UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

OPENKNOWLEDGE DELIVERABLE 3.1.: DYNAMIC ONTOLOGY MATCHING: A SURVEY

Pavel Shvaiko, Fausto Giunchiglia, Marco Schorlemmer, Fiona McNeill, Alan Bundy, Maurizio Marchese, Mikalai Yatskevich, Ilya Zaihrayeu, Bo Ho, Vanessa Lopez, Marta Sabou, Joaqu´ın Abian, Ronny Siebes, and Spyros Kotoulas

June 2006

Technical Report # DIT-06-046

OpenKnowledge* Deliverable 3.1.: Dynamic Ontology Matching: a Survey

Coordinator: Pavel Shvaiko[1]

with contributions from

Fausto Giunchiglia[1], Marco Schorlemmer[2], Fiona McNeill[3], Alan Bundy[3], Maurizio Marchese[1], Mikalai Yatskevich[1], Ilya Zaihrayeu[1], Bo Ho[4], Vanessa Lopez[5], Marta Sabou[5], Joaquín Abian[6], Ronny Siebes[7], and Spyros Kotoulas[7]

[1] Department of Information and Communication Technology (DIT), University of Trento, Povo, Trento, Italy
{pavel|fausto|marchese|yatskevi|ilya}@dit.unitn.it

[2] Artificial Intelligence Research Institute (IIIA-CSIC), Barcelona, Spain
marco@iiia.csic.es

[3] The University of Edinburgh, Edinburgh, UK
{f.j.mcneill|bundy}@ed.ac.uk

[4] University of Southampton, UK
bh@ecs.soton.ac.uk

[5] Knowledge Media Institute (KMi), The Open University, Milton Keynes, UK
{v.lopez|r.m.sabou}@open.ac.uk

[6] Institute for Biomedical Research of Barcelona (IIBB-CSIC), Barcelona, Spain
jambam@iibb.csic.es

[7] Vrije Universiteit Amsterdam, The Netherlands
{ronny,kotoula}@few.vu.nl

Abstract. *Matching* has been recognized as a plausible solution for the semantic heterogeneity problem in many traditional applications, such as schema integration, ontology integration, data warehouses, data integration, and so on. Recently, there have emerged a line of new applications characterized by their *dynamics*, such as peer-to-peer systems, agents, web-services. In this deliverable we extend the notion of ontology matching, as it has been understood in traditional applications, to dynamic ontology matching. In particular, we examine real-world scenarios and collect the requirements they pose towards a plausible solution. We consider five general matching directions which we believe can appropriately address those requirements. These are: (i) approximate and partial ontology matching, (ii) interactive ontology matching, (iii) continuous "design-time" ontology matching, (iv) community-driven ontology matching and (v) multi-ontology matching. We give an overview of state of the art matching systems as well as their evaluation principles from the dynamic ontology matching perspective. Finally, the key open issues and challenges towards a plausible dynamic ontology matching solution are discussed, thereby providing a vision for future activities.

* OpenKnowledge is a 3 year long STReP project financed under the European Commission's 6th Framework Programme. See, http://www.openk.org/ for details.

1 Introduction

Matching is a critical operation in many application domains, such as schema integration, ontology integration, data warehouses, data integration, and so on. It takes as input the ontologies, each consisting of a set of discrete entities (e.g., tables, XML elements, classes, properties, rules, predicates), and determines as output the relationships (e.g., equivalence, subsumption) holding between these entities [81]. Many diverse solutions to the matching problem have been proposed so far, see for surveys [8, 49, 52, 53, 70, 79, 81, 88]. Some examples of particular matching systems can be found in, e.g., [9, 15, 20, 22, 25, 31, 38, 45, 48, 51, 58, 60, 68] [1].

Matching has been viewed so far as a plausible solution to the semantic heterogeneity problem almost always in traditional applications, such as data warehouses, data integration (also known as enterprise information integration), and so on. Typically, these applications are based on a *design-time* matching operation. The explicit use of design-time matching to address the semantic heterogeneity problem is, however, more in tune with a classical codification-centered knowledge management tradition (see, for instance, the discussion in [18]). Such tradition comprises several efforts carried out mainly in the 90s to define standard upper-level ontologies [54, 66] or to establish public ontology repositories for specific domains to favour knowledge reuse [32]. Although the proliferation of efforts on ontologies have shifted the emphasis toward defining *mappings* between separately kept ontologies, design-time matching still continues this codification-centered knowledge management tradition. Corrêa da Silva and Agustí remark in [18] that "centralized ontologies [. . .] promise to bring the control of the organization back to what was possible under classical management techniques. The problem is that they may also bring back the rigidity of agencies organized under the classical management tenets."

Experience has proved that semantic interoperability is highly context- and task-dependent and that common conceptualizations or ontology mappings will rarely turn out to be generally reusable, regardless of how abstract and high-level they might be. At present, there is an emerging line of applications which can be characterized by their *openness* and *dynamics* (e.g., agents, peer-to-peer systems, web services) [81]. In these sorts of applications it is more realistic to confine semantic interoperability within the scope of an interaction (or class of interactions) between peers, achieving certain levels of semantic interoperability by matching terms dynamically. Contrary to traditional applications, such applications ultimately may require a *run-time* matching operation. This means that the corresponding matching solutions have to (besides being effective) possess some real-time performing characteristics, namely being able to discover and execute mappings in a short period of time (e.g., 2 seconds) in order to avoid having a user waiting too long for the system respond.

In this deliverable, with the help of three real-world scenarios from biomedicine, emergency response and open browsing/query answering applications, we collect the requirements they pose towards a plausible solution. We consider five general matching directions which we believe can appropriately address requirements of the dynamic applications. These are: (i) approximate and partial ontology matching, (ii) interactive

[1] See, `www.OntologyMatching.org` for a complete information on the topic.

ontology matching, (iii) continuous "design-time" ontology matching, (iv) community-driven ontology matching and (v) multi-ontology matching.

The main contributions of this deliverable are:

- an identification of dynamic ontology matching as a separate class of matching problems;
- a set of requirements from a number of real world applications towards dynamic ontology matching;
- an analytical overview of the current matching technology from the dynamic ontology matching perspective, including theoretically sound foundations of the problem, plausible matching approaches, state of the art systems and their evaluation principles.

The rest of the deliverable is organized as follows. Section 2 introduces, via examples, the conventional ontology matching problem as well as some basic motivations behind the dynamicity characteristic of the emerging applications. Section 3 overviews three application scenarios and the requirements they pose towards a matching solution. Section 4 provides a conceptual framework for comparison of dynamic ontology matching techniques and discusses in detail a number of possible alternatives. Section 5 discusses some of the relevant state of the art in matching systems and evaluation efforts. Section 6 addresses some open issues and challenges towards a plausible dynamic ontology matching solution. Finally, Section 7 reports conclusions.

2 Preliminaries

In this section we first briefly introduce the problem of ontology matching as it has been understood in traditional applications. Then, we provide a common ground for understanding the *dynamicity* characteristics of many emerging applications via an example of peer-to-peer information systems.

2.1 Ontology matching

We view *ontology matching* as the process that takes as input at least two formal structures, generically called *ontologies*[2], O_1 and O_2, and computes as output at least a third formal structure, called *alignment*, A. An alignment captures the semantic relations between O_1 and O_2's discrete entities with respect to some background knowledge K underlying the matching process. K itself may be determined by matching parameters (p) such as weights and thresholds, and external resources (r) such as common knowledge and domain specific thesauri. This definition of matching can be extended in a straightforward way to the use of an input alignment (A_0) which is to be completed by the process and also to *multi-ontology matching*, that is, when multiple ontologies are taken as input.

[2] An ontology typically provides a vocabulary that describes a domain of interest and a specification of the meaning of terms used in the vocabulary. Depending on the precision of this specification, the notion of ontology includes sets of terms (queries), classifications, thesauri, database schemas, axiomatized theories [80].

