A Matchmaking-based Ontology Evolution Methodology*

Silvana Castano, Alfio Ferrara, and Stefano Montanelli

Università degli Studi di Milano
DICo - Via Comelico, 39, 20135 Milano - Italy
{castano,ferrara,montanelli}@dico.unimi.it

Abstract. In this paper, we present the H-CHANGE methodology we have specifically conceived for evolving independent ontologies in open networked systems. Furthermore, we describe the change detection techniques based on semantic matchmaking for determining the most appropriate location where to frame new incoming knowledge within an existing ontology. Change assimilation techniques for evolving ontology metadata to incorporate the new incoming knowledge at different integration levels are also outlined.

1 Introduction

The introduction of the Semantic Web vision and the success of open networked infrastructures, such as P2P and Grids, have been produced a new attention to the problem of distributing information resources and data over a high number of independent peers [1]. In order to semantically enhance the capability of retrieving and sharing this large amount of resources, the use of ontology-based networked systems, either Web-based or P2P-based, are seen as a wide repository of knowledge, where peers interact each other and new concepts and data descriptions are continuously acquired by peers and/or can change during time. In such a scenario, the ontologies themselves have to be open and independent, in that concept descriptions are given also in terms of concepts defined elsewhere in the system and shared by the peers. A key requirement in such an open and dynamic scenario, is to equip the peers with appropriate methods and techniques for automatically evolving their ontology knowledge by assimilating new concepts that have been acquired from the network.
In this paper, we first describe the H-CHANGE methodology specifically conceived for evolving independent ontologies in open networked systems. Then, we focus on the change detection techniques based on semantic matchmaking for determining the most appropriate location where to frame new incoming knowledge within an existing ontology. Change assimilation techniques for evolving ontology metadata to incorporate the new incoming knowledge at different integration levels are also outlined.

* This paper has been partially funded by NoE INTEROP, IST Project n. 508011 - 6th EU Framework Programme.

The paper is organized as follows. In Section 2, we describe the H-CHANGE methodology. Matchmaking-based change detection techniques are discussed in Section 3. In Section 4, we provide some considerations on the presented H-CHANGE methodology and we discuss some applicability issues. Finally, related work and concluding remarks are provided in Section 5 and 6, respectively.

2 The H-CHANGE methodology

In open networked systems, like P2P and Grids, peers act as independent agents with their own knowledge (i.e., peer ontology) and interact each other by submitting discovery queries and by replying with relevant knowledge. Peers have no prior reciprocal knowledge and gradually discover the location and the contents of the other peer ontologies by i) submitting to the system a number of discovery queries, with one or more target concepts of interest; and ii) by analyzing the replies provided by the other peers to these queries (*independency requirement*). Moreover, each peer has the responsibility to autonomously evolve/enrich its own ontology in a consistent way according to the relevance of the incoming knowledge acquired from the network (*autonomy requirement*). To this end, the H-CHANGE methodology (see Figure 1) is defined for evolving independent ontologies in open networked systems and it is articulated in the following phases.

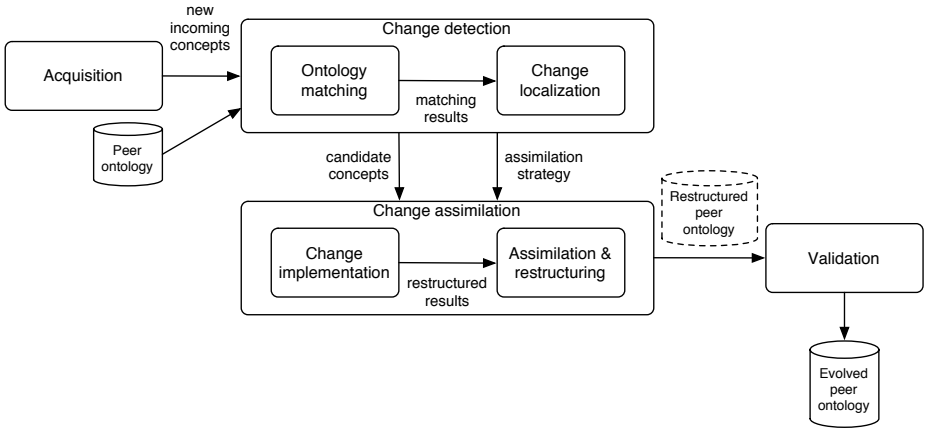

Fig. 1. The H-CHANGE methodology

1. **Acquisition**. The acquisition phase involves the definition of one or more discovery query and their submission to the system with the aim to locate the external ontologies capable of providing relevant knowledge. In response

to the submitted discovery queries, a set of new incoming concepts are received and need to be considered in order to evolve/enrich the peer ontology. New incoming concepts constitute the proposed ontology changes, already represented according to an ontological specification.

