with the meanings of the new local classes mapped by these attributes;
- The meaning of new global attributes have to be set according to the rules defined before (see 3.2).

<table>
<tr><td></td><td>$lcOld_1$</td><td>...</td><td>$lcOld_k$</td><td>$lcNew_1$</td><td>$lcNew_t$</td><td>$lcNew_n$</td></tr>
<tr><td>$gcOldAtt_1$</td><td colspan="3" rowspan="3">the same mappings as in $gcOld$</td><td colspan="3" rowspan="5">new mappings</td></tr>
<tr><td>...</td></tr>
<tr><td>$gcOldAtt_m$</td></tr>
<tr><td>$gcNewAtt_1$</td><td colspan="3" rowspan="3">null mappings</td></tr>
<tr><td>...</td></tr>
<tr><td>$gcNewAtt_p$</td></tr>
</table>

Table 6. New mapping table example.

b) A global class of the new integrated schema is composed of only new local classes:

$$\texttt{gcNew} = \{\texttt{lcNew}_1,\ldots,\texttt{lcNew}_i,\ldots,\texttt{lcNew}_n\}$$

This situation describes the case in which the GVV is extended without interfering with the previous one.

The new global class (`gcNew`) has a name (`gcNewName`) and a set of new global attributes ($\texttt{gcNewAtt}_i$), where each new global attribute maps only new local attributes. The names and meanings of the global attributes are defined following the rules stated before (see 3.2).

c) A global class of the new integrated schema is composed of more than one global class of the GVV and at least one local class of the new source we are integrating.

$$\texttt{gcNew} = \{\texttt{gcOld}_1, \ldots, \texttt{gcOld}_p, \texttt{lcNew}_1, \ldots, \texttt{lcNew}_i, \ldots, \texttt{lcNew}_n\}$$

In this case the previous GVV is modified; side effects can influence the applications based on the previous schema. The new global class (`gcNew`) has a name (`gcNewName`) and a set of new global attributes ($\texttt{gcNewAtt}_i$).

5 Concluding remarks

In this paper, we presented a methodology for supporting the semi-automatic building, annotation and extension of a domain ontology obtained by integrating web documents with the Ontology Builder component of the SEWASIE System. Talking about the evolution issue and, in particular, the addition of a new source, we had to face two different problems: the system overload to maintain the built ontology corresponding to the involved sources, and, the insertion of a new source that may modify the existing ontology, with a side effect to each application based on the ontology.

We tried to solve both problems and the most relevant advantages of our methodology of integrating a new source into a GVV is that the process is less expensive than starting from scratch and it is done starting from semantically annotated results of previous integration processes. Possible limitations are:

- mistakes of the previous integration process might propagate to the new GVV;
- the new GVV is based on the previous one, and so it might not perfectly represent all the sources.

Acknowledgements

This work is supported in part by the $5th$ Framework IST program of the European Community through project SEWASIE within the Semantic Web Action Line. The SEWASIE consortium comprises in addition to the authors' organization (Sonia Bergamaschi is the coordinator of the project), the Universities of Aachen RWTH (M. Jarke), Roma La Sapienza (M. Lenzerini, T. Catarci), Bolzano (E. Franconi), as well as IBM Italia, Thinking Networks AG and CNA (Association of SMEs) as user organizations.

References

1. D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, and M. Vincini. Information integration: The momis project demonstration. In *VLDB 2000, Proc. of 26th International Conference on Very Large Data Bases, 2000, Egypt*, 2000.

2. D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. The MOMIS approach to information integration. In *AAAI International Conference on Enterprise Information Systems (ICEIS 2001)*, 2001.
3. D. Beneventano, S. Bergamaschi, S. Lodi, and C. Sartori. Consistency checking in complex object database schemata with integrity constraints. *IEEE Transactions on Knowledge and Data Engineering*, 10:576–598, July/August 1998.
4. D. Beneventano, S. Bergamaschi, and C. Sartori. Description logics for semantic query optimization in object-oriented database systems. *ACM Transaction on Database Systems*, 28:1–50, 2003.
5. D. Beneventano, S. Bergamaschi, C. Sartori, and M. Vincini. ODB-Tools: A description logics based tool for schema validation and semantic query optimization in object oriented databases. In *Proc. of Int. Conf. on Data Engineering, ICDE'97*, Birmingham, UK, April 1997.
6. S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini. Semantic integration of heterogenous information sources. *Data and Knowledge Engineering*, 36(3):215–249, 2001.
7. S. Castano, V. De Antonellis, and S. De Capitani di Vimercati. Global viewing of heterogeneous data sources. *IEEE Transactions on Data and Knowledge Engineering*, 13(2), 2001.
8. Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: automatic data extraction from data-intensive web sites. In *SIGMOD Conference*, 2002.
9. B. Everitt. Cluster analysis. *Heinemann Educational Books Ltd*, 1974.
10. R. Baumgartner S. Flesca and G. Gottlob. Visual web information extraction with lixto. In *the 27th International Conference on Very Large Data Bases (VLDB 2001)*. Roma, Italy, September, 2001.
11. T. R. Gruber. *A translation approach to portable ontology specifications.*, volume 5. 1993.
12. N. Guarino. Formal ontologies and information systems. In *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS'98)*, Trento, Italy, june 1998.
13. J. Heflin and J. Hendler. Dynamic Ontologies on the Web. In *In Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 443–449. AAAI/MIT Press, 2000.
14. M. Klein and D. Fensel. Ontology Versioning on the Semantic Web. In *First Intl' Semantic Web Working Symposium. 2001*, 2001.
15. M.Lenzerini. Data integration: A theoretical perspective. In Lucian Popa, editor, *Proc. of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 233–246, Madison, Wisconsin, USA, 2002. ACM.
16. Jussi Myllymaki. Effective web data extraction with standard xml technologies. In *WWW*, pages 689–696, 2001.