Dynamic Integration of Medical Ontologies in Large Scale

Vít Nováček
Digital Enterprise Research Institute
National University of Ireland, Galway
IDA Business Park, Lower Dangan, Galway, Ireland
vit.novacek@deri.org

Loredana Laera
Department of Computer Science
University of Liverpool, UK
lori@csc.liv.ac.uk

Siegfried Handschuh
Digital Enterprise Research Institute
National University of Ireland, Galway
IDA Business Park, Lower Dangan, Galway, Ireland
siegfried.handschuh@deri.org

ABSTRACT

The paper presents a novel ontology lifecycle scenario that explicitly takes the dynamics and data-intensiveness of the medical application domains into account. Changing and growing knowledge is handled by semi-automatic incorporation of ontology learning results into a collaborative ontology development framework. This integration bases mainly on automatic negotiation of agreed alignments, inconsistency resolution, ontology versioning system and support of natural language generation tools, which alleviate the end-user effort in the incorporation of new knowledge.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods; I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning*

General Terms

dynamic ontology integration; ontology alignment and negotiation; ontology learning; medical ontologies; ontology-based data integration

Keywords

knowledge acquisition; life cycle; information filtering

1. INTRODUCTION

Ontologies on the Semantic Web, and especially in case of real world applications, are very likely subject to change given the dynamic nature of domain knowledge. Knowledge changes and evolves over time as experience accumulates – it is revised and augmented in the light of deeper understanding; new facts are getting known while some of the old ones need to be revised and/or retracted at the same time.

This holds especially for scientific domains – we have to incorporate newly discovered facts and possibly change the inappropriate old ones in the respective ontology as the scientific research evolves further. However, even virtually any industrial domain is dynamic – changes typically occur in product portfolios, personnel structure or industrial processes, which can all be reflected by an ontology in a knowledge management policy.

The domain of medicine is both scientific (bio-medical research) and industrial (clinical practice, pharmaceutics). The need for ontologies in bio-medicine knowledge and data management has already been reflected in the community. They can serve as structured repositories giving a shared meaning to data and thus allowing to process and query them in more efficient and expressive manner. The shared meaning also results in facilitation of integration between different medical data formats once they are bound to an ontology. Moreover, the state of the art ontology-based techniques (like alignment or reasoning) can help to integrate the data even if they adhere to different ontologies.

In the medicine domain, ontology construction is usually the result of collaboration (which involves cooperation among ontology engineers and domain experts) through a manual process of the extraction of the knowledge. However, it is not always feasible to process all the relevant data and extract the knowledge from them manually, since we might not have a sufficiently large committee of ontology engineers and/or dedicated experts at hand in order to process new data anytime it occurs. This implies a need for (partial) automation of ontology extraction and management processes in dynamic and data-intensive medical environments. This can only be achieved by ontology learning [19]. Therefore, a lifecycle of an ontology development process apt for universal application in the medicine domain should also support appropriate mechanisms for dealing with the large amounts of knowledge that are *dynamic* in nature.

1.1 Motivation

While there has been a great deal of work on ontology learning for ontology construction, e.g. [2], as well as on collaborative ontology development [24], relatively little attention has been paid to the integration of both approaches within an ontology lifecycle scenario. In this paper, we introduce our framework for practical handling of dynamic and large data-sets in an ontology lifecycle, focusing particularly on dynamic integration of learned knowledge into collaboratively developed ontologies. However, the introduced integration mechanism is not restricted only to learned ontologies – arbitrary "external" ontology can be integrated into the collaboratively designed ontology in question by the very same process.

One of the key elements supporting our integration is the ability to reach an agreement on the semantics of the terms used in these ontologies. Since the medical ontolo-

Copyright is held by the author/owner(s).
WWW2007, May 8–12, 2007, Banff, Canada.
.

gies are very often created under different circumstances and conditions and thus might represent different perspectives over similar knowledge, the process by which to come to an agreement will necessarily only come through a (partially automated) negotiation process.

The dynamic nature of knowledge is one of the most challenging problems not only in medicine, but in the whole current Semantic Web research. Here we provide a solution for dealing with dynamics in large scale, based on properly developed connection of ontology learning and dynamic collaborative development. We do not concentrate on formal specification of respective ontology integration operators, we focus rather on implementation of them, following certain practical requirements:

1. the ability to process new knowledge (resources) automatically whenever it appears and when it is inappropriate for humans to incorporate it;

2. the ability to automatically compare the new knowledge with a "master" ontology that is manually and collaboratively designed and select the new knowledge accordingly;

3. the ability to resolve possible major inconsistencies between the new and current knowledge, possibly favouring the assertions from presumably more complex and precise master ontology against the learned ones;

4. the ability to automatically sort the new knowledge according to user-defined preferences and present it to them in a very simple way, thus further alleviating human efforts in the task of final incorporation of the knowledge.

On one hand, using the automatic methods, we are able to deal with large amounts of changing data. On the other hand, the final incorporation of new knowledge is to be decided by the expert human users, repairing possible errors and inappropriate findings of the automatic techniques. The key to success and applicability is to let machines do most of the tedious and time-consuming work and provide people with concise and simple suggestions on ontology integration.

1.2 Structure of the Paper

The rest of the paper is organized as follows: Section 2 presents a brief discussion of related work. Section 3 gives an overview of our ontology lifecycle scenario and framework, whereas Section 4 presents the integration of manually designed and learned ontologies in more detail. In Section 5, we describe the current state of the work and give a simple illustrative example of concrete usage of the our integration approach. Section 6 discusses realistic medicine application domains in which our lifecycle framework can help. Section 7 concludes the paper and sums up our future work.

2. RELATED WORK

Recent overviews of the state-of-the-art in ontologies and related methodologies can be found in [23] and [13]. However, none of them offers a direct solution to the previously mentioned problems.

Methontology [10] is a methodology developed in the *Esperonto* project for designing ontologies to serve as a base for extending it towards evolving ontologies. It is provided with an ontology lifecycle based on evolving prototypes [11] and defines stages from specification and knowledge acquisition to configuration management. The particular stages and their requirements are characterised, but rather generally. The automatic ontology acquisition and evaluation methods are considered in *Methontology*, however, no distinction is made in their placement within the lifecycle. The ODESeW and WebODE suite [4] projects provide an infrastructure and tools for semantic application development/management, which is in the process of being extended for networked and evolving ontologies. However, they focus rather on the application development part of the problem than on the ontology evolution parts.