Following the work in [81], for instance, an alignment consists of a set of *mapping elements*[3]: 5-tuples $\langle id, e, e', n, R \rangle$, where id is a unique identifier of the given mapping element; e and e' are the entities (e.g., classes, properties, XML elements) of the first and the second ontology respectively; n is a *confidence measure* in the [0 1] range holding for the correspondence between the entities e and e'; R is a relation (e.g., *equivalence* (=); *subsumption* ($\sqsubseteq, \sqsupseteq$), disjointness ($\perp$)) holding between the entities e and e'.

This sort of matching assumes (in formal approaches at least) that (i) the background knowledge K is brought into the matching process by means of some particular kind of formal structure (e.g., a taxonomic hierarchy, propositional expressions, or a set of axioms in a description logic, coming thus equipped with a logical semantics); and (ii) each ontologies' local entities are put into relation with this formal structure (e.g., by means of a theory interpretation as defined, for example, for first-order logic in [28]). Ontology matching is therefore relative to the background knowledge K and to the way local vocabularies are interpreted in K.

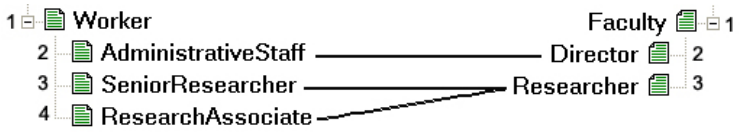


Fig. 1: Two simple ontologies and alignment

Figure 1 shows parts of two ontologies describing an academic department[4]. For example, according to some matching algorithm based on linguistic and structure analysis, the confidence measure (for the fact that the equivalence relation holds) between entities with labels Research Associate in ontology on the left, and Researcher in ontology on the right could be 0.68, thereby producing the following mapping element: $\langle id_{4,3}, ResearchAssociate, Researcher, 0.68, = \rangle$. However, the relation between the same pair of entities, according to another matching algorithm which is able to determine that the first entity is *a kind of* the second entity, could be exactly the less general relation (without computing the confidence measure). Thus, in this case, the 5-tuple $\langle id_{4,3}, ResearchAssociate, Researcher, n/a, \sqsubseteq \rangle$ is returned to the user [91]. Notice that output correspondences can have different cardinalities. In fact, the correspondence which involves node 2 in the ontology on the right represents one-to-one (1-1) relationship, while the correspondence which involves node 3 in the ontology on the right represents one-to-many (1-n) relationship.

Finally, it is worth mentioning that heterogeneity is typically reduced into two steps: (i) match the ontologies, thereby determining the alignment, and (ii) execute the alignment according to application needs (e.g., query answering, data translation). In this deliverable, we focus only on the first step.

[3] These also sometimes called mappings in the literature.

[4] A large part of the example under consideration has been taken from [91].

2.2 Peer-to-peer information management systems

Peer-to-Peer (P2P) is a fully distributed communication model in which totally autonomous parties have equivalent functional capabilities in providing each other with data and services, done in a transitive point-to-point manner [90]. Information systems (ISs), built on top of P2P architectures, offer the prospects of higher scalability, lower cost of ownership and better fault resilience and aggregate performance than traditional ISs, built on intrinsically centralized architectures [63].

A P2P IS consists of system components, also called *peers*, where each of them is logically represented by a *local information source* (LIS) and a *local ontology* (LO). The LIS of a peer stores its data, and the LO describes the part of the data *shared* with other peers. Each peer has an owner, which is independent from the owners of the other peers in the system. Owners can submit queries to the LO of their peers, and collect query results from a set of relevant peers in the system.

While answering a user query, peers *transitively* propagate the query and, subsequently, query results from one to another. To do this, peers establish a set of *acquaintances* with other peers in the system. Acquaintances are other peers that a peer knows about and which have data that can be used to answer a specific query [42]. Thus, when it receives a query, a peer answers it using the data from its LIS and propagates it to a subset of its acquaintances; they, in turn, propagate it to their acquaintances, and so on.

Because peers are totally autonomous, their LOs will almost always be mutually heterogeneous in the way they represent the same or similar concepts. This fact requires that there must be a way for a translation of a query expressed with respect to the peer's LO, to a query expressed with respect to the LO of its acquaintance. A widely adopted solution for this problem in many heterogeneous schema-based P2P data management systems is to specify a set of *mappings* between the LOs of the two peers (e.g., [14, 44, 90]). In the simplest case, a mapping defines, for an element of one LO, to which element of another LO it corresponds and what the type of this correspondence is (recall the example shown in Figure 1). In some other approaches the mappings are defined not between atomic elements but between complex structures, such as queries to the LOs (e.g., [44]). The mappings are then used for the identification of acquaintances, relevant to a query and for the query rewriting and propagation, as well as for the propagation of query results [90].

In Figure 2 we present a logical view of a P2P IS with three peers. Here, peer 2 is an acquaintance of peer 1, and peer 3 is an acquaintance of peer 2. When peer 1 processes a user query, it translates and propagates the query to peer 2 using the mappings specified from peer 1 to peer 2. When peer 2 receives the query, it computes it, sends the result back to peer 1, and then transitively propagates the query to peer 3 using the mappings from peer 2 to peer 3. In this simple example we assume that the mappings between peers are *valid*, i.e., they correctly relate corresponding elements of the peers' schemas.

The total autonomy of peers implies that the P2P IS can be highly dynamic in many aspects. Below we list some forms of autonomy and show how they affect dynamics. The reader interested in a detailed discussion of various forms of autonomy in P2P ISs and their impact on dynamics is referred to [90].

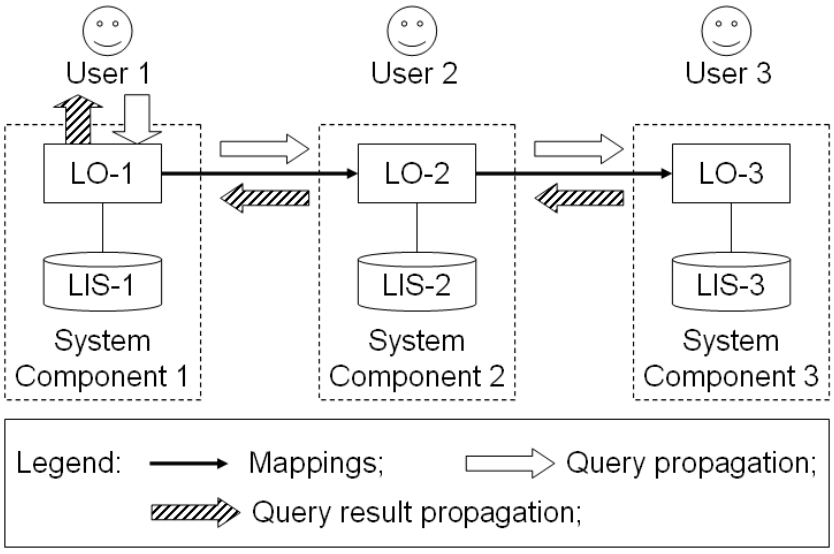


Fig. 2: A logical view of a P2P information system

- *Design autonomy:* peers are free to decide what data to store, how to describe the data, what constraints to use on the data and what interpretation to associate with the data. Peers are free to change any of these parameters at any time;
- *Association autonomy:* peers are free to decide how much of their functionality and data to share and which other peers to share them with. Peers are free to change their sharing settings at any time;
- *Participation autonomy:* peers decide when to join and when to leave the system, which logically means that they decide when to communicate and share their resources with other peers. This fact implies that the constitution of the P2P IS can be very transient – new peers may join it, and participating peers may eventually leave it.