2. **Change detection**. The change detection phase has the goal of evaluating the impact of a new incoming concept on the existing peer ontology knowledge. To this end, the ontology matching and the change localization phases are defined.
   - *Ontology matching*. The ontology matching phase exploits ontology matchmaking techniques to semantically compare the new incoming concepts against the peer ontology in order to produce a set of matching results containing the discovered semantic affinities.
   - *Change localization*. The change localization phase exploits the matching results in order to detect i) the best assimilation point in the peer ontology for each new incoming concept, namely the *candidate concept*; and ii) the assimilation strategy to apply for the insertion according to a threshold-based mechanism.
3. **Change assimilation**. The change assimilation phase is responsible for applying the peer ontology changes previously detected. This phase is composed of the change implementation and the assimilation and restructuring phases, as follows.
   - *Change implementation*. The change implementation phase modifies a candidate concept by assimilating a new incoming concept in order to produce a restructured concept. The kind of restructuring operations and resulting restructured concepts depends on the selected assimilation strategy.
   - *Assimilation and restructuring*. The restructured concepts are introduced in the peer ontology and the changes become effective. In this phase, consistency check is also performed on the resulting restructured peer ontology.
4. **Validation**. The validation phase enforces conventional recovery facilities to ensure that any change performed in the change assimilation phase can be backtracked if required. In this phase, the restructured peer ontology becomes effective and a final evolved peer ontology is produces as a result. This phase can be implemented by adopting, for instance, ontology versioning techniques (see Section 6).

Expected user interactions. The H-CHANGE methodology enforces an interactive approach, allowing the designer to validate and choose among proposed alternative choices. To limit the amount of manual activity required, default choices in each phase of the H-CHANGE methodology are expected. In particular, two kinds of interactions with the designer are expected in the H-CHANGE tool support:

- *Parameter settings*: the designer is required to properly set all the involved parameters (i.e., the matching threshold in the change localization phase). A default value is provided and the designer can modify it if required.

– *Result inspection and validation*: the designer is asked to validate the results of the automated change detection and change assimilation phases. However, the proposed solution can be always modified to manually force a specific design choice if required.

As an example, we consider the peer P together with a portion of its peer ontology, called PublicationOntology, as represented in Figure 2 [1]. In this example, peer P is receiving two new incoming concepts with their related context, namely Volume and Library, in response to a discovery query. In this scenario, ontology evolu-

