A Controlled Language for Semantic Annotation and Interoperability in e-Business Applications[1]

M.Missikoff, F.Schiappelli, F.Taglino

LEKS, Lab for Enterprise Knowledge and Systems,
IASI-CNR (Italy),
{missikoff, fed_schi, taglino}
@iasi.cnr.it
http://leks.iasi.rm.cnr.it/home

Abstract. In this paper we present the basic ideas underlying a solution for software application interoperability in a business context. Key elements of our solution are a *Reference Ontology*, aimed at modelling the key aspects of a business domain, and a *Semantic Annotation Language*, SMAIL, used to associate semantic expressions, defined in terms of the reference ontology, to business elements.

Keywords. Semantic annotation, formal languages, ontology, business, interoperability.

1 Introduction

The considerable impact of the Internet on computer interconnection raised a high expectation in the area of application software interoperability. However, the experience shows that, despite the advances of current technology, two different legacy systems hardly can cooperate to carry out a common business task, even if data and procedures deal with the same business entities.

Our work is based on the idea that, to achieve interoperability among different systems, it is necessary to expose the actual semantics of data and programs, often deeply concealed by superficial differences, such as naming, syntactic and structural discrepancies.

To provide an effective solution to this problem it is necessary to shift towards a semantic level of interaction, i.e., semantic interoperability; to this end we need to make explicit the semantics hidden in, e.g., an application interface. A promising approach to achieve semantic interoperability requires the use of a Reference Ontology (RO) and a Semantic Annotation Language, based on the former.

Semantic Annotation (SA) has been proposed in literature mostly to annotate documents and web pages [SeWeb]. Only few proposals are aimed at the creation of additional structures that represent (in a formal, controlled way) the semantic content of a web resource (e.g., a document, a business process or an eService).

Among typical applications of semantic annotation, we can find:

- *Document Management*, for semantic search;
- *Knowledge Management*, for organization and retrieval of enterprise knowledge;
- *Web Services* publishing and discovery, with semantic matchmaking of requested and offered services;
- *Semantic Interoperability*, by annotating local resources (information and processes) to support business cooperation among enterprise software applications.

In the literature, the first two applications have attracted most of the attention. They are addressed by solutions referred to as "human-oriented" annotations. This kind of annotation solutions are provided by systems such as Annotea [KPS01], Annotation System for Semantic Web [VR02], Trellis [GV02]. Such systems are interactive environments that allow users to add, in a ***descriptive*** way (plain text), an annotation representing the content of the documents.

A second important class is represented by the solutions referred to as "machine-oriented" annotations. This kind of SA is provided by systems such as MnM [VM*02], SMORE [KP*02], SHOE Knowledge Annotator [LS*97], COHSE [BG01]. These solutions aim at representing, in a ***formal*** way, the conceptual content of a given web resource. The user annotates segments of text, typically in a web page, using tags based on the concepts defined in an *Ontology*. This activity is known as "ontology driven mark-up".

Our work evolves along the second line, since we propose a solution for ontology-driven semantic annotation, aimed at the interoperability of software applications in an e-business context. The main difference, with respect to the previously mentioned

[1] This work was partially supported by IST European Project Harmonise

tools, is that they mainly aim at enriching a web page, embedding the SA in the document itself. Our method allows formal, ontology-based, structures to be created externally (but tightly connected) to the web resource, on the line of the OntoMat Annotizer [HS02, HS03] approach. We refer to this formal structure as the "semantic image" of the resource. In this way, the annotation, being not embedded, can be associated to any kind of resource, such as a video, a sound, or a web service. Furthermore, starting from a collection of web resources, it is possible to gather their semantic images to build a semantic index for a Semantic Web architecture. Semantic search and matchmaking can be implemented for fast retrieval of web resources, based on the actual knowledge they carry.

Another important characteristic of our approach is that the proposed annotation language is tightly controlled, based on a reference ontology that, in its terminological content, is part of the language.

2 Semantic Annotation for Enterprise Interoperability

The goal of interoperability is to allow different software applications to exchange data and services, despite the fact that the two software systems were not originally conceived for cooperation. It is well known that, even if data and procedures deal with the same business entities, existing software applications exhibit deep differences in their internal organization, database schemas, software architectures, and other important technical characteristics, that hinder a smooth cooperation.