Dynamic changes as described above happen chaotically in the P2P IS and there is no way to predict them or restrain them from happening. Also, the fact that something has changed is usually discovered at *run-time* when a certain peer receives a request for a service or a certain mapping is attempted to be used for query or query result propagation. The number of discovered changes in the system is conditional on two factors:

- *Rate of change:* which defines how often a change is introduced in the P2P IS. The rate of change associated with the design form of autonomy is inversely proportional to the sunk costs spent for setting up of data and metadata at peers. Changes associated with the association and participation forms of autonomy can occur much more often because they are of much less cost to the peers which are subject to these changes;

– *Propagation scope:* as noted above, for each user query there is a finite set of peers which answer the query. This set defines the propagation scope for the query. The size of the propagation scope of a query affects the probability that a change in the system is discovered during the query answering. The size of the P2P IS is not the only factor that defines query propagation scopes. Generally, the propagation scope for a query depends on the parameters of the query, on the peer to which the query was submitted and on the moment in time when the query was submitted [42].

The dynamics leads to two main problems: *resource discovery* and *heterogeneity resolution*. The former problem has to deal with the fact that peers may need to find new acquaintances. The latter problem deals with the fact that new mappings between peers have to be specified, or existing ones have to be reconsidered. And, while the former problem can be dealt with by exploiting sophisticated resource discovery machineries of P2P platforms such as JXTA [1], the latter represents a serious research challenge. Particularly, in order to compensate for the effect of the dynamics the P2P IS, at the *run-time* of a query, may need to:

– recompute an invalidated mapping with an existing acquaintance;
– find a new acquaintance and establish a new mapping relevant for the answering of the query;
– approximate query answers if some mappings became (temporarily) unavailable or invalid;
– (re-)compute a direct mapping between two peers, taking into consideration the mappings on an existing path between these peers.

Resolving the problems introduced by dynamics at the run-time of a query may lead to unacceptably long execution times and too poor query results. Therefore, the activities listed above can also be performed *off-line*, i.e., before query answering. In this case the system can take advantage of the fact that the execution time becomes a non-critical requirement and that the user can be involved into the process if needed. Combining off-line and run-time activities to resolve the problems introduced by dynamics is the core strategy for designing large-scale dynamic P2P IS applications.

Note that dealing with the problem of dynamics at the *run-time of a query* is what makes the crucial difference between P2P and standard data integration solutions. In the latter class of systems, dynamics are seen rather as a *planned* activity performed by a system administrator. At the time of a query submission, all the mappings in such systems are assumed to be valid and query results are therefore correct and complete. Note that the activities suggested to be performed at the run-time of a query in a P2P IS do not make sense in the context of data integration systems.

3 Motivating scenarios

In this section we discuss three concrete scenarios from dynamic applications which require ontology matching. These scenarios include: P2P proteomics (§3.1), emergency response (§3.2) [5], and open browsing/query answering (§3.3). We summarize the requirements that these applications pose towards a plausible ontology matching solution in §3.4.

3.1 P2P proteomics in OpenKnowledge

Proteomics studies the quantitative changes occurring in a proteome (which is the protein equivalent of a genome) and its application for disease diagnostics and therapy and for drug development. We shall focus here on the initial step of protein analysis, called *expression proteomics*. During this step proteins are extracted from the cells and tissues, are separated either by *two dimensional gel electrophoresis* (2DE) or *liquid chromatography* (LC) techniques, and further digested, identified and sequenced by *mass spectrometry* (MS) methods. These techniques take advantage of the current knowledge of the genome from humans and other species, which is available in public databases and can be accessed through data-mining software that relates MS spectrometric information with database sequences. Protein sequence data held in databases, however, is mostly produced from the direct translation of gene sequences. But protein activity is determined by maturation events that include so called pre- and post-translational modifications of their structure. The importance of these modifications is so high that gene and protein expression in eucariotes show no correlation in many cases.

Currently, however, technology allows the high throughput sequencing of proteomes using techniques such as *multidimensional liquid cromatography coupled with tandem mass spectrometry* (MDLC-MS/MS), which not only offer information on the proteins present in the proteome but also on their sequence (that can differ from the one in the translated databases), and type and position of their modifications. Unfortunately, MDLC-MS/MS proteomic analysis is currently an impossible task for humans to achieve. It produces a huge amount of spectra, each yielding several peptide or peptide tags candidates that can belong to the same or different proteins. Each step produces an identification score whose final evaluation (of hundreds of spectra) is performed manually or by taking high probability data.

The speed of production of this type of information is increasing very fast as a good number of proteomic laboratories being involved in the characterization of proteomes, protein complexes and networks using these strategies. Sequencing information, are especially those archives that do not produce clear identifications with the tools available to the source laboratory at a given moment, is rarely accessible to other groups involved in similar tasks and most of it will never be reflected in protein database annotation. The most probable scenario is that this information is eventually trashed. This information, however, could be of high importance for other groups analysing the sequence/function

[5] P2P proteomics and emergency response are the two main working scenarios of the OpenKnowledge project.

of this or other homologue proteins. Modification information and sequence tags generated in one lab could be used by other labs in order to evaluate the confidence of experimental or predicted sequences derived from their work in the same or other species.

We envision, therefore, a scenario in which various proteomic laboratories join a P2P network into which they feed the sequencing information generated locally at their respective labs so that other proteomic laboratories of the network can look for sequencing information in those files that proteomics laboratories deemed "useless", because they did not yield the information they required for their own particular proteomic analysis. This is particularly so with the mass spectra themselves, as no mass spectrum database is currently available, and spectra whose sequences do not give hits in a database search are trashed.

The level of semantic heterogeneity in P2P information sharing in expression proteomics is currently not at the level of sequence tags or mass spectra. However, there are a high number of proteomics technologies, each of which has developed several approaches and analytical conditions, and although proteomics facilities in Spain, for instance, are using up to seven different types of mass spectrometers producing their own file formats and using different procedures for raw data management, standard procedures for mass spectra interchange have already been proposed, such as mzXML [69] or HUP-ML[50]. Semantic heterogeneity arises at the annotation level of mass spectra and sequence tags. Annotation information ranges from the identification of the organism, cell, organelle or body fluid from which the analysed sample was extracted, to the name of the identified peptide/protein of a mass spectra or sequence tag.

Although most of this annotation will usually not yield semantic mismatches between proteomics laboratories, it may nevertheless be the case that such semantic mismatches need to be addressed. This is particularly the case with protein names, due to the variability of terms used for identical sequences. For example, the protein *lymphocyte associated receptor of death* has several synonyms including LARD, Apo3, DR3, TRAMP, wsl, and TnfRSF12. Researchers often use different names to refer to the same protein across sub-domains. Semantic matching techniques will need to dynamically resort to external sources, such as scientific publications in which protein equivalences have been identified, in order to overcome this sort of semantic mismatches. They will also need to be capable of disambiguating homonyms, i.e., two or more protein names spelled alike but different in meaning [89].

Success of peptide and protein identification depends on database and file quality, in terms of non-annotated protein sequences, database errors in sequence annotations, post-translational modifications, protein mixtures, etc. It is necessary to get as much good quality precision and recall of hits in public databases and peer files as possible, as this increases the confidence in the identification of the proteins of the analysed sample. Still, matching of sequence annotation may be approximate and partial and still be valuable for the task of protein identification: a sequence tag taken from a human tissue, for instance, matching that of a protein coming form the sample of a rat's kidney still may provide high confidence measure to the identification task as both organism are mammals.

3.2 Emergency response in OpenKnowledge

This scenario describes the situation in which a fire engine is fighting a fire and requires more assistance from other fire engines. The fire engine has a peer on board which can be accessed by the firemen as necessary. One option the firemen would have in this situation is to contact a control centre and get them to find a nearby fire engine to send to help. However, in an emergency scenario, lines of communication may be unreliable, the control centre may be swapped with requests for assistance, or be otherwise unable to help. In this case, the fire engine peer would broadcast a request for help to all other fire engine peers. Any fire engines that expected to have to work together would ensure that they were using mutually comprehensible interaction models (IMs) [6] and so communication would be straightforward. However, in an unexpectedly large disaster it may be that fire engines from a different region were called in and they may have slightly different or outdated expectations of the interaction.