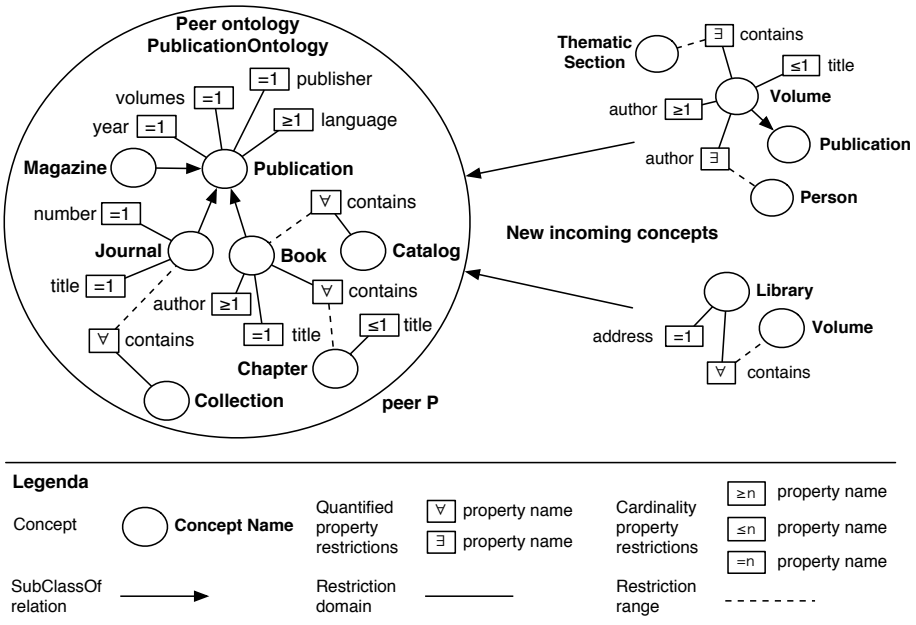

Fig. 2. The peer P with a portion of its peer ontology PublicationOntology and two new incoming concepts

tion regards the capability of peer P to autonomously decide *whether* and *how* it can acquire the new externally received knowledge, evolving its own peer ontology PublicationOntology accordingly. In order to address such requirements, the H-CHANGE methodology relies on the H-MATCH ontology matchmaking techniques we have developed in the framework of the HELIOS project for matching independent ontologies [2, 3].

[1] In Figure 2, a graphical representation of the OWL PublicationOntology is provided. The complete OWL specification is available at http://islab.dico.unimi.it/ontologies/-PublicationOntology.owl

3 Using matchmaking techniques for change detection

In this section, we first describe the H-MATCH matchmaking techniques and then we discuss the role of matchmaking for evaluating the relevance of new incoming concepts and for selecting the most appropriate change assimilation strategy.

3.1 Semantic matchmaking with H-MATCH

H-MATCH performs ontology matching at different levels of depth by deploying four different *matching models* spanning from surface to intensive matching, with the goal of providing a wide spectrum of metrics suited for dealing with many different matching scenarios that can be encountered in comparing concept descriptions of real ontologies. H-MATCH takes two ontologies as input and returns the mappings that identify corresponding concepts in the two ontologies, namely the concepts with the same or the closest intended meaning. H-MATCH mappings are established after an analysis of the similarity of the concepts in the compared ontologies. In H-MATCH we perform similarity analysis through affinity metrics to determine a measure of semantic affinity in the range $[0, 1]$. A threshold-based mechanism is enforced to set the minimum level of semantic affinity required to consider two concepts as matching concepts. Given two concepts c and c', H-MATCH calculates a semantic affinity value $SA(c, c')$ as the linear combination of a linguistic affinity value $LA(c, c')$ and a contextual affinity value $CA(c, c')$. The linguistic affinity function of H-MATCH provides a measure of similarity between two ontology concepts c and c' computed on the basis of their linguistic features (i.e., concept names). For the linguistic affinity evaluation, H-MATCH relies on a thesaurus of terms and terminological relationships automatically extracted from the WordNet lexical system. The contextual affinity function of H-MATCH provides a measure of similarity by taking into account the contextual features of the ontology concepts c and c'. The context of a concept can include properties, semantic relations with other concepts, and property values. The context can be differently composed to consider different levels of semantic complexity, and four matching models, namely, *surface*, *shallow*, *deep*, and *intensive*, are defined to this end. In the surface matching, only the linguistic affinity between the concept names of c and c' is considered to determine concept similarity. In the shallow, deep, and intensive matching, concept similarity is determined by considering both linguistic and contextual affinities. In particular, the shallow matching computes the contextual affinity by considering the context of c and c' as composed only by their properties. Deep and intensive matching extend the depth of concept context for the contextual affinity evaluation of c and c', by considering also semantic relations with other concepts (deep matching model) as well as property values (intensive matching model), respectively. The comprehensive semantic affinity $SA(c, c')$ is evaluated as the weighted sum of the Linguistic Affinity value and the Contextual Affinity value, that is:

$$SA(c, c') = W_{LA} \cdot LA(c, c') + (1 - W_{LA}) \cdot CA(c, c') \quad (1)$$

where W_{LA} is a weight expressing the relevance to be given for the linguistic affinity in the semantic affinity evaluation process.