To solve such a problem, it is necessary to identify the business entities addressed, i.e., the semantics of the information elements and operations that are involved in the cooperation, beyond the syntactic and structural differences.

The problem that we address bears some similarity with the area of heterogeneous information sources integration, addressed in the database field. In this area, two basic approaches have been proposed: global-as-view (GAV) [MP*97, TRV98, GB*99] and local-as-view (LAV) [Hal01, Lenz02]. The LAV approach (for information interoperability) implies that each application system interacts with any other system as if its own data organisation was the only existing solution, i.e., as if all the other software applications were organised in the same way. The LAV approach [KLS95, AD98, CD*01] does not require a global schema to be built, but the existence of a common view of the business scenario where the cooperation takes place. This common view, in our approach, is represented by a shared Reference Ontology (RO). The RO, built by a team of domain experts, provides precise and formal (therefore, computer processable) definitions of relevant (for the business context) domain entities. The terminological component is used to annotate the resources managed by the cooperating systems.

In this paper we restrict the focus to information interoperability; the set of information elements of a given software application, participating in the interoperability process, will be referred to as PLCS (Public Local Conceptual Schema).

2.1 The Annotation Process

The semantic annotation process is a critical one; it represents a first phase in which a given legacy system is confronted with its inherent inclination to interoperate within a given business community. In fact, the reference ontology RO is assumed to be a proper representation of the business domain, and in particular of the part that will be involved in the networked business activities. It is fair to assume that a given information or service that is not defined in the RO is not of interest of the community. Therefore, in the semantic annotation process, a PLCS element that cannot be annotated is assumed to be (at that moment) of scarce interest for the networked business. But, since the reality continuously evolves, we assume that suitable mechanisms will be implemented to update the RO whenever a sufficient consensus is reached in order to modify it.

Besides the cases where some PLCS elements fall outside the (ontological) scope of the business domain, there are other cases where an annotation that precisely captures the intended meaning of a PLCS element is not possible. We refer to these cases as "annotation mismatches". In fact we can have:

Lossless annotation: when the annotation fully captures the intended meaning,

Lossy annotation: when the annotation fails to fully representing the intended meaning.

In the first case, a PLCS element exactly corresponds to a concept in the RO or its meaning can be precisely expressed by a suitable composition of concepts. In the second case, the meaning of a PLCS element does not have a matching concept in the ontology, nor the possibility of compositionally express it, since either:

- the intended meaning is outside the scope of the RO;

- the PLCS element is not sufficiently refined (i.e., it does not match the accuracy level of the ontology) (*underspecification*)
- the PLCS element present a level of refinement not deemed useful, that does not match the level of refinement of the RO (*overspecification*).

Annotation mismatches may derive from different organizations of information in the PLCS and RO, but also from different views of the world. However, having in mind a specific concept (or a set of concepts), represented in two different models, there are a limited number of possible divergences. We have a first list of differences that the knowledge engineer must consider in annotating a PLCS. We present them divided in the two broad categories previously introduced: lossless and lossy (see an excerpt in Table 1).

Lossless mismatches	
Path-Naming	different labels for the same content (the attribute names *Name* and *Denomination* to indicate a hotel name)
Encoding	different formats of data or units of measure (a *Price* expressed in dollars and in euro)
Structuring	different structures for the same content (an *Address* represented as a string or a composition of the *Street_name*, and *Street_number* fields)
Lossy mismatches	
Content	different content denoted by the same concept - typically expressed by enumeration (the *hotel services* concept described by different enumeration items)
Coverage	presence/absence of information (the *mobile phone* in the PLCS, but not in the RO)
Precision	the accuracy of information (the *distance* expressed by an integer value or by strings such as *near, far*)
Abstraction	level of specialisation refinement of the information (the distinction bw *indoor* and *outdoor swimming pool* versus a generic *swimming pool* concept)
Granularity	level of decomposition refinement of the information (the *restaurant* represented as a whole or as an aggregation of a *terrace* and an *indoor_rooms*)

Table 1. Sorts of mismatches

Semantic annotation is a critical process that requires deep knowledge on the domain, the RO and the legacy system. Given a software application, and in particular its Public Local Conceptual Schema, its semantic annotation is accomplished by performing the following steps:

- *Identification of the PLCS elements*. The first step consists in the identification of the elements, essentially information (provided or requested), necessary for the system to participate in the cooperation with other systems.
- *Identification of the intended meaning*. Then, each PLCS element is clearly assigned with an "intuitive" semantics, by associating the business elements represented (informal annotation).
- *Identification of the related concepts in the RO*. The most appropriate concepts that express the intended meanings are then chosen.
- *Definition of the semantic expressions*. By using the selected RO concepts, an expression that specifies the intended meaning is constructed. For each PLCS element, the best fitting annotation expression is built (formal semantic annotation expressions).
- *Semantic coverage assessment*. The intended meaning of the PLCS element is contrasted with the semantic annotation expression, to see if there is any loss of semantics.
- *Association of the semantic annotation expressions to the PLCS elements*. Finally, each semantic annotation expression can be associated to the correspondent PLCS element, using the correct connective (for lossless or lossy cases). In the following section, this process is further elaborated.

3 SMAIL: a Controlled Semantic Annotation Language

To create the semantic annotation expressions, we propose to use SMAIL (Semantic Mediation and Application Interoperability Language).

It is important to note that SMAIL is characterized by a closed vocabulary. This means that, unlike the other annotation languages, the user cannot define his/her own terms nor named concepts. The sentences of SMAIL can be constructed only using the terms (i.e., concepts) defined in the Reference Ontology. A naming policy, inventing labels for variables, subroutines, relation names, etc., is one of the most critical aspects of the development of an information system. For this reason, one of the main characteristics of SMAIL is the fact that, in building a semantic expression, the user needs to look at the ontology and can only select terms denoting defined concepts. Therefore the terminological elements of

the ontology become part of the language; the generation of the annotation expressions is performed by a composition and/or transformation of ontology elements.

More in detail, an annotation expression is composed of a left-hand-side and a right-hand-side. On the left-hand-side only a name (or path) of an information element in the PLCS can appear; it identifies the PLCS element to be annotated. On the right-hand-side only ontology elements, and SMAIL constructors, can appear. As anticipated we have lossless and lossy annotations and we introduce specific connectives to express these kinds of annotation.

Lossless annotations can be expressed with a Semantic Equivalence connective

PLCS_elem =: SA_expression

Lossy annotations can be expressed with Over/Underspecification connectives

PLCS_elem >: SA_expression (Local Overspecification)

PLCS_elem <: SA_expression (Local Underspecification)

3.1 The SMAIL Grammar

In the following we give a formal specification of the SMAIL language by defining the grammar that generates it.

$$G_{SMAIL} = (N,T,P,\Sigma)$$

where

- N is the set of non-terminal symbols,
- T is the set of terminal symbols, where labels are terms in the ontology
- P is the set of production rules,
- Σ is the start symbol.

In Fig.2,3 the element of the 4-tuple are presented in detail. Terminal symbols are in *italic*, while non-terminal symbols are in UPPER CASE.

- N = {SE,OE,OE_SEQ,COND,EXP}
- T = $V_{ont} \cup O \cup A \cup D \cup \{\perp,\varepsilon\}$
  - V_{ont} = set of the ONTOLOGY TERMS
  - O = {*,* , *;* , *and* , *or* , *(* , *)* }
  - A = {*>*, *>=*, *<*,*<=*, *=*, *+*,*-*,***,*/*,*%*}
  - D={*0..9*}

Figure 2: Nonterminal and Terminal sets for G_{SMAIL}

SE ::= ⊥
  | EXP
EXP ::= OE COND
  | OE_SEQ COND
  | ARITM_EXP COND
  | BOOL_EXP
OE ::= *oelem* ... *oelem* ∈ V_{ont}
OE_ SEQ ::= OE
  | OE, OE_SEQ
ARITM_EXP ::= grammar for the arithmetic expressions, where the operands are either ontology elements (oelem) or real numbers
BOOL_EXP ::= *(*EXP*)*
  | *(*EXP*) and* BOOL_EXP
  | *(*EXP*) or* BOOL_EXP
COND ::= ε | *;* ...grammar for the boolean expressions, where the operands are either ontology elements (oelem) or real numbers

Figure 3: Production rules for G_{SMAIL}

Please, note that SMAIL is not intended for direct use by a knowledge engineer. It is at the basis of the annotation tool associated to SymOntoX [MT02], the Ontology Management System developed at LEKS, IASI-CNR. Therefore, in actual applications, the complexity of the annotation language is shielded from the user by a friendly graphical user interface. Furthermore, we are currently working on a version of OWL [GH03] referred to as SMOWL, to cast the annotation approach of SMAIL into an XML-based ontology language.