This scenario is formalized with the help of Lightweight Coordination Calculus (LCC) [73] as follows:

```
a(find_available_peers(Relevant_peers,Peer_info,Final_peer_info),RFE)::
  (
   check_suitability(fire(Size,Type,Urgency),Location) => a(fire_engine_peer,Peer1)
                        <- first_peer(Peer1,Rest_peers,Relevant_peers)
                           & size_of_fire(Size) & type_of_fire(Type)
                           & urgency_of_fire(Urgency) & location(Location)
                                                                  then
     (
      available(Location1) <= a(fire_engine_peer,Peer1)
                                                                  then
      a(find_available_peers(Rest_peers,[[Peer1,Location1]|Peer_info],
                                                      Final_peer_info),RFE)
     )
    or
     (
      not_available <= a(fire_enginge_peer,Peer1)
                                                                  then
      a(find_available_peers(Rest_peers,Peer_info,Final_peer_info),RFE)
     )
    )
  or
    null <- Peer_group = [] and Peer_info = Final_peer_info
```

In the IM above, note that $a(rolename, PeerID)$ refers to a particular peer playing the specified role, double arrows ($\Leftarrow; \Rightarrow$) refer to message passing and single arrows ($\leftarrow; \rightarrow$) refer to constraints on and effects of the message passing. The example above shows a portion of an IM for the requesting fire engine peer (RFE) which details how it finds which fire engine peers are available. The role $find\_available\_peers$, which is taken on by the RFE, takes three arguments: (i) a list of relevant peers, (ii) a list of information about their whereabouts, which is initially empty and is built up during the performance of the role as the peer gathers information, and (iii) a final peer information list which is uninstantiated until the end of the procedure, when it takes the value that

[6] An IM is a set of protocols, one for each role in the interaction, which describe the messages that must be passed and received by the peer playing that role, and the constraints on and effects of those messages. The example in this section shows a part of an IM detailing the protocol of only one of the roles.

has been built up in the peer information list. Firstly, the RFE will choose the first peer from the list and send it a request to return information about its location if it is suitable for it to attend. Information about the fire ($size$, $type$, $urgency$ and $location$) is given so that the fire engine peer can determine its suitability. The RFE then waits for a response: if this response is positive (the fire engine is available) it adds the fire engine to its list of peers, together with its location, and recurses to check with the other peers. If the response is negative, the RFE behaves in the same way except that it doesn't add the fire engine peer to its list of peers. The recursion continues until it is not possible to find the first peer in the list: i.e., the list is empty. The value $Final\_peer\_info$ is then instantiated. Note that this portion of the IM shows only a part of what the RFE must do. After it has gathered the names of all potential helpers, it must then decide which to summon.

In this scenario there are three areas in where matching may be necessary:

- Role identifiers;
- Constraints on messages;
- Message contents.

Role identifiers indicate what kind of role peers may be performing, and these have to be mapped to peer's expectations of what this name may be. For example, a peer on a fire engine may consider itself to be a $fire\_truck\_peer$ or $fire\_emergency\_assistant$ and thus will not immediately identify with the role name $fire\_engine\_peer$, which is the name of the peer with which the example interaction above is taking place (the role description for this peer is not included in this example for the sake of brevity). Therefore, a matching (notice, which includes an option of approximation) between these roles is needed.

Once a peer has identified the role it wishes to play, it must ensure that it can fulfil the constraints on the messages that must be passed whilst playing that role. This entails firstly that it must be able to interpret the constraints and secondly that it must be able to satisfy them. The matching process must assist in the interpretation aspect of this, but whether or not a peer can satisfy the constraints depends on its abilities and knowledge. In the section of role description above, there are constraints on the first message. The first constraint $first\_peer(Peer1, Rest\_peers, Relevant\_peers)$ is intended to identify the first peer from a list of peers and the other four constraints are intended to instantiate variables that must be passed to the $fire\_engine\_peer$. For example, a $fire\_engine\_peer$ may expected a constraint $fire\_size(Size)$, which must be mapped to $size\_of\_fire(Size)$, or it expected a constraint $attributes\_of\_fire(Size, Type)$, which is an amalgam of the size and type constraints, or perhaps it has a more sophisticated notion of what type a fire can have, such as $type\_of\_fire(Source, Risk\_level)$. Alternatively, the peer may have expected more or fewer constraints than are present.

Message content cannot be fully determined by observation of the IM; this can only be discovered during the interaction. The IM may give detail about the format of the message: in the first message of the example above, for instance, it is determined that the message must take two arguments and that the first argument will be a relation $fire$ which will take three arguments. Nevertheless, however much structure a message may be given, it will still contain variables that are not instantiated until interaction

commences, and these variables may be instantiated in various ways. In this example, the $fire\_engine\_peer$ will know that the first message it receives will be of the form $check\_suitability(fire(Size, Type, Urgency), Location)$, but it cannot determine exactly how the variables $Size$, $Type$, $Urgency$ and $Location$ will be instantiated. It is therefore necessary to match message contents on-the-fly during interaction. The difficulty of doing this will depend on the complexity of the message.

Matching is not always a precise process: e.g., $type\_of\_fire(Source, Risk\_level)$ is only an approximate match to the constraint $type\_of\_fire(Type)$. It will often be the case that the matches found are not exact matches; they may be only approximate or partial matches because there is no guarantee that different sources (such as different fire brigades) will choose to represent things to the same level of detail or consider the same aspect of the problems. If a mapping can be found that allows sufficient information to instantiate a constraint found in the LCC then this is considered to be good enough. For example, if the fire engine peer knows that $Type$ and $Source$ are related or equivalent then it can use its knowledge of how to instantiate $type\_of\_fire(Source, Risk\_level)$ to instantiate $type\_of\_fire(Type)$: $Source$ is matched to $Type$ and $Risk\_level$ is ignored. Since the peers are engaged in interaction, they are also able to use the interaction to negotiate about matching if necessary. For example, the fire engine peer may not know anything about the concepts $Source$ or $Risk\_level$ but, from matching on the predicate name $type\_of\_fire$, could infer that one or both of them might somehow be connected to $Type$. It could then initiate a discussion with the other peer as to the meaning of these terms: if the other peer can provide some context about the concepts, the fire engine peer may be able to match them into its own concept hierarchy.

3.3 Open browsing and query answering

These scenarios are based on experience with the Magpie [24] and AquaLog/PowerAqua [55, 56] tools. Magpie aims to support web browsing by exploiting semantic data associated with the viewed web pages. AquaLog is an ontology based question answering system. Let us discuss in turn the scenarios where these tools are operating.

Magpie makes use of the semantic annotation associated with a web page to help the user get a quicker and better understanding of the information on that web page. Concretely, given an ontology and a set of instances belonging to this ontology, Magpie is able to recognize instances in the web pages viewed by the user. When one concept is selected (e.g., Person) all its instances in the web page are highlighted. Using this mechanism, users can quickly identify interesting items in the page being viewed. Additionally, for each identified instance a set of services is available, e.g., show all publications of a selected person. Magpie allows the user to choose the appropriate ontology from a list of ontologies that are known to the tool. However, the current version relies on a single ontology active at any moment in time. In order to extend the current approach towards open browsing, it is necessary to be able to select, *at run-time*, the appropriate ontologies for the given browsing context. Thus, ontology matching is needed to identify correspondences between a set of terms that describe the topic of the actual web page and available on-line ontologies.

Let us consider an example. The following short news story is both about trips to exotic locations and talks.