According to the matching model definitions, we note that the surface model is suited for poorly structured ontologies with very simple concept descriptions, the shallow and the deep models are suited for dealing with schematic ontologies with taxonomic concept descriptions, and the intensive model is suited for articulated ontologies with rich concept descriptions. H-MATCH has been extensively tested on several real ontology test matching cases in order to evaluate the matching models with respect to performance and quality of results [2]. By analyzing the obtained results, we note that the most accurate and precise results are achieved with the deep and intensive matching models provided that the ontology descriptions are detailed enough. For ontology evolution purposes, we are interested in exploiting the H-MATCH results to discover the best insertion points where to assimilate a given new incoming concept. To this end, since we focus on OWL ontologies, the intensive matching model has been selected for the change detection phase, in that it guarantees the highest level of accuracy in finding the concept that best matches a new incoming concept by considering all the contextual features characterizing the involved concepts. A detailed description of H-MATCH and related matching models is provided in [2].

3.2 Change detection and assimilation

In the change detection phase of the H-CHANGE methodology, we exploit the results of H-MATCH in order to detect the candidate concept of the peer ontology that represents the best assimilation point for a new incoming concept and we choose the most appropriate strategy to adopt in the next phase of H-CHANGE for implementing the assimilation (see Figure 3). Given a new incoming concept c' and a peer ontology $\mathcal{O}$, H-MATCH returns the ordered list of the top-k matching concepts in $\mathcal{O}$ with respect to c'. For the evolution purposes, since we are interested in the best assimilation point for c' in $\mathcal{O}$, the matching concept at the top of the matching results list is always chosen as the candidate concept c for the assimilation of c'. The assimilation of c' in $\mathcal{O}$ can be implemented according to two strategies, called *assimilation-by-merging* and *assimilation-by-alignment*.

Assimilation-by-merging. The assimilation-by-merging strategy is adopted for assimilating c' in $\mathcal{O}$ when c' and the candidate concept c have a high level of semantic affinity. The idea is that a high level of semantic affinity means that c' and c have the same or strongly similar intended meaning and denote the same or similar real object in a domain. For this reason, we always import the description of c' into $\mathcal{O}$ and we choose to add c' to $\mathcal{O}$ (i.e., add operation) or to unify them into a new, unique concept $\overline{c}$ (i.e., unify operation). This strategy enforces a *heavy* assimilation, in that the peer ontology knowledge is always modified and restructured to explicitly incorporate the definition of c'.

Assimilation-by-alignment. The assimilation-by-alignment strategy is adopted when there is a lower level of semantic affinity between c' and c. In this case,

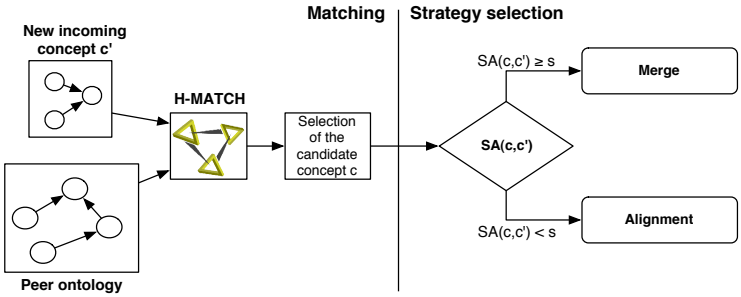

Fig. 3. The change detection phase

there is a weak correspondence between c' and c that denotes a weak similarity between the intended meaning of the two concepts. For this reason, the change represented by c' is not considered to be strong enough for modifying the current concept descriptions. However, we keep track of the change by defining a mapping m between c and c' that is defined as a triple of the form $m = \langle c, c', SA(c, c')\rangle$, where $SA(c, c')$ denotes a measure in the range (0,1] of the semantic affinity holding between c and c'. We distinguish between a local mapping (i.e., map operation), where the definition of the new incoming concept is imported as a local concept by the peer, and a distributed mapping (i.e., dmap operation), where the definition of the new incoming concept is not imported by the peer and only a reference to the external peer ontology providing c' is maintained. This strategy enforces a *light* assimilation, in that the peer ontology knowledge is kept unaltered and an appropriate mapping is set to align the peer ontology with the external knowledge provided by c'.