4 A few examples

Some examples of semantic mismatches introduced in Table 1 and the SMAIL expressions aimed at solving them are shown in Tables 2-3. In the *Address* case, a simple string is annotated with a concatenation of two strings. Please note that syntactical details, such as separators in the PLCS *Address*, are not dealt with here since we focus on the semantic aspect of annotation. Such implementation details will be addressed in later phases, when semantic annotation will be used to build semantic adaptors for interoperability [MT03]. Please note that the nihil symbol (assumed to be defined in the RO) is used to denote the undefinedness. Furthermore, the *Swimming Pool* example, a concept *Swimming_Pool* is supposed to exist in the RO, defined as a generalization of the *Indoor_Sw_Pool* and *Outdoor_Sw_Pool* concepts.

PLCS **Hotel**	**RO** **Hotel**	**Mismatch**
Name: *literal*	Denomination: *literal*	Naming
Address: *literal*	Address [Street_name: *literal* Street_number: *literal*]	Structuring
Services: enum ('security box', 'hamam', 'parking')	Services: enum ('safe', 'sauna', 'ironing center')	Content
Telephone: *literal* Mobile-phone: *literal* Fax: *literal*	Contact_Info [Phone: *literal* Fax: *literal* Email: *literal*]	Coverage, Structuring, Naming
Location: enum ('near city', 'far city', 'near airport', 'far airport')	Location [Distance: *literal* from: enum ('city', 'airport')]	Precision
SwimmPool: enum ('yes', 'no')	Facilities: [Indoor_Sw_Pool: enum ('yes', 'no') Outdoor_Sw_Pool: enum ('yes', 'no')]	Abstraction, Structuring, Naming
Restaurant: enum ('yes', 'no')	Restaurant: [Terrace: enum ('yes', 'no') Indoor_Room: enum ('yes', 'no')]	Granularity

Table 2: Mismatches examples

Semantic Annotation	**Mismatch**
PLCS.Hotel.Name=: RO.Hotel.Denomination	Naming
PLCS.Hotel.Address.Location =: RO.Hotel.Address.Street_name, RO.Hotel.Address.Street_number	Structuring
PLCS.Hotel.Services("security box") =: RO.Hotel.Services("safe") PLCS.Hotel.Services("hamam") =: RO.Hotel.Services("sauna") PLCS.Hotel.Services("parking") <: ⊥	Content
PLCS.Hotel.Mobile-phone <: ⊥	Coverage
PLCS.Hotel.Location("near city") =: (RO.Hotel.Location.From("City")) and (RO.Hotel.Location.Distance(); RO.Hotel.Location.Distance<=2) PLCS.Hotel.Location("near airport") =: (RO.Hotel.Location.From("Airport")) and (RO.Hotel.Location.Distance(); RO.Hotel.Location.Distance<=2) …	Precision
PLCS.Hotel.SwimmPool =: RO.Hotel.Facilities.Swimming_Pool	Abstraction
The ontology is richer than the PLCS	Granularity

Table 3: Semantic annotation examples

5 Conclusions

In this paper we briefly presented the main issues of SMAIL, an ontology-based semantic annotation language conceived for semantic interoperability among software applications. The proposed language is used to assign meaning to elements of legacy systems that are exchanging information with each other. Semantic annotation is the first, preliminary phase, to allow semantic interoperability.

The proposed language is based on a Reference Ontology that determines the expressions that can be built. In this way any possible expression has a precise, unambiguous semantics. SMAIL is, therefore, a controlled language with a closed, ontology-based, vocabulary.

Along this line, a first solution for semantic interoperability has been developed within the European Project Harmonise [Harmo],[Miss*03], that originated the presented work.