"For April and May 2005, adventurer Lorenzo Gariano was part of a ten-man collaborative expedition between 7summits.com and the 7summits club from Russia, led by Alex Abramov and Harry Kikstra, to the North Face of Everest. This evening he will present a talk on his experiences, together with some of the fantastic photos he took." [7]

An ontology that covers such terms as *adventurer, expedition, talk* and *photos* should be selected (discovered) from the web. This requires that the above mentioned terms are matched to the corresponding terms from the available on-line ontologies. Our preliminary experiments with Swoogle[8] have shown that queries containing (equal or) more than three terms drawn from different topic domains are likely to retrieve no ontologies if only string-based matching algorithms are used to match between the query terms and the concept labels of ontologies. Therefore, more sophisticated techniques are needed to broaden the scope of the query and ensure that relevant semantic data is found. In other words, if some of the search terms cannot be found in an ontology, mappings to more/less general concepts in the ontology are acceptable. This means that matching is required to have an option of being approximate. Finally, it is worth noting that not all the entities of the ontology need to be involved in matching. It is sufficient to consider only those entities that are similar to the query terms. This means that matching can be partial.

AquaLog is an ontology based question answering system. To concisely give an impression of how the system operates, let us consider an example. Suppose that the system is aware of an ontology about academic life[9] which has been populated to describe KMi related knowledge[10]. Also, let us suppose that the following query is posed to the system: *Which projects are related to researchers working with ontologies?* To answer this query, Aqualog needs to interpret it in terms of entities available in the system's ontology. For this, Aqualog first translates this query into the following triples: $\langle projects, related\ to, researchers \rangle$ and $\langle researchers, working, ontologies \rangle$. Then it attempts to match these triples to the concepts of the underlying ontology. For example, the term *projects* should be identified to refer to the ontology concept *Project* and *ontologies* is assumed equivalent to the *ontologies* instance of the *Research-Area* concept. In general, the matching problem exists here since the user specifies her/his query by using her/his own terminology, which is unlikely to be identical to the one of the system.

Currently, the scope of AquaLog is limited by the amount of knowledge encoded in the ontology of the system. A new version of AquaLog, called PowerAqua [55], extends its predecessor (as well as some other systems with similar goals, such as OBSERVER [61]) towards open query answering. PowerAqua aims to select and aggregate information derived from multiple heterogeneous ontologies on the web. Matching constitutes the core to this selection task. Notice that, unlike AquaLog, matching is now performed between the triples and *many* on-line ontologies (not just one ontology of the system).

[7] `http://stadium.open.ac.uk/stadia/preview.php?s=29&whichevent=657`

[8] `http://swoogle.umbc.edu/`

[9] `http://kmi.open.ac.uk/projects/akt/ref-onto/`

[10] This populated ontology can be browsed through a semantic portal at: `http://semanticweb.kmi.open.ac.uk`

This matching also differs from what is required by Magpie: it is performed between a set of triples and on-line ontologies and not just a set of (sometimes, random) terms from a web page and ontologies. Our experiments with Swoogle have shown that identifying relations that link the subject and the object of the triples is rather difficult and decreases the chances of finding relevant ontologies. For example, we found no ontologies on the web to cover the triples mentioned in the example above. However, it is not necessary to match all query triples within one ontology. When no ontology concept is found for the element of a triple, the use of more general concepts is also acceptable. This means that matching can be approximate. Similar to the case of Magpie, it is not necessary to try to match the whole ontology against the query, but only the relevant fragments. This means that matching can be partial.

3.4 Requirements

Let us summarize requirements of the dynamic applications discussed in §3.1-§3.3, see Table 1. We present them according to *functional*[11] and *non-functional*[12] views on the requirements.

Functional requirements are discussed according to the dimensions of *input*, *process*, *output* and *ultimate goal* of matching:

- Input is concerned with the (i) knowledge representation formalism, e.g., OWL [83], RDF [12], FOAF [13], and (ii) type of knowledge, namely either schema-level, or instance-level, or both, that a matching system is expected to handle.
- Process is concerned with some specific behavior of matching, such as options for (i) being approximate, (ii) being partial, and (iii) being performed at run-time.
- Output is concerned with (i) the cardinality of the resulting alignment, namely, whether it is 1-1, 1-n, or n-m, correspondences that are needed; (ii) the type of relation which stands for the discovered correspondence, namely, whether it is equivalence, subsumption, or any other types of relations, e.g., disjointness, that are needed; (iii) the output format, e.g., OWL [87], C-OWL [11], SKOS [62], SWRL [47] in which the discovered alignment is expected to be specified.
- Ultimate goal stands for the type of processing the discovered alignment needs in order to finally make the application operational. Some examples of the ultimate goal of matching exercise include query answering, navigation on the web, data translation, and so on.

Non-functional requirements are discussed according to the *matching performance*, *quality of matching results* and *number of peers* dimensions. Matching performance is concerned with the execution time and main memory indicators. Quality of matching results is concerned with the standard measures of correctness (precision) and completeness (recall). Number of peers estimates quantitively how many of peers can potentially be involved in the scenario.

[11] Functional requirements specify specific behavior or functions available from a system in order to implement a scenario (business case).

[12] Non-functional requirements specify criteria (technical constraints) that can be used to judge the operation of a system to be developed.

	P2P proteomics	Emergency response	Open browsing/ query answering
Input			
language	mzXML, HUP-ML	LCC	OWL, FOAF
data instances	√		√
Process			
approximate	√	√	√
partial	√	√	√
run-time (ideally)		√	√
negotiation of mismatches		√	
Output			
1-1, 1-n, n-m	1-1, 1-n	1-1, 1-n	1-1, 1-n
type of relation	$=; \sqsubseteq; \sqsupseteq; \perp$	$=; \sqsubseteq; \sqsupseteq; \perp$	$=; \sqsubseteq; \sqsupseteq$
output format	C-OWL	C-OWL	C-OWL
Ultimate action	query answering	semantic coordination	navigation, query answering
Quality of alignment			
precision (low, medium, high)	medium	high	high
recall (low, medium, high)	medium	medium	low
Matching performance			
execution time, sec.	$< 10^4$	< 0.5	< 1
main memory, MBs.	no limit	< 128	< 256
Number of participating peers	$\sim 10^2$	$\sim 10^2 + \sim 10^4$	$\sim 10^3$

Table 1: A summary of the requirements from the scenarios under consideration

Below we discuss in some detail non-functional requirements (since these were not a part of scenarios descriptions) based on the emergency response example from §3.2.

Quality of answers and performance. Poor or slow information can be fatal; on the other hand, a reasonable relaxation of what to do may be better than doing nothing. The components themselves should contain some indication of the necessary quality of matching: in some situations, this should be very high, and in other situation it could be lower. For example, precision is more important than recall. Indeed, we do not need to retrieve all relevant information as long as what is retrieved is correct.

Number of peers. There is a potentially large number of peers who may become involved. These are:

- *Institutional*, such as fire centres, hospitals, etc., probably less than 20;
- *Local responder teams*, such as fire engines, ambulance crews, etc., larger than the number of institutions but probably less than 100;
- *Specialised resources/services*, which could be a wide number of things, depending on the type of emergency: for example, the doctor of an injured person with a particular medical history. Numbers could range from none to hundreds, depending on the circumstances;
- *Public at large*, who can provide information on the ground to emergency services. Numbers could be large, potentially into the thousands;
- *Sensors and devices*, which, depending on the circumstances, could be providing important information from within buildings. Numbers could be into the thousands or hundreds of thousands.

4 Dynamic ontology matching

From the functional perspective, more precisely in terms of inputs/outputs, in dynamic settings the matching problem remains the same as stated in §2.1. The major difference is in the matching process, namely the addition of at least three knobs in the matching process. These are the requirements of approximate (α), partial (π) and run-time (ρ) matching. The dynamic matching process is schematically presented in Figure 3.

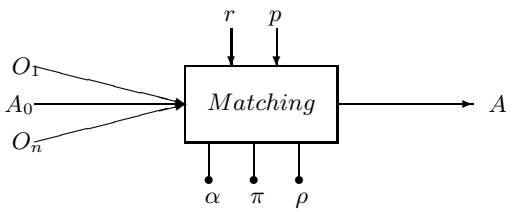


Fig. 3: The dynamic matching process

From the non-functional perspective, the major differences between ontology matching in dynamic applications and ontology matching in traditional ones include the matching performance and quality of matching results requirements.