To enforce a semi-automated approach, in H-CHANGE the choice of the assimilation strategy is performed according to a threshold-based mechanism. A threshold s represents the minimum level of semantic affinity required in order to adopt the assimilation-by-merging strategy; based on the experimental results of matching, high values for s have to be preferred (e.g., $s \in [0.7, 0.9]$).

Example. As an example of change detection, we consider the new incoming concept Volume of Figure 2 and the peer ontology PublicationOntology of Figure 2. In this example, we set a detection threshold $s = 0.7$ and Volume is matched against the peer ontology PublicationOntology using the H-MATCH intensive model and $W_{LA} = 0.5$. The top-k matching concepts returned by H-MATCH for Volume are shown in Figure 4. The candidate concept for Volume is the top concept in the list, namely Book. Using a threshold $s = 0.7$, the assimilation-by-merging strategy is then selected. With the same procedure, we determine the candidate concept for the other new incoming concept Library of Figure 2. Since SA(Library, Collection) = 0.62, the assimilation-by-alignment strategy is selected for the new incoming concept Library.

SA(Volume,Book) = 0.875
SA(Volume,Journal) = 0.681
SA(Volume,Publication) = 0.610
SA(Volume,Magazine) = 0.539

Fig. 4. Top-k matching concepts for Volume against the peer ontology PublicationOntology

4 Considerations and applicability issues

Matchmaking-based change detection techniques have the role to evaluate the relevance of the new incoming concept with respect to the peer ontology and to suggest the most appropriate change assimilation strategy according to the level of identified semantic affinity.

When the assimilation-by-merging strategy is selected, add and unify operations are available to assimilate a new incoming concept that has a strong relevance for the peer ontology. The designer can choose the most appropriate operation between add and unify, driven by the semantic affinity value $SA(c, c')$. Concept addition is the default operation and it is always applicable for the assimilation-by-merging strategy implementation. In presence of very high values of $SA(c, c')$ the designer evaluates the opportunity of applying concept unification. The unify operation is implemented by a set of unification rules [4]. The unification rules are conceived to address all the possible cases of incompatibility or inconsistency among the definition of the two concepts to be unified. The new incoming concept is imported with a *heavy* impact on the peer ontology, thus the H-CHANGE assimilation and restructuring phase has the role to preserve the peer ontology correctness and consistency by performing a number of restructuring operations. In this phase, also the instances already present in the peer ontology need to be validated in order to verify that their descriptions are still compliant with the changed concept definitions, and appropriate techniques can be adopted to this end [5]. By handling OWL ontologies, the consistency check can be enforced by exploiting well known reasoning tools, such as Racer [6]. On the basis of the results, the designer can revise the restructured ontology until its consistency is reached.

When the assimilation-by-alignment strategy is selected, map and dmap operations are defined to assimilate a new incoming concept that has a limited relevance for the peer ontology. The type of mapping to be used for implementing the assimilation-by-alignment strategy is chosen by the designer depending on the level of matching $SA(c, c')$. A threshold-based mechanism is adopted for suggesting a default choice. By handling OWL ontologies, distributed mappings can be implemented by using a XML namespace. A local/distributed mapping keeps track of the new incoming concept with a *light* impact on the peer ontology in terms of restructuring operations that are required.

For a detailed description of change assimilation operations and techniques, the reader can refer to [4]

Applicability to open networked systems. When working in open networked systems, the choice of the correct assimilation strategy depends on the relevance of the new incoming concept for the peer ontology, and thus on the usage that the new incoming concept is expected for. As shown in Table 1, assimilation-by-merging techniques are suited for knowledge sharing scenarios where a peer is interested in extending/enriching its ontology knowledge. In this respect, add/unify operations ensure that the most relevant incoming concepts are assimilated with a high level of integration with the existing peer ontology knowledge. In this way, peers that need to cooperate on the basis of emerging and dynamic requirements are enabled to exchange their knowledge and to negotiate an agreement (i.e., a shared ontology) in order to enforce semantic collaboration and sharing. On the other side, assimilation-by-alignment techniques are suited for knowledge discovery scenarios where a peer is interested in maintaining indications about the location of external ontologies capable of providing relevant knowledge with respect to a given target. In this respect, add/dmap operations allow to align the peer ontology knowledge with the external peer ontologies by requiring a minimal modification of the peer ontology by defining a semantic reference (i.e., local or distributed mapping) between a local concept and a new incoming concept. Such references are then exploited to enforce semantic routing strategies by allowing to forward an incoming search query to those peers of the system that have similar and relevant contents in their peer ontologies with respect to the query target, thus enforcing knowledge discovery.