References

[AD98] Abiteboul, S., Duschka, O.: Complexity of answering queries using materialized views. In: Proc. Of PODS'98. (1998) 254–265

[BG01] Sean Bechhofer, Carole Goble. Towards Annotation using DAML+OIL. K-CAP 2001 workshop on Knowledge Markup and Semantic Annotation, Victoria B.C, October 2001

[CB84] D.J.Cook, H.E.BEZ; Computer Mathematics; Cambridge University Press, 1984

[CD*01] Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Data integration in data warehousing. Int. J. of Cooperative Information Systems 10 (2001) 237–271

[GB*99] Goh, C.H., Bressan, S., Madnick, S.E., Siegel, M.D.: Context interchange: New features and formalisms for the intelligent integration of information. ACM Trans. on Information Systems 17 (1999) 270–293

[GH03] Deborah L. McGuinness and Frank van Harmelen eds. Web Ontology Language (OWL): Overview, W3C Last Call Working Draft 31 March 2003.

[GV02] Gil, Yolanda and Varun Ratnakar: Trusting Information Sources One Citizen at a Time. Proceedings of the First International Semantic Web Conference (ISWC), Sardinia, Italy, June 2002.

[Hal01] Halevy, A.Y.: Answering queries using views: A survey. VLDB Journal 10 (2001) 270–294

[Harmo] www.harmonise.org

[HS02] S. Handschuh and S. Staab. Authoring and annotation of web pages in CREAM. In The Eleventh International World Wide Web Conference (WWW2002), Honolulu, Hawaii, USA 7-11 May, 2002.

[HS03] S. Handschuh, S. Staab, R.Volz: On Deep Annotation, WWW-2003, Budapest, Hungary, May 20-24, 2003

[HU79] J.E.Hopcroft, J.D.Ullmann; Introduction to Automata Theory, Languages and Computation; Addison-Wesley, 1979

[KLS95] Kirk, T., Levy, A.Y., Sagiv, Y., Srivastava, D.: The Information Manifold. In: Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments. (1995) 85–91

[KP*02] Kalyanpur, A., Bijan Parsia, James Hendler, Jennifer Golbeck. SMORE - Semantic Markup, Ontology, and RDF Editor. in Submitted to WWW2003. 2002

[KPS01] J. Kahan, M. Koivunen, E. Prud'Hommeaux and R. Swick: Annotea: Open RDF Infrastructure for Shared Web Annotations. In Proc. of the WWW10 International Conference. Hong Kong, 2001.

[Lenz02] Lenzerini, M.: Data Integration: A Theoretical Perspective, Proceedings. of PODS Conference, 2002.

[LS*97] S. Luke, L. Spector, D. Rager, and J. Hendler. Ontology-based Web agents. In Proceedings of the First International Conference on Autonomous Agents, pages 59–66. ACM, 1997.

[Miss*03] M.Missikoff et Al.; The Architecture of the Project Harmonise; Proc. Of ENTER Conference, Helsinki, Jan 2003.

[MP*97] Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J.D., Vassalos, V., Widom, J.: The TSIMMIS approach to mediation: Data models and languages. J. of Intelligent Information Systems 8 (1997) 117–132

[MT02] M.Missikoff, F.Taglino; Business and Enterprise Ontology Management with SymOntox; Proc. Of ISWC 02, Sardinia, June 2002.

[MT03] M.Missikoff, F.Taglino; The Architecture of an Ontology-based Platform for Semantic interoperability; The Handbook of Ontologies in Information Systems (S.Staab and R.Studer, Eds), Springer Verlag, to appear, 2003.

[NLF99] Naumann, F., Leser, U., Freytag, J. C.: Quality-driven integration of heterogeneous information systems. In Proceedings. of the 25th Int. Conf. on Very Large DataBases (VLDB'99), pages 447–458, 1999.

[SeWeb] annotation.semanticweb.org

[TRV98] Tomasic, A., Raschid, L., Valduriez, P.: Scaling access to heterogeneous data sources with DISCO. IEEE Trans. on Knowledge and Data Engineering 10 (1998) 808–823

[UG96] M.Uschold, M.Gruninger; Ontologies: Principles, Methods and Applications; The Knowledge Engineering Review, V.11, N.2, 1996.

[VM*02] Maria Vargas-Vera, Enrico Motta, John Domingue, Mattia Lanzoni, Arthur Stutt and Fabio Ciravegna, MnM: Ontology Driven Tool for Semantic Markup. European Conference on Artificial Intelligence (ECAI 2002). In proceedings of the Workshop Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002). Lyon France, July 22-23, 2002.

[VR02] S.Venkatasubramani, R.K.V.S.Raman: Annotations In Semantic Web. In the Eleventh International World Wide Web Conference (WWW2002)