Therefore, the core of dynamic ontology matching approaches is in addressing those functional and non-functional requirements. We envisage at least two directions for such dynamic ontology matching approaches. These are based on:

- exploiting a possibility of discovering an incomplete alignment, and/or of matching with relaxed quality results (thereby yielding some efficiency) which are still good enough for the application. We group such approaches under the heading of *approximate and partial ontology matching*. Notice, these approaches are different from those used in traditional applications. For example, in traditional applications mistakes in matching results are not acceptable, hence, mappings have to be correct and complete. Therefore, ontology matching approximation techniques gain a particular importance in dynamic settings.
- involving and exploiting the participating parties in the matching process, thereby resulting in *interactive ontology matching*. In this case, when there are mismatches, it is perhaps possible to *negotiate* them among the involved parties, still in an automated way. These approaches are different from traditional ones because in the latter there is no possibility for negotiation at all. In fact, there is only one party, namely database/knowledge base administrator, who fixes the mismatches, while in dynamic settings other parties, such as multiple agents, can participate in this process.

There are a number of plausible matching approaches which are used in traditional applications. These, under some reasonable assumptions, can also be adapted or can provide some support for ontology matching in dynamic settings. Thus, based on the

dynamic application peculiarities, we also assume that (i) changes in the ontologies of the involved parties will be evolutionary (and not revolutionary), which is quite reasonable; and that (ii) in these emerging scenarios there are multiple agents (not only, a single, e.g., database/knowledge base administrator, as in the traditional applications). We envisage a number of possible relevant matching alternatives which are worth being considered here. These are:

- performance of *updates* of mappings between the ontologies of necessary parties [13] when the actual application is idle. We group such solutions under the heading of *continuous "design-time"* matching. Similarly to traditional applications, where matching is performed at design-time, which means that mappings are precompiled, the same (precompiling) strategy can contribute (to some extent) to dynamic ontology matching. However, updates of mappings in this case should happen more frequently according to the dynamics of the application.
- involvement and exploitation of the participating parties in the matching process in order to re-distribute the workload of producing the alignments. For example, it is possible to perform and exploit annotations of mappings, thereby enabling mappings' recommendation and propagation mechanisms. We group such solutions under the heading of *community-driven (social) ontology matching* [91]. Eventually, once an alignment has been determined, it can be saved, annotated and further reused as any other data on the web. Thus, a (large) repository of annotated mappings has a potential to increase the effectiveness of matching systems by providing yet another source of domain specific knowledge besides, e.g., WordNet. In this sense this approach can be used in traditional applications. However, this approach seems promising and worth being taken into account in dynamic settings at least for the following two reasons. Firstly, it enables a set of repositories of domain specific (subjective) knowledge. This is of high importance since lack of background knowledge and, most often, domain specific knowledge is one of the main problems of state of the art matching systems these days [39]. Secondly, this approach can be naturally combined with *interactive ontology matching* in the sense of negotiating (e.g., logically contradictory) mappings.
- involvement in the matching operation of all the relevant ontologies to be matched, as opposed to a pair-wise ontology matching. We group these approaches under the heading of *multi-ontology matching*. The rationale behind these solutions is that it is often the case that matching is reduced to a simple term to term matching. Therefore, the context in which terms occur is often opaque, and hence, by taking as input more ontologies (presumably, from the same application domain, e.g., book selling), it is more likely to guess the context, see, e.g., [45]. Of course, these approaches are used in traditional applications as such. However, they are worth being re-considered in dynamic settings since, as in the item above, they elaborate on the problem of the lack of background knowledge in ontology matching.

[13] Notice, it is assumed here that these necessary parties have already been located.

4.1 A conceptual framework

We shall now sketch a formal framework in which to provide precise characterizations of the basic concepts at work in ontology matching in general, and dynamic ontology matching in particular. It takes its inspiration from the mathematical theories of logic and the flow of information in distributed systems [7, 35, 43].

We call *vocabulary* the set V of words and symbols used by a system to represent and organise its local knowledge, such as XML elements, properties, roles, classes, and so on. The *language* is then the set $L(V)$ of all well-formed formulae over a given vocabulary V, i.e., grammatically correct complex expressions based on the language's vocabulary (e.g., concept descriptions, first-order sentences, DB queries). We shall also write L when we do not want to explicitly refer to the underlying vocabulary language L. We call the elements of a language L *sentences*.

In formal, logic-based representation languages, knowledge is represented and organised by means of theories. DL-based ontologies are such an example. A convenient way to abstractly characterise theories in general is by means of the notion of consequence relation. Given a language L, a *consequence relation* over L is, in general, a binary relation $\vdash$ on subsets of L which satisfying certain structural properties [14]. Consequence relations are also suitable to capture other sorts of mathematical structures used to organise knowledge in a systematic way, such as taxonomic hierarchies. When defined as a binary relation on L (and not on subsets of L), for instance, it coincides with a partial order. Furthermore, there exists a close relationship between consequence relations and classification relations (which play a central role in ontological knowledge organisation), which has been thoroughly studied from a mathematical perspective in [7, 23, 35].

We call a *theory* a tuple $T = \langle L_T, \vdash_T \rangle$, where $\vdash_T \subseteq \mathcal{P}(L_T) \times \mathcal{P}(L_T)$ is a consequence relation, hence capturing with this notion the formal structure of an ontology in general. Finally, in order to capture the relationship between theories, we call a *theory interpretation* a map between the underlying languages of theories that respects consequence relations. That is, a function $i : L_T \rightarrow L_{T'}$ is a theory interpretation between theories $T = \langle L_T, \vdash_T \rangle$ and $T' = \langle L_{T'}, \vdash_{T'} \rangle$ if, and only if, for all $\Gamma, \Delta \subseteq L$ we have that $\Gamma \vdash_T \Delta$ implies $i(\Gamma) \vdash_{T'} i(\Delta)$ (where $i(\Gamma)$ and $i(\Delta)$ are the direct set of images of Γ and Δ along i, respectively)[15].

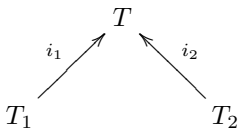
[14] These are commonly those of Identity, Weakening and Global Cut, see [7, 23].

[15] Theories and theory interpretations as treated here can also be seen as particular cases of the more general framework provided by institution theory and thoroughly studied in the field of algebraic software specification, see [43].

Paraphrasing now the problem statement given in Section 2.1, we shall call *ontology matching* the process that takes (at least) two theories T_1 and T_2 as input —called *local theories*— and computes (at least) a third theory $T_{1\leftrightarrow 2}$ as output —called a *bridge theory*— that captures the semantic alignment of T_1 and T_2's vocabularies with respect to some reference theory T underlying the semantic interoperability of T_1 and T_2 achieved by the matching process[16].

Next, we provide precise definitions of what we mean for a bridge theory to capture a semantic alignment of vocabularies. This will be important later when we attempt to situate approaches to dynamic ontology matching with those discussed in the beginning of Section 4 within the context of our formal conceptual framework. For this reason we need first to introduce the formal notion of *semantic integration*.

Definition 1. *Two theories T_1 and T_2 are* semantically integrated with respect to T*, if there exist theory interpretations $i_1 : T_1 \to T$ and $i_2 : T_2 \to T$.*



We call $\mathcal{I} = \{i_i : T_i \to T\}_{i=1,2}$ the semantic integration *of* local theories T_1 *and* T_2 *with respect to* reference theory T.

In ontology matching we are interested in determining the semantic relationship between languages L_{T_1} and L_{T_2} on which semantically integrated theories T_1 and T_2 are expressed. Therefore, a semantic integration $\mathcal{I}$ of T_1 and T_2 with respect to a reference theory T as defined above is not of direct use yet. What we would like to have is a theory $T_{\mathcal{I}}$ over the combined language $L_{T_1} + L_{T_2}$ expressing the semantic relationship that arises by interpreting local theories in T. We call this the *integration theory* of $\mathcal{I}$, and it is naturally defined as the inverse image of the reference theory T under the sum of the theory interpretations in $\mathcal{I}$. Following are the precise definitions.