	Assimilation-by-merging	**Assimilation-by-alignment**
Recommended target	• Knowledge sharing	• Knowledge discovery
Assimilation operations and impact on the peer ontology knowledge	• Add/unify • Extended/enriched peer ontology knowledge	• Map/dmap • Aligned peer ontology knowledge
Enforced applications	• Semantic collaborations and sharing schemes	• Semantic routing schemes

Table 1. Applicability of the assimilation strategies

5 Related work

In recent years, ontology evolution research work has mainly focused on the problem of evaluating the impact of a modification of an ontology as a consequence of requirement changes [5, 7]. In [8], ontology evolution is defined as a complex operation that combines both organizational and technical aspects. To

this end, the authors propose a six-phase evolution process capable of dealing with the ontology changes required by the mutated business requirements. The six-phase process ensures the consistency of the ontology and possible dependent artifacts after that any modification has been applied. Such an approach has been extended in [9] in order to handle the evolution of multiple, distributed ontologies. In this respect, a pull synchronization mechanism is used to avoid possible inconsistencies while preserving the autonomy of each system node. In particular, an *evolution log ontology* is defined to track the history of changes applied to the ontology and to allow users to undo the changes that caused undesired effects. The overall approach has been implemented within the KAON infrastructure for business-oriented ontology management [2]. Another framework for ontology evolution and change management in a distributed and dynamic environment is presented in [10]. The proposed framework is based on the *ontology of change operations* for providing a formal description of the ontology modifications to be applied in order to perform a given evolution task. The ontology of change operations is defined for the OWL knowledge model and contains the *basic change operations* and the *complex change operations*. A basic operation describes the procedure for modifying only one specific feature of the OWL knowledge model (e.g., type and cardinality restriction change), while a complex operation describes an articulated change procedure and is composed of multiple basic operations. Some examples of basic and complex operations actually supported in the framework are presented. Moreover, a set of rules and heuristics are defined to create new complex operations by combining basic ones. A number of change representation formalisms are supported with the aim to foster the implementation of robust and efficient tools for ontology management. In such a context, the requirements that an ontology editor should address for ontology evolution are discussed in [11]. In particular, the authors emphasize that an editor should provide a set of evolution operations according to the supported ontology models (*functional requirement*) and that changes should be discovered semi-automatically by analyzing user behavior (*refinement requirement*). During the evolution process, the editor has to reflect the user preferences (*user supervision requirement*) by providing advanced facilities, such as change-visualization and inconsistency-detection (*transparency and usability requirements*). Moreover, history of changes needs to be supported to eventually undo any change applied to the ontology (*auditing and reversibility requirements*). Due to the complexity of the evolution task, not all the previous requirements have been fully considered for the implementation of the existing ontology evolution tools that still only provide basic functionalities. The recent success of distributed and dynamic infrastructures for knowledge sharing has raised the need of semi-automatic/automatic ontology evolution strategies. An overview of some proposed approaches in this direction is presented in [12], even if very few concrete results have appeared in literature. In most recent work, formal and logic-based approaches to ontology evolution are also being proposed. In [13], the authors provide a formal model for handling the semantics of change

[2] http://kaon.semanticweb.org/

phase embedded in the evolution process of an OWL ontology. The proposed formalization allows to define and to preserve arbitrary consistency conditions (i.e., structural, logical, and user-defined). Finally, the problem of ontology evolution can be considered as a special case of the more general and well studied problem of belief change. In [14], some of the most important concepts of the belief change literature have been revised in order to apply them to the ontology evolution context.

Original contribution of H-CHANGE. The main contribution of H-CHANGE is that it has been conceived to be suited not only for the local evolution of a peer ontology, but also for evolving independent ontologies in open and networked contexts, where distributed concepts definitions emerge dynamically through interactions of independent peers and they are maintained by the involved community of peers still through network interactions. State of the art approaches mainly focus on consistent modification of an ontology, while assuming that change detection is manually performed based on the designer domain knowledge. In this respect, an additional contribution regards the fact that H-CHANGE provides semi-automated techniques for change detection and assimilation, driven by H-MATCH matchmaking techniques.

6 Concluding remarks and future work

In this paper, we have presented the H-CHANGE methodology and related techniques for evolving open and independent ontologies in networked scenarios. In future work, we plan to integrate the H-CHANGE techniques within the HELIOS system we have developed for ontology knowledge sharing in P2P systems [3]. A prototype tool is under development, which exploits H-MATCH as a matchmaking engine to provide an interactive environment, to assist the designer in evolving a peer ontology on the basis of incoming results of knowledge discovery and sharing queries submitted to the system. Such a prototype tool will be used for extensive experimentation on real ontologies test cases with the aim to assess the effectiveness of the H-CHANGE approach under different evolution scenarios. Furthermore, ontology evolution issues are tightly connected with ontology versioning that is defined as the ability to handle changes in ontologies by creating and managing different variants of it [15]. We are working on the approaches proposed in the literature with the aim to adapt the existing solutions to be implemented in the H-CHANGE validation phase.

References

1. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Computing Surveys **36**(4) (2004) 335–371
2. Castano, S., Ferrara, A., Montanelli, S.: Matching Ontologies in Open Networked Systems: Techniques and Applications. Journal on Data Semantics (JoDS) **V** (2006) 25–63

3. Castano, S., Ferrara, A., Montanelli, S.: Dynamic Knowledge Discovery in Open, Distributed and Multi-Ontology Systems: Techniques and Applications. In: Web Semantics and Ontology. Idea Group (2006)
4. Castano, S., Ferrara, A., Montanelli, S.: Evolving Open and Independent Ontologies. Technical report, Università degli Studi di Milano (2006) Accepted for publication in *International Journal of Metadata, Semantics and Ontologies*.
5. Noy, N., Klein, M.: Ontology Evolution: Not the Same as Schema Evolution. Knowledge and Information Systems **6**(4) (2004) 428–440
6. Haarslev, V., Möller, R.: Racer: A Core Inference Engine for the Semantic Web. In: Proc. of the 2nd ISWC Int. Workshop on Evaluation of Ontology-based Tools (EON2003), Sanibel Island, Florida, USA (2003) 27–36
7. Maedche, A., Motik, B., Stojanovic, L., Studer, R., Volz, R.: Ontologies for Enterprise Knowledge Management. IEEE Intelligent Systems **18**(2) (2003) 26–33
8. Stojanovic, L., Maedche, A., Motik, B., Stojanovic, N.: User-driven Ontology Evolution Management. In: Proc. of the 13th Int. Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web (EKAW 2002), Siguenza, Spain (2002) 285–300
9. Maedche, A., Motik, B., Stojanovic, L., Studer, R., Volz, R.: An Infrastructure for Searching, Reusing and Evolving Distributed Ontologies. In: Proc. of the 12th Int. World Wide Web Conference (WWW 2003), Hungary (2003) 439–448
10. Klein, M., Noy, N.: A Component-Based Framework for Ontology Evolution. In: Proc. of the IJCAI-2003 Workshop on Ontologies and Distributed Systems, Acapulco, Mexico (2003)
11. Stojanovic, L., Motik, B.: Ontology Evolution within Ontology Editors. In: Proc. of the EKAW Workshop on Evaluation of Ontology-based Tools (EON 2002), Siguenza, Spain (2002)
12. Ding, Y., Foo, S.: Ontology Research and Development. Part 2 - a Review of Ontology Mapping and Evolving. Journal of Information Science **28**(5) (2002) 375–388
13. Haase, P., Stojanovic, L.: Consistent Evolution of OWL Ontologies. In: Proc. Of the 2nd European Semantic Web Conference (ESWC 2005), Heraklion, Crete, Greece (2005) 182–197
14. Flouris, G., Plexousakis, D., Antoniou, G.: Evolving Ontology Evolution. In: Proc. of the 32nd Int. Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 06), Merin, Czech Republic (2006)
15. Klein, M., Fensel, D.: Ontology Versioning on the Semantic Web. In: Proc.of the 1st Semantic Web Working Symposium (SWWS 2001), Stanford University, California, USA (2001) 75–91