The above projects have all focused on either a single part of ontology evolution, or on a rather abstract study of the knowledge management cycle. However, mechanisms that would provide a clue on how to incorporate the dynamics into the lifecycle are typically put off only by introduction of the version management, which we find insufficient. Moreover, the need for automatic methods of ontology acquisition in data-intensive environments is acknowledged, but the place of the automatic techniques is usually not distinguished in the dynamic lifecycle settings. Our approach [21] offers a complex picture of how to deal with the dynamics in the general lifecycle scenario. Here we pay attention and develop in more detail the combination of ontology learning and manual (collaborative) development in dynamic settings.

There are more specific approaches similar to the one presented by our lifecycle framework. [6] incorporates automatic ontology extraction from a medical database and its consequent population by linguistic processing of corpus data. However, the mechanism is rather task-specific – the ontology is represented in RDF(S) format that is less expressive than the OWL language, which we use. The extraction is oriented primarily at taxonomies and does not take the dynamics directly into account. Therefore the approach can hardly be applied in universal settings, which is one of our aims.

Protége [12] and related PROMPT [22] tools are designed for manual ontology development and semi-automatic ontology merging, respectively. PROMPT provides heuristic methods for identification of similarities between ontologies. The similarities are offered to the users for further processing. However, the direct connection to ontology learning, which we find important for dynamic and data-intensive domains like medicine, is missing. Moreover, the support of collaborative ontology development and integration is also unclear.

3. DINO – A DYNAMIC ONTOLOGY LIFECYCLE SCENARIO

DINO is an abbreviation of three key elements of our ontology lifecycle scenario and framework – *Dynamics*, *INtegration* and *Ontology*. However, the first two can also be *Data* and *INtensive*. All these features express the primary aim of our efforts – *to make the knowledge efficiently and reasonably manageable in data-intensive and dynamic domains.*

Figure 1 below depicts the scheme of the proposed dynamic and application-oriented ontology lifecycle that deals with the problems mentioned in the previous sections.

Our ontology lifecycle builds on four basic phases of an

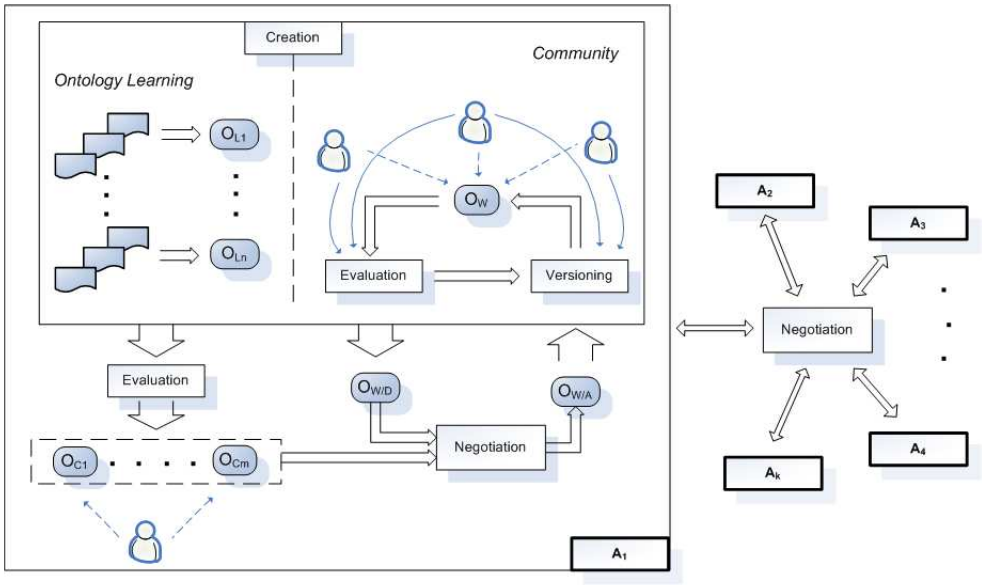


Figure 1: Dynamics in the ontology lifecycle

ontology lifecycle: *creation* (comprises both manual and automatic ontology development and update approaches), *versioning*, *evaluation* and *negotiation* (comprises ontology alignment and merging as well as negotiation among different possible alignments). The four main phases are indicated by the boxes annotated by respective names. Ontologies or their instances in time are represented by circles, with arrows expressing various kinds of information flow. The A boxes present actors (institutions, companies, research teams etc.) involved in ontology development, where A_1 is zoomed-in in order to show the lifecycle's components in detail.

The general dynamics of the lifecycle goes as follows. The community experts (or dedicated ontology engineers) develop a (relatively precise and complex) domain ontology (the *Community* part of the *Creation* component). They use means for continuous ontology *evaluation* and *versioning* to maintain high quality and manage changes during the development process. If the amount of data suitable for knowledge extraction is too large to be managed by the community, *ontology learning* takes its place. Its results are *evaluated* and partially (we take only the results with quality above a certain threshold into account) integrated into the more precise (but typically relatively small) reference community ontology. The integration is based on alignment and merging covered by the *negotiation* component. Its proposal and implementation principles form the key contribution of this paper (see Section 4 for details). The *negotiation* component takes its place also when interchanging or sharing the knowledge with other independent actors in the field. All the phases support ontologies in the standard OWL format [1], namely in its OWL DL flavour. In the following we will concentrate on the integration component. More information on other parts of the lifecycle can be found in [21].

4. DYNAMIC INTEGRATION OF THE NEW LEARNED KNOWLEDGE IN THE DINO FRAMEWORK

The key novelty of the presented lifecycle scenario is its support for incorporation of changing knowledge in data-intensive domains. This is achieved by implementation of a specific integration mechanism introduced in this section. The scheme of the integration process is depicted in Figure 2.

The integration scheme details the usage of generic lifecycle's components – mainly the *negotiation* and *versioning* – in the process of incorporation of learned ontologies into the collaboratively developed ones. However, the generic components serve only as a base for specific wrappers. Each of the phases of integration and their connections are described in the following sections.

4.1 Ontology Learning Wrapper

In this phase, machine learning and NLP methods are used for the processing of relevant resources and extracting knowledge from them (ontology learning). The ontology learning is realised using the Text2Onto framework [3]. We interface the toolbox indirectly within the collaborative ontology development portal based on MarcOnt Portal architecture (see Section 4.2). Configuration of the learning algorithms is set using a special user interface in the portal. The settings is used for batch processing of the new resources fed to the ontology learning component. The results of one round of ontology learning – the O_L circle in Figure 2 – are optionally evaluated or refined using the Text2Onto confidence values and passed to the alignment/negotiation wrapper (see Section 4.3).

Note that although we aim at integration of learned ontologies, any other "external" ontology can be provided as O_L here and passed further in the integration process, following the very same principles. Thus we can integrate e.g. different ontologies from the same medicine subdomain or specialised/general ontologies, and not strictly automatical-

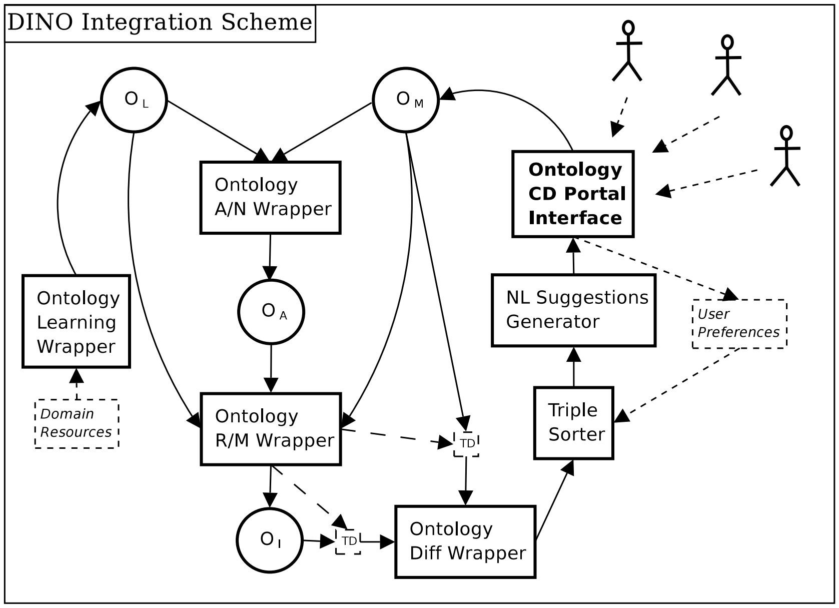


Figure 2: Dynamic integration scheme

ly learned ones.

4.2 Ontology Collaborative Development Portal

The whole integration as well as the DINO framework is based on the MarcOnt Portal architecture [15] for collaborative ontology development. It is a part of a broader initiative aimed mainly at facilitation of various digital library development and maintenance efforts[1].

MarcOnt Portal offers domain-independent means for efficient distributed and collaborative ontology development. It supports features like ontology editing, ontology versioning (supported by the SemVersion system [27]), voting on ontology changes and evaluation of these votes. The elements of DINO realising various parts of the lifecycle are being implemented into the portal's core, with access provided by respective new parts of portal's user and administrative interfaces.

The ontology being developed using the portal's collaborative interfaces is the master reference ontology in the whole lifecycle. It is also the source for deployment of official version of ontology. The O_M circle in Figure 2 represents its dump that serves as a reference to be integrated with the O_L ontology resulting from the learning process. The final suggestions (see Section 4.7) form a base for a next version of the O_M ontology submitted after the integration.

4.3 Alignment/Negotiation (A/N) Wrapper

Once the learned ontology O_L and the master ontology O_M have been created, they need to be reconciled since they cover the same domain, but might be structured differently. The reconciliation of these ontologies depends on the ability to reach an agreement on the semantics of the terms used. The agreement takes the form of an alignment between the ontologies, that is, a set of correspondences (or mappings) between the concepts, properties, and relationships in the ontologies. However, the ontologies are developed in different contexts and under different conditions and thus they might represent different perspectives over similar knowledge, so the process by which to come to an agreement will necessarily only come through a negotiation process. The negotiation process is performed using argumentation-based negotiation that uses preferences over the types of correspondences in order to choose the mappings that will be used to finally merge the ontologies (see Section 4.4). The preferences depend on the context and situation. A major feature of this context is the ontology, and the structural features thereof, such as the depth of the subclass hierarchy and branching factor, ratio of properties to concepts, etc. The analysis of the components of the ontology is aligned with the approach to ontology evaluation, demonstrated in [5], and can be formalized in terms of feature metrics. Thus the preferences can be determined on the characteristics of the ontology. For example, we can select a preference for terminological mapping if the ontology is lacking in structure, or prefer extensional mapping if the ontology is rich in instances.

Thus, the alignment/negotiation wrapper interfaces two tools – one for the ontology alignment discovery and one for negotiation of agreed alignment. We call these tools *AKit* and *NKit*, respectively, within this section. For the former, we use the ontology alignment API [8] developed by INRIA Rhone-Alpes[2]. For the negotiation we use the framework described in [17]. Both tools are used by the wrapper in order to produce O_A – an ontology consisting of axioms[3] merging classes, individuals and properties in the O_L and O_M ontologies. It is used in consequent factual merging

[1] See `http://www.marcont.org` for details on the whole MarcOnt Initiative.

[2] See `http://alignapi.gforge.inria.fr/` for up-to-date information on the API.

[3] Using constructs like *owl:equivalentClass*, *owl:sameAs*, *owl:equivalentProperty*, *rdfs:subClassOf* or *rdfs:subPropertyOf*.

and refinement in the ontology reasoning and management wrapper (see Section 4.4 for details).

The wrapper itself works according to the meta-code in Algorithm 1. The ontology alignment API offers several possibilities of actual alignment methods, which range from trivial lexical equality detection through more sophisticated string and edit-distance based algorithms to an iterative structural alignment by the OLA algorithm [9]. The ontology alignment API has recently been extended by a method for the calculation of a similarity metric between ontology entities, an adaptation of the SRMetric used in [26]. We also consider a set of justifications, that explain why the mappings have been generated. This information forms the basis for the negotiation framework that dynamically generates arguments, supplies the reasons for the mapping choices and negotiates an agreed alignment for both ontologies O_L and O_M.

Algorithm 1 Meta-algorithm of the alignment and negotiation

```
Require: $O_L, O_M$ — ontologies in OWL format
Require: $AKit, NKit$ — ontology alignment and alignment negotiation tools, respectively
Require: $ALMSET$ — a set of the alignment methods to be used
Require: $PREFSET$ — a set of alignment formal preferences corresponding to the $O_L, O_M$ ontologies (to be used in N-kit)

$S_A \leftarrow \emptyset$
for $method \in ALMSET$ do
    $S_A \leftarrow S_A \cup AKit.getAlignment(O_L, O_M, method)$
end for
$A_{agreed} \leftarrow NKit.negotiateAlignment(S_A, PREFSET)$
$O_A \leftarrow AKit.produceBridgeAxioms(A_{agreed})$
return $O_A$
```

4.4 Reasoning/Management (R/M) Wrapper

This wrapper is used for merging of the O_L and O_M ontologies. It uses Jena 2 Ontology API[4]. It merges the O_L and O_M ontologies according to the statements in O_A, preferring the lexical labels from the master O_M ontology. Moreover, the wrapper resolves possible inconsistencies caused by the merging – favouring the assertions in the O_M ontology, which are supposed to be more relevant, again. The resulting ontology O_I is passed to the ontology diff wrapper. As the Jena ontology model is internally based on a graph/triple (RDF) structure, it allows to easily export or transform an ontology in a triple format needed for the consequent wrapper (see Section 4.5 for details).

Algorithm 2 describes the meta-code of the process arranged by the ontology merging and reasoning wrapper. The inconsistency resolution is somewhat tricky. However, we can apply a sort of "greedy" heuristic, considering the assertions in the master O_M ontology to be more valid. Therefore we query the R_{ref} structure (with the axioms of learned ontology, possibly with replaced labels) in the resolution process. We currently handle the following inconsistencies:

- **sub-class** hierarchy **cycles**: these are resolved by cutting the cycle by removing an *owl:subClassOf* statement present in R_{ref};
- **disjointness-subsumption** conflicts: if classes are said to be disjoint and a sub-class relationship holds between them at the same time, the conflicting assertion indicated by R_{ref} is removed;
- **disjointness-instantiation** conflicts: if an individual is said to be an instance of classes that are disjoint, the assertion indicated by R_{ref} is removed.

Algorithm 2 Meta-algorithm of the merging and inconsistency resolution

```
Require: $O_L, O_M, O_A$ — ontologies in OWL format
Require: $getEq()$ — function selecting all assertions of type owl:equivalentClass, owl:sameAs, owl:equivalentProperty
Require: $getRM()$ — function returning wrapper combining a generic ontology manager and (incomplete OWL Full) reasoner bound to the given ontology

 1: $O_{tmp} \leftarrow copy(O_L)$
 2: $O_I \leftarrow copy(O_M)$
 3: $R_M \leftarrow getRM(O_M)$
 4: $R_{tmp} \leftarrow getRM(O_{tmp})$
 5: $R_L \leftarrow getRM(O_L)$
 6: $R_A \leftarrow getRM(O_A)$
 7: $R_I \leftarrow getRM(O_I)$
 8: $equivalencies \leftarrow \{owl : equivalentClass, owl : sameAs, owl : equivalentProperty\}$
 9: $UNIFIED \leftarrow \emptyset$
10: for $id \in getEq(O_A)$ do
11:     $R_{tmp}.replaceLabels(id.O_L, id.O_M)$
12:     $UNIFIED \leftarrow UNIFIED \cup id.O_M$
13: end for
14: $R_{ref} \leftarrow copy(R_{tmp})$
15: for $eq \in R_{tmp}.getAxiomsWithLabels(UNIFIED)$ do
16:     $R_{tmp}.retractAxioms(eq)$
17:     $R_I.addAxioms(eq)$
18: end for
19: $R_A.removeAxiomsOfType(equivalencies)$
20: $R_I.addAxioms(R_{tmp}.getAllAxioms())$
21: $R_I.addAxioms(R_A.getAllAxioms())$
22: $R_I.resolveInconsistencies(R_{ref})$
23: $R_I.augmentStructure()$
24: return $O_I$
```

When there are several removal candidate axioms involved in one inconsistency, we sort them according to the confidence provided by the Text2Onto learning algorithms [3], which is stored in the R_{ref} reference structure. Similarly to [14], we start removing the axioms with least overall confidence, until we do not resolve the inconsistency (thus keeping the more "relevant" discoveries intact). We keep the conflicting assertions when they all originate from the O_M master ontology and let the users to cope with this fact. Note that the sources of inconsistencies are provided by simple natural language description and recorded for further examinations by human users – they can eventually decide to favour the learned assertions if appropriate for the given task in the given context.

The function *augmentStructure*() attempts to complete the structure of learned axioms using the more precise and complex knowledge in the O_M master ontology. Currently, augmentation of *owl:subClassOf* and instantiation relations using *rdfs:domain* and *rdfs:range* assertions in property definitions from O_M ontology is taken into account (see Section 5 for an example). More sophisticated extensions are possible in the future.

If we want to include even the "equal" labels from the learned ontology, we can omit the renaming and subtractions in lines 10-16 and 19 and include the respective equality statements from O_A into O_I, together with respective axioms from O_L. The decision depends on users – whether they want to prefer the labels from master ontology or not (e.g.

[4] See `http://jena.sourceforge.net/ontology/index.html`.

when looking for possible unknown synonyms of important terms from O_M in domain resources; this could be useful for example in the medicine domain in task of identification of different names for the same drugs and/or proteins).

4.5 Ontology Diff Wrapper

Possible extension of a master ontology O_M by elements contained in the merged and refined ontology O_I naturally corresponds to the differences between them. These are discovered by means of the SemVersion library [27], which is interfaced within this wrapper. In particular, the possible extensions are equal to the additions O_I brings into O_M. We compute the additions from the triple-based representation[5] of O_I and O_M ontologies. The additions are passed to the triple sorter then (see Section 4.6 for details).

4.6 Triple Sorter

The addition triples passed to this component form a base to the eventual extension suggestions for the domain experts. However, the number of additions can generally be quite large, so an ordering that takes a relevance measure of possible suggestions into account is needed. Thus we can for example eliminate suggestions with low relevance level when presenting the final set to the users (without overwhelming them with a large number of possibly irrelevant suggestions).

As a possible solution to this task, we have proposed and implemented a method based on string subsumption and Levenshtein distance [18]. These two measures are used within relevance computation by comparing the predicate, subject and object lexical labels of a triple to two sets (S_p, S_n) of words, provided by users. The S_p and S_n sets contain preferred and unwanted words respectively, concerning the lexical level of optimal extensions. The general structure of the sorting function is given in Algorithm 3.

Algorithm 3 Meta-algorithm of relevance-based triple sorting

```
Require: $TRIPLES$ — list of triples
Require: $PREF = \{S_p, S_n\}$ — user preferences

$HASH = \{\}$
for $T \in TRIPLES$ do
    $HASH[getScore(T, S_p, S_n)] \leftarrow T$
end for
return $sort(HASH)$
```

The $getScore()$ function is crucial in the sorting algorithm. It is given by the formula:

$$getScore(T, S_p, S_n) = rel(T, S_p) - rel(T, S_n),$$

where $rel(T, S)$ is a function measuring the relevance of the triple T with respect to the words in the set S. The higher the value, the more relevant the triple is. The function[6] naturally measures the "closeness" of the P, S, O labels to the set of terms in S_w. The value of 1 is achieved when the label is a direct substring of or equal to any word in S_w or vice versa. When the Levenshtein distance between the label and a word in S_w is lower than or equal to the defined threshold t, the relevance decreases from 1 by a value proportional to the fraction of the distance and t. If this is not the case (i.e. the label's distance is greater than t for each word in S_W), a similar principle is applied for possible word-parts of the label and the relevance is further proportionally decreased (the minimal possible value being 0).

4.7 Mapping Triples to Natural Language Suggestions

The DINO framework is supposed to be used primarily by users who are not experts in ontology engineering. Although the MarcOnt Portal [15] already offers a simple ontology editing interface, we would like to further help the user in ontology augmentation by the learned knowledge. Therefore the suggestions are produced in the form of very simple natural language statements. These are obtained directly from the sorted triples passed to this component, using a minor modification of the generation process in CLIE described in [25]. Examples of this final form of suggestions can be found in Section 5. The suggestions are still bound to the underlying triples, therefore the user can very easily add the respective OWL axioms into the new version of the O_M master ontology without actually dealing with the intricate OWL syntax itself.

5. EVALUATION AND USAGE EXAMPLE

The DINO framework is still a work in progress[7], and thus no proper evaluation has been carried out as yet. However, preliminary evaluation of two of the core parts – negotiation and preference-based suggestion sorting techniques – has been made. The implemented sorting algorithm placed 80.7% of triples from a test sample into an order intuitively prepared by a human user. Details on the sorting evaluation are in [21, 20]. The negotiation component has been evaluated using the Ontology Alignment Evaluation Initiative test suite[8] and experiments on the impact of the argumentation approach over a set of mappings. A comparison wrt. current alignment tools is presented in [16]. The preliminary results of these experiments are promising and suggest that the argumentation approach can be beneficial and an effective solution to the problem of dynamically aligning heterogeneous ontologies.

5.1 Current State of the Implementation

We have recently completed initial draft implementation of the DINO integration technique in line with the architecture and algorithms described here. All but two presented components are fully incorporated. The negotiation (see Section 4.3) currently returns identity on input alignments – full connection with the tool described in [17] is scheduled for very near future. Implementation of function returning natural language representation of suggestions and inconsistencies is very naïve and hard-coded now, however, the working connection with general natural language generation tools mentioned in [25] should also be ready very soon.

Besides the basic implementation of the two remaining functionalities, we are currently in the phase of intensive

[5] Since SemVersion does not currently support full OWL diff computations. The triple representation is provided by the ontology R/M wrapper, as indicated by the TD (triple dump) squares in Figure 2.

[6] We described the relevance function in more detail in [21, 20], together with complexity analysis (which is in feasible class of $O(m \log(m))$ with respect to the number of triples).

[7] The current state of the implementation is summed up in Section 5.1 below.

[8] See `http://oaei.ontologymatching.org/`.

debugging and testing of the whole DINO integration proof-of-concept implementation. The testing data we take into account are mainly PubMed digital archive[9] as ontology learning resource pool and (fragments of) Galen ontology[10] as a master knowledge base.

After delivering the working implementation of the DINO integration mechanism, it will be incorporated as a library with respective user interfaces into MarcOnt Portal (see Section 4.2). This is scheduled for summer, 2007, in parallel with MarcOnt Portal reorganisation into more flexible SOA and thick-client architecture (currently being prepared by the MarcOnt Portal development team). This is the final step before deployment and evaluation of the whole DINO framework (together with the collaborative ontology development interface) in practical real-world application scenarios.

5.2 Usage Example

In the following we provide a simple illustrative example of concrete usage of the DINO integration mechanism. Imagine a medical institution that has developed an ontology O_M covering the basic concepts in clinical practice and research, possibly with help of ontology engineering experts when deploying the DINO framework. The ontology may need to be extended by new information in research (e.g. when new treatments or diagnosis methods are developed and published). Related information can be found in respective documents (research papers, industry white-papers, etc.). Figure 3 presents a sample text fragment with the respective learned OWL O_L ontology (we omit the namespace for simplicity).

The ontologies O_L and O_M are aligned and negotiated (see Figure 4). The preferences have been chosen on the basis of the ontological information of O_L and O_M (see Section 4.3 for details.

The O_M ontology and the ontology O_A, consisting of axioms produced from the negotiated mappings are shown in Figure 5.2.

When trying to merge the O_M and O_L ontologies into O_I according to the technique described in Section 4.4, we find out that there is one inconsistency – "*disease*" is said to be a subclass of "*dysfunction*" and vice versa, which creates a cycle in the taxonomy. Therefore we remove the respective "invalid" assertion that originated from the O_L ontology. On the other hand, we can extend the learned knowledge based on range and domain of the "*DiscoveredUsing*" property. We can infer new assertions on the instantiation of "*cerebellar astrocytoma*" (instance of "*Manifestation*") and "*CT*" (instance of "*DiagnosisProcedure*").

Now we can produce the triples (with O_L equivalent labels replaced by those from O_M) from the O_I merge, together with respective suggestions based on the differences between O_I and O_M. We present the sorted triples and their transformations into natural language statements[11] in Table 1.

Note that the above example may be also used if we just need to align and possibly extend the ontology with another institution's knowledge base – the only difference is that we do not perform the ontology learning and also omit retractions in the integration process, as noted in Section 4.4. This can be applied in the critical task of intermediation of medicine information, for example.

6. SELECTED APPLICATION DOMAINS

The application domains are discussed according to the use case areas identified in [7] within the EU IST 6th Framework project RIDE. The areas are rather broad, however, we can track the needs that can be at least partially covered by an appropriate ontology lifecycle framework. We do this for five selected domains in the following paragraphs. The DINO ontology lifecycle framework can serve as a substantial part of the respective semantics-enabled solutions in all of the presented application domains, since it provides complete framework for ontology creation, maintenance and mediation in data-intensive dynamic environments.

6.1 Longitudinal Electronic Health Record

The main topic here is development of standards and platforms supporting creation and management of long-term electronic health records of particular patients. These should be able to integrate various sources of data coming from different medical institutions a patient may have been treated in during his whole life.

6.1.1 Needs

Need for integration of different data sources imposes need for respective, possibly automatised, technologies able to facilitate this task. Common abstract conceptual structure of the electronic health record needs to be populated and/or extended by concrete data, present very often in unstructured natural language form. The electronic health record should also be opened to efficient and expressive querying.

6.1.2 Solutions Provided by DINO

Ontologies bound to patient data resources in particular institutions can very naturally support integration of respective data into longitudinal electronic health records. Once there is an ontology describing the underlying data, we can directly use the integration mechanism presented here in order to manage the needed integration semi-automatically. Moreover, the DINO framework can serve for easy and laymen-oriented ontology development already at the particular institutions' side. Support for ontology learning directly facilitates the population/extension. Querying of ontology-enabled electronic health records is straightforward in our framework, since it is possible using the state of the art OWL reasoning tools.

6.2 Epidemiological Registries

Epidemiology is concerned with events occurring in population – diseases, their reasons, statistical origins and their relation to a selected population sample's socioeconomic characteristic. Epidemiological registries should be able to reasonably store and manage data related to population samples and their medical attributes in order to support efficient processing of the respective knowledge by the experts.

6.2.1 Needs

The needs of this application domain can be seen as an extension of the needs in Section 6.1. Again, we have to integrate various sources of patient data, however, this time

[9] See `http://www.pubmedcentral.nih.gov/`.

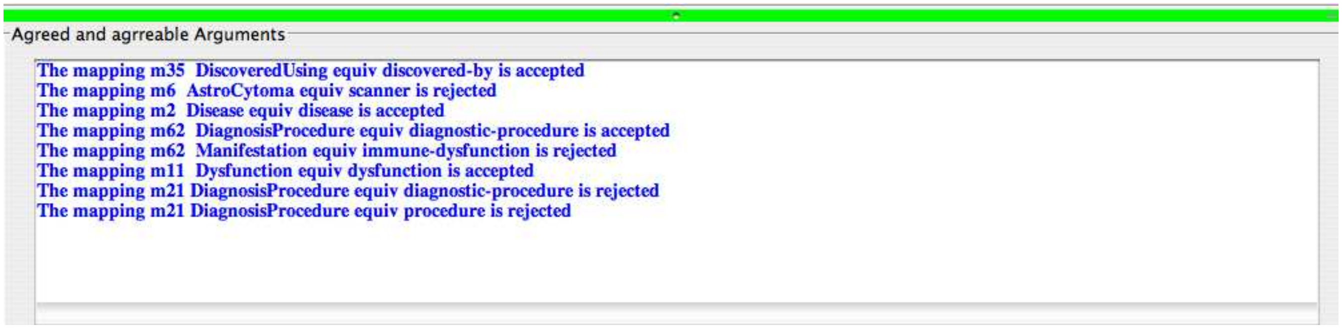
[10] Its OWL DL translation, see `http://www.co-ode.org/galen/`.

[11] They are preceded by respective sample relevance values, corresponding to {`Scanning, discover, cytoma`} and {`subclass, disease, dysfunction`} sets of preferred and unwanted terms, respectively.

. . . *while* **cerebellar astrocytoma** *is usually* **discovered by** *means of* **CT**. . . *using a* **diagnostic procedure** *of* **scanning**. . . **GVHD**, *an* **immune dysfunction**. . . *GVHD, a* **disease** *being a type of* **dysfunction**. . .

```
...
<owl:ObjectProperty rdf:ID="discovered-by"/>
<owl:Thing rdf:ID="CT"/>
<owl:Thing rdf:ID="cerebellar-astrocytoma">
 <discovered-by rdf:resource="#CT"/>
</owl:Thing>
<owl:Class rdf:ID="diagnostic-procedure"/>
<owl:Class rdf:ID="immune-dysfunction"/>
<owl:Class rdf:ID="dysfunction"/>
<owl:Class rdf:ID="scanning">
 <rdfs:subClassOf rdf:resource="#diagnostic-procedure"/>
</owl:Class>
<immune-dysfunction rdf:ID="GVHD"/>
<owl:Class rdf:ID="disease">
 <rdfs:subClassOf rdf:resource="#dysfunction"/>
</owl:Class>
...
```

Figure 3: A text sample and the learned ontology



The mapping m35 DiscoveredUsing equiv discovered-by is accepted
The mapping m6 AstroCytoma equiv scanner is rejected
The mapping m2 Disease equiv disease is accepted
The mapping m62 DiagnosisProcedure equiv diagnostic-procedure is accepted
The mapping m62 Manifestation equiv immune-dysfunction is rejected
The mapping m11 Dysfunction equiv dysfunction is accepted
The mapping m21 DiagnosisProcedure equiv diagnostic-procedure is rejected
The mapping m21 DiagnosisProcedure equiv procedure is rejected

Figure 4: Negotiated mappings

we would rather like to gather knowledge from the electronic health records to create population-wise repositories. Furthermore, when studying relations between diseases and population samples, global drug efficiency measures, etc., we need efficient mechanisms of dealing with classes and their attributes while querying the stored data.

6.2.2 Solutions Provided by DINO

Once there are ontology-enabled electronic health records (as described in Section 6.1), we can easily integrate them within another instance of "epidemiology" ontology developed in the DINO framework. The ontology representation of data in an epidemiology repository can add additional dimension to usual statistical processing of population data. Using DL-based reasoning on the data semantics expressed by the respective OWL ontologies, we could obtain additional qualitatively different (symbolic) valuable results.

6.3 Public Health Surveillance

Public health surveillance presents ongoing collection, analysis, interpretation and dissemination of health-related data in order to facilitate a public health action reducing mortality and/or improving health [7]. It has several public health functions, including estimating the impact of a disease, determining the distribution and spread of illness, outbreak detection or evaluating prevention and control measures.

6.3.1 Needs

The needs are similar to Section 6.2. However, there are important differences, as the active public health functions (e.g. outbreak detection) directly require efficient dynamic processing of newly coming data. Moreover, the need for tools able to automatically process free natural language text is explicitly emphasised in this application domain concerning the dynamic knowledge processing.

6.3.2 Solutions Provided by DINO

The basic design principles of DINO directly conform to the needs here. Ontologies created and dynamically extended by or confronted with newly coming critical data can efficiently support expert decisions in risk management tasks. Continuous integration of less critical data from various sources can back the study of public health issues in long term perspective at the same time.

6.4 Management of Clinical Trials

Briefly put, clinical trials are studies of the effects of newly developed drugs on selected sample of real patients. They are essential part of approval of new drugs for normal clinical use and present an important bridge between medical research and practice.

6.4.1 Needs

A need for electronic representation of clinical trials data is emphasised. However, even if the data are electronically represented, problems with their heterogeneity and integration occur as there are typically several different institutions involved in a single trial. Efficient querying is demanded, stating it can reduce the overall cost of clinical trials significantly.

6.4.2 Solutions Provided by DINO

```
...
<owl:ObjectProperty rdf:ID="InstrumentalProperty"/>
<owl:ObjectProperty rdf:ID="DiscoveredUsing">
 <rdfs:subPropertyOf rdf:resource="#InstrumentalProperty"/>
 <rdfs:range rdf:resource="#Manifestation"/>
 <rdfs:domain rdf:resource="#DiagnosisProcedure"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Manifestation"/>
<owl:Class rdf:ID="Procedure"/>
<owl:Class rdf:ID="DiagnosisProcedure">
 <rdfs:subClassOf rdf:resource="#Procedure"/>
</owl:Class>
<owl:Class rdf:ID="SoftTissueCytoma"/>
<owl:Class rdf:ID="AstroCytoma">
 <rdfs:subClassOf rdf:resource="#SoftTissueCytoma"/>
</owl:Class>
<owl:Class rdf:ID="Disease">
<owl:Class rdf:ID="Dysfunction">
 <rdfs:subClassOf rdf:resource="#Disease"/>
</owl:Class>
...
```

```
...
<owl:ObjectProperty rdf:ID="DiscoveredUsing">
 <owl:equivalentProperty rdf:resource="#discovered-by"/>
</owl:ObjectProperty>
<AstroCytoma rdf:ID="cerebellar-astrocytoma"/>
<owl:Class rdf:ID="DiagnosisProcedure">
 <owl:equivalentClass rdf:resource="#diagnostic-procedure"/>
</owl:Class>
<owl:Class rdf:ID="immune-dysfunction">
 <owl:subClassOf rdf:resource="#Dysfunction"/>
</owl:Class>
<owl:Class rdf:ID="Dysfunction">
 <owl:equivalentClass rdf:resource="#dysfunction"/>
</owl:Class>
...
```

Figure 5: A master ontology sample and the respective mapping

`<AstroCytoma rdf:ID="cerebellar-astrocytoma"/>`	+0.667: CEREBELLAR ASTROCYTOMA is a *new instance* of ASTROCYTOMA.
`<Manifestation rdf:ID="cerebellar-astrocytoma"/>`	+0.667: CEREBELLAR ASTROCYTOMA is a *new instance* of MANIFESTATION.
`<DiagnosisProcedure rdf:ID="CT"/>`	+0.389: CT is a *new instance* of DIAGNOSIS PROCEDURE.
`<immune-dysfunction rdf:ID="GVHD"/>`	+0.333: GVHD is a *new instance* of IMMUNE DYSFUNCTION.

`<owl:Class rdf:ID="scanning">`
`<rdfs:subClassOf rdf:resource="#DiagnosisProcedure"/>`
`</owl:Class>`
	-0.444: A *new class* SCANNING is a *sub-class* of DIAGNOSIS PROCEDURE.

`<owl:Thing rdf:ID="cerebellar-astrocytoma">`
`<DiscoveredUsing rdf:resource="#CT"/>`
`</owl:Thing>`
	-0.667: CEREBELLAR ASTROCYTOMA is DISCOVERED USING CT.

`<owl:Class rdf:ID="immune-dysfunction">`
`<rdfs:subClassOf rdf:resource="#Dysfunction"/>`
`</owl:Class>`
	-0.833: A *new class* IMMUNE DYSFUNCTION is a *sub-class* of DYSFUNCTION.

Table 1: Extension triples and the respective NL suggestions

Once again, ontologies developed and/or mediated using the DINO framework can facilitate the integration problems. Universal formal OWL representation allows unified querying of different clinical trial data then.

6.5 Genomics and Proteomics Research

Similarly to Section 6.4, this application domain is related to translational medicine and to bridging the research and clinical practice. Genomics and proteomics research studies genes, proteins, their effects, mutual influences and interactions within human organism. It covers both basic and applied medical and pharmaceutical research.

6.5.1 Needs

Integration of various knowledge repositories is needed when pursuing study in a particular sub-domain of genomics and proteomics. We may need to integrate specific knowledge e.g. in GO or UMLS controlled dictionaries[12] and in clinical reports on drug compounds and their effects in practice. Merits of efficient querying of the knowledge are obvious even in this case.

6.5.2 Solutions Provided by DINO

The ontology development and integration services, together with OWL-based formalised support for efficient reasoning, cover the needs even in this application domain to some extent. Unfortunately, there are practical limitations mainly in the lack of formal structure of genomics and proteomics knowledge bases. Their transformation into a formal ontology is thus not trivial. However, after development/adaptation and implementation of a certain methodology and rules of this translation, the semi-automatic relevance-guided integration proposed in DINO can help in this task even if the translation itself would not perform very well.

7. CONCLUSIONS AND FUTURE WORK

We have presented the basic principles of DINO – a novel framework for ontology development in dynamic and data-intensive domains like medicine. As a core contribution of the paper, we have described the mechanism of integration of learned and collaboratively developed medical knowledge. It covers all the requirements specified in Section 1.1. The proposed combination of automatic and collaborative tools in knowledge acquisition, integration and inconsistency resolution ensures production of reliable, broad and precise ontologies when using DINO in dynamics settings. The analysis of factual needs in the medicine application domains presented in Section 6 has shown that the proposed scenario we are implementing is relevant for the medicine research and clinical practice.

Our present and future work concentrates mainly on full implementation of the DINO framework by the respective extensions of the MarcOnt Portal architecture (as outlined in this paper). We also plan to continuously evaluate and improve the framework in line with demands of interested

[12] See http://www.ebi.ac.uk/ego and http://umlsinfo.nlm.nih.gov, respectively.

partners in the medicine industry (possibly, but not only within the presented application domains) and also in other applicable fields.

Acknowledgements

This work has been supported by the EU IST 6th framework's Network of Excellence 'Knowledge Web' (FP6-507482) and partially by Academy of Sciences of the Czech Republic, 'Information Society' national research program, the grant AV 1ET100300419.

8. REFERENCES

[1] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. *OWL Web Ontology Language Reference*, 2004. Available at (February 2006): `http://www.w3.org/TR/owl-ref/`.

[2] F. C. C. Brewster and Y. Wilks. User-centred onlology learning for knowledge management. In *In Proceedings 7th International Workshop on Applications of Natural Language to Information Systems, Stockholm.*, 2002.

[3] P. Cimiano and J. Völker. Text2Onto - a framework for ontology learning and data-driven change discovery. In *Proceedings of the NLDB 2005 Conference*, pages 227–238. Springer-Verlag, 2005.

[4] O. Corcho, A. Lopez-Cima, and A. Gomez-Perez. The ODESeW 2.0 semantic web application framework. In *Proceedings of WWW 2006*, pages 1049–1050, New York, 2006. ACM Press.

[5] K. Dellschaft and S. Staab. On how to perform a gold standard based evaluation of ontology learning. In *Proceedings of the International Semantic Web Conference. Athens, GA, USA.*, 2006.

[6] R. Dieng-Kuntz, D. Minier, M. Ruzicka, F. Corby, O. Corby, and L. Alamarguy. Building and using a medical ontology for knowledge management and cooperative work in a health care network. *Computers in Biology and Medicine*, 36:871–892, 2006.

[7] M. E. (edited by). Requirements analysis for the ride roadmap. Deliverable D2.1.1, RIDE, 2006.

[8] J. Euzenat. An API for ontology alignment. In *ISWC 2004: Third International Semantic Web Conference. Proceedings*, pages 698–712. Springer-Verlag, 2004.

[9] J. Euzenat, D. Loup, M. Touzani, and P. Valtchev. Ontology alignment with ola. In *Proceedings of the 3rd International Workshop on Evaluation of Ontology based Tools (EON)*, Hiroshima, Japan, 2004. CEUR-WS.

[10] M. Fernandez-Lopez, A. Gomez-Perez, and N. Juristo. Methontology: from ontological art towards ontological engineering. In *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*, pages 33–40, Stanford, USA, March 1997.

[11] M. Fernandez-Lopez, A. Gomez-Perez, and M. D. Rojas. Ontologies' crossed life cycles. In *Proceedings of International Conference in Knowledge Engineering and Management*, pages 65–79. Springer–Verlag, 2000.

[12] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubezy, H. Eriksson, N. F. Noy, and S. W. Tu. The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human–Computer Studies*, 58(1):89–123, 2003.

[13] A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho. *Ontological Engineering*. Advanced Information and Knowledge Processing. Springer-Verlag, 2004.

[14] P. Haase and J. Völker. Ontology learning and reasoning - dealing with uncertainty and inconsistency. In P. C. G. da Costa, K. B. Laskey, K. J. Laskey, and M. Pool, editors, *Proceedings of the Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, pages 45–55, NOV 2005.

[15] S. Kruk, J. Breslin, and S. Decker. MarcOnt initiative. Líon Deliverable 3.01, DERI, Galway, 2005.

[16] L. Laera, I. Blacoe, V. Tamma, T. Payne, J. Euzenat, and T. Bench-Capon. Argumentation over ontology correspondences in mas. In *In Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007), Honolulu, Hawaii, USA. To Appear*, 2007.

[17] L. Laera, V. Tamma, J. Euzenat, T. Bench-Capon, and T. R. Payne. Reaching agreement over ontology alignments. In *Proceedings of 5th International Semantic Web Conference (ISWC 2006)*. Springer-Verlag, 2006.

[18] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics Control Theory*, 10:707–710, 1966.

[19] A. Maedche and S. Staab. Ontology learning. *Handbook on Ontologies*, 2004.

[20] V. Nováček, M. Dabrowski, S. R. Kruk, and S. Handschuh. Extending community ontology using automatically generated suggestions. In *Proceedings of FLAIRS 2007*. AAAI Press, 2007. In press.

[21] V. Nováček, S. Handschuh, L. Laera, D. Maynard, M. Völkel, T. Groza, V. Tamma, and S. R. Kruk. Report and prototype of dynamics in the ontology lifecycle (D2.3.8v1). Deliverable 238v1, Knowledge Web, 2006.

[22] N. Noy and M. Musen. The prompt suite: Interactive tools for ontology merging and mapping, 2002.

[23] S. Staab and R. Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer-Verlag, 2004.

[24] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative Ontology Development for the Semantic Web. In *1st International Semantic Web Conference (ISWC2002)*, Sardinia, 2002. Springer.

[25] V. Tablan, T. Polajnar, H. Cunningham, and K. Bontcheva. User–friendly ontology authoring using a controlled language. In *Proceedings of LREC 2006 - 5th International Conference on Language Resources and Evaluation*. ELRA/ELDA Paris, 2006.

[26] B. L. S. V. Tamma, I. Blacoe and M. Wooldridge. Introducing autonomic behaviour in semantic web agents. In *In Proceedings of the Fourth International Semantic Web Conference (ISWC 2005), Galway, Ireland, November.*, 2005.

[27] M. Völkel and T. Groza. SemVersion: RDF-based ontology versioning system. In *Proceedings of the IADIS International Conference WWW/Internet 2006 (ICWI 2006)*, 2006.