Definition 2. *Let $i : T \to T'$ be a theory interpretation. The* inverse image *of T' under i, denoted $i^{-1}[T']$, is the theory over the language of T such that $\Gamma \vdash_{i^{-1}[T']} \Delta$ if, and only if, $i(\Gamma) \vdash_{T'} i(\Delta)$.*

It is easy to prove that for every theory interpretation $i : T \to T'$, T is a *subtheory* of $i^{-1}[T']$, i.e., $\vdash_T \subseteq \vdash_{i^{-1}[T']}$.

[16] Rigorously speaking, ontology matching process actually takes two presentations of local theories as input and computes a presentation of the bridge theory as output. However, from a conceptual perspective, in this formal framework we shall characterise ontology matching in terms of the theories themselves. If the language L is infinite, any consequence relations over L will be infinite as well. Therefore, one deals in practice with a finite subset of $\mathcal{P}(L) \times \mathcal{P}(L)$, called a *presentation*, to stand for the smallest consequence relation containing this subset. A presentation may be empty, in which case the smallest consequence relation over a language L containing it is called the *trivial theory*. We will write $Tr(L)$ for the trivial theory over L.

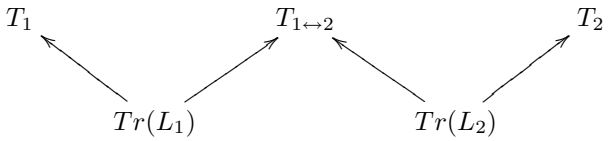
Definition 3. *Given theories* $T_1 = \langle L_{T_1}, \vdash_{T_1} \rangle$ *and* $T_2 = \langle L_{T_2}, \vdash_{T_2} \rangle$*, the* sum $T_1 + T_2$ of theories *is the theory over the sum of language (i.e., the disjoint union of languages)* $L_{T_1} + L_{T_2}$ *such that* $\vdash_{T_1+T_2}$ *is the smallest consequence relation such that* $\vdash_{T_1} \subseteq \vdash_{T_1+T_2}$ *and* $\vdash_{T_1} \subseteq \vdash_{T_1+T_2}$.

Given theory interpretations $i_1 : T_1 \to T$ *and* $i_2 : T_2 \to T$*, the* sum $i_1 + i_2 : T_1 + T_2 \to T$ of theory interpretations *is just the sum of their underlying map of languages.*

Definition 4. *Let* $\mathcal{I} = \{i_{1,2} : T_{1,2} \to T\}$ *be a semantic integration of* T_1 *and* T_2 *with respect to* T*. The integration theory* $T_{\mathcal{I}}$ *of the semantic integration* $\mathcal{I}$ *is the inverse image of* T *under the sum of interpretations* $i_1 + i_2$*, i.e.,* $T_{\mathcal{I}} = (i_1 + i_2)^{-1}[T]$.

The integration theory faithfully captures the semantic relationships existing between sentences in L_{T_1} and L_{T_2} as determined by their respective interpretation into T, but expressed as a theory over the combined language $L_{T_1} + L_{T_2}$. The sum of local theories $T_1 + T_2$ is therefore always a subtheory of the integration theory $T_{\mathcal{I}}$, because it is through the interpretations in T that we get the semantic relationship between languages. It captures the important idea that an integration is more than the sum of its parts.

In ontology matching one usually isolates as output to the matching process the bit that makes $T_{\mathcal{I}}$ genuinely a supertheory of $T_1 + T_2$. The idea is to characterise a theory $T_{1 \leftrightarrow 2}$ over the disjoint union of subsets $L_1 \subseteq L_{T_1}$ and $L_2 \subseteq L_{T_2}$, called a *bridge theory*, which, together with T_1 and T_2, uniquely determines the integration theory $T_{\mathcal{I}}$. To keep everything characterised uniformly in the same conceptual framework, the bridge theory, together with its relationship to the local theories T_1 and T_2, can be expressed by a diagram of theory interpretations as follows:



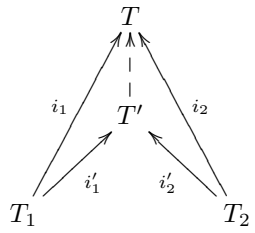
All arrows in this diagram are inclusions of theories. Following [46] we call a diagram of the sort above a *semantic alignment* of local theories T_1 and T_2 through bridge theory $T_{1 \leftrightarrow 2}$. It will be a semantic alignment underlying a semantic integration $\mathcal{I}$ of local theories if the integration theory $T_{\mathcal{I}}$ is a colimit (in a category-theoretical sense [57]) of the above diagram in the category of theories and theory interpretations.

Dynamic ontology matching. What conceptually distinguishes particular instances of ontology matching is the integration $\mathcal{I}$, which is basically (a) the virtual reference theory T with respect to which semantic integration occurs, and (b) the theory interpretations i_1 and i_2 capturing the way local vocabulary is interpreted in this reference theory. In the remainder of this section we briefly discuss these conceptual differences in the context of dynamicity. A thorough theoretical treatment of these differences, however, is outside scope of this deliverable.

First, though, it is important to remark that, for dymanic ontology matching, the reference theory T is usually *not* a fixed, explicit input of the ontology matching process

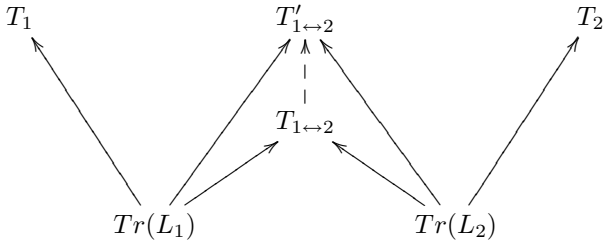
and not even a presentation of it. Instead it should be understood as the mathematical object that stands for the background knowledge determined by the concrete matching parameters and external resources used by an ontology matcher during a particular matching process in order to infer semantic relationships between the underlying vocabularies of the respective input theories. Hence, T is relative to a particular occurrence of matching. For a manual matcher, for instance, the reference theory may be entirely dependent on the particular user input during the matching process, while a fully automatic matcher would need to rely on automatic services (either internal or external to the matcher) to infer such reference theory. It is for this reason that we talk of a *virtual reference theory*, since it is neither explicitly provided to the ontology matcher, nor is it a fixed, unchanging object. It is implicit in the way parameters and external or internal sources are brought into a particular matching process as background theory for ontology matching.

Approximation and partiality. Relativity of ontology matching with respect to the virtual reference theory underlying each particular matching process brings up the issue of relationship between separate occurrences of matching.

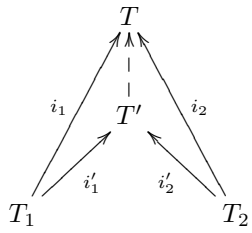


If the above diagram commutes, i.e., if interpreting each local theory T_i into T' and consequently into T is equivalent to interpreting it directly into T, we say that the semantic integration with respect to T' is a refinement of the semantic integration with respect to T. A direct consequence of this is that we shall be able to give a precise characterisation of when the resulting semantic alignment is 'more accurate' or when we will have 'less information loss' when moving from one theory to the other: the integration theory of the refined semantic integration will include the theory of the less refined integration.

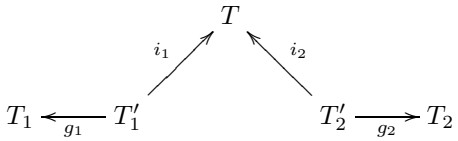
Continuous ontology matching. Reuse of previously established mappings brings up the issue of incrementally constructing semantic alignments from previous ones. Conceptually, this amounts to finding the refinement $T_{1\leftrightarrow 2} \rightarrow T'_{1\leftrightarrow 2}$ between bridge theories, which translates to a refined semantic alignment



from a refined semantic integration:



Interactive ontology coordination and negotiation. We shall model meaning coordination with a coordinated semantic integration, a semantic integration that captures how T_1 and T_2 are progressively coordinated and which captures the integration achieved through an interaction between two peers [76]. The coordinated semantic integration is the mathematical model of this coordination that captures the degree of participation of a peer at any stage of the coordination process. This degree of participation can be captured in a straightforward way with a theory interpretation $g_i : T'_i \rightarrow T_i$. The coordination is then established not between the original theories T_i but between the subtheories T'_i that result from the interaction carried out so far:



4.2 Plausible dynamic ontology matching directions

Below, we discuss five general matching directions that we have identified previously as plausible ones.

Approximate and partial matching. Let us exemplify approaches to approximate and partial matching in turn.

Approximate querying. Many existing automated or semi-automated systems treat ontology matching as a black-and-white question, i.e., two entities are either equivalent or disjoint. When matching is invoked to facilitate, for example, query answering among peers, "gray" areas also become significant. Confidence values are used in some systems [4] mainly indicating to what extent a human user trusts the results provided by the system. Such a trust measurement helps to introduce fuzziness when exact mutual understanding cannot be established among peers. For instance, using LCC notation, $similar\_term(B, n) \Leftarrow a(aligner, P)$ suggests that an imperfect correspondence together with a numeric value (n) is passed from a peer (P) acting as *aligner*. Peers receiving this message can compile the similarity value, the trust value of *aligner* and their local behaviour protocol, and make further decisions accordingly.

Similarity values, whose semantics are not clearly defined, are, however, highly subjective. It neglects the ambiguity of human expressions in conceptualising and conveying meanings and the fact that the accuracy of matching results can only be justified within the application context of the matching. Approximate querying, which has been studied in database and information systems literature [3, 65], might be an alternative. Approximate querying allows us to obtain the semantic relaxation during interactions of peers. Suppose a peer submits a query $ask(\mathsf{C}_i)$ with concepts (terms) from its local vocabulary. Let us also suppose that $ask(\mathsf{C}_i)$ returns an empty set as the answer. This means either that there is no correspondence of C_i or that all the potential correspondences are untrustworthy. In such a case, a query (and the underlying correspondences, in turn) can be relaxed to include neighborhood answers to the original query. Generally, answers to a query can be expanded by i) rewriting predicates as more general ones and ii) rewriting a constant (concept or property) with more general ones. The first relaxation replaces predicates with equivalent but less restrictive ones. For instance, if a peer receives a query $flight(A, B)$ and fails to find any local correspondences, it relaxes the message, breaks it down into sub-queries and accepts interim stops to increase the chance of finding answers:

$$flight\_dest(A, From),\ flight\_dest(B, To),\ travel(From, \_, To)$$

The second relaxation is performed by replacing concepts with their approximations, e.g., the least upper bounds and the greatest lower bounds of C [85] drawn from the hierarchies that the concepts are related to. This is done under the assumption that by querying with a more general concept, we can retrieve a larger set of answers, including the ones that we are looking for. For instance, instead of asking for a specific digital camera, e.g., $asks(cannon\_S500)$, a more general concept $asks(cannon\_S\_series)$ can be used to relax a query. Although it might be the case that *enquirees* might not have correspondences of $asks(cannon\_S500)$ in his/her local vocabulary, we stand more chance of obtaining a nonempty set of instances when using $asks(cannon\_S\_series)$ instead. When multiple concepts (terms) are present in the query, the upper and lower bounds are computed as the conjunction $\bigwedge lub(\mathsf{C}_i)$ and disjunction $\bigvee glb(\mathsf{C}_i)$ of respective upper and lower bounds of each individual concepts.

Partial Matching. When inspected from P2P perspective, it is more likely that mappings are only sought between fragments of ontologies due to the prohibitive cost of identifying a complete set of alignments among entire ontologies, especially for enormous ontologies such as Gene Ontology and Anatomist Foundational Model (FMA) [74]. Such a situation gives raise, for example, to techniques for segmenting (partitioning of) ontologies. Ontology segmentation reduces the complexity of the ontology matching problem by focusing only on most relevant concepts.

Given an arbitrary concept C in ontology O, fragmentation methods, e.g., [78], single out the contexts that C is niched into. Fragmenting ontologies helps to reduce the overhead that is required for processing a particular query. For instance, an LCC script

$$\begin{array}{l} a(buyer, B) :: \\ \quad ask(canonS500) \ \Rightarrow \ a(shopkeeper,\ S) \leftarrow \ldots \end{array}$$

might trigger a fragmentation around concepts camera, shopkeeper and buyer. The resultant set of concepts are given higher priority than others when ontology matching is to be performed, while matching with regard to other concepts then falls into the category of lower priority.

The reduced cost benefits from partial matching and approximate matching that comes with a price: inaccuracy. It is, however, our contention that in P2P environment, the inaccuracy of ontology matching cannot be entirely prevented for the following reasons. In a real-time environment, time and resource constraints simply mean that establishing complete and accurate matching is impractical, if not impossible. Meanwhile, peers exist in contexts and thus the perfectness/imperfectness of matches are context-specific. Implanting defined mappings from their original context into another introduce approximateness. Moreover, the freedom and dynamic nature of peers imply that mappings established *a priori* might often be broken. Residual peers might have to live with incomplete mappings before such links are recovered.

Interactive ontology matching. When ontology matching is done during run-time, full information about the ontologies to be matched is not usually available; instead, each peer attempts to interpret specific statements from other peers with which it is interacting by matching those statements to parts of its own ontology. This means that the context of statements is not revealed and so there may not be enough information to discover correct mappings. However, since this matching is occurring specifically during the interactions of the peers, it is possible for the peers to use the interaction to discuss the matching process. This allows the peers to extract information about the statement to be mapped that it may not be possible to infer. Additionally, this allows the peers to negotiate about what terms will be used in the discussion which will be comprehensible to both of them.

For example, consider a buying-selling interaction in which peers pass messages concerning what they wish to purchase, what the cost of that is, and so on. Let us suppose that the buying peer requests an item (say, a book) and the selling peer returns the following message: $cost(book, 10, euros, visa\_card)$. The buying peer would likely have been expecting a message about cost, but may have been expecting it to be slightly different: in this situation, the payment method is relevant - perhaps the cost is greater if a different card is used - but perhaps the buying peer's ontology does not consider this aspect and represents cost as a ternary function: $cost(Item, Amount, Currency)$. If the buying peer does not recognise the term $visa\_card$, it may not know how to interpret this extra argument. However, because it is currently interacting with the peer that is using this representation, they can temporarily leave the buying-selling protocol and enter an ontology negotiation protocol. The buying peer may ask what type of object $visa\_card$ is and discover from the selling agent that it is of type $credit\_card$. The buying agent is then able to expand its concept classification to include this new object, and also alter its understanding of the relation $cost$ to be one that may take an additional argument of type $credit\_card$. Or perhaps it already understood the relation $cost$ to take four arguments, the last of which was $cash$: this may have come about through a similar interaction in which a selling peer wanted cash. Encountering this argument instantiated as $credit\_card$ indicates that the peer's original notion of the correct type for the fourth

argument was too specific and it should be extracted to be of type $payment\_method$. Additionally, now that it can tie this unexpected object into its own ontology, it can negotiate about other possibilities. It cannot fulfil the request to pay by $visa\_card$ since, as it has only just learnt what a visa card is, it clearly does not have one. However, it will now be able to deduce that $access\_card$ is a sibling concept, and can attempt to negotiate with the selling peer about paying with that.

Continuous "design-time" matching. Traditionally, ontology matching is performed *off-line*, mainly at design-time by dedicated ontology engineers and/or domain experts. It has long been thought that good initial ontology matching at design-time will eliminate, or at least alleviate, follow-up matching efforts. Many of us have grown to believe that matching tasks performed at design-time would have knock-on effect on ontology related interoperability issues, and if such issues persist it is probably because of a wrongly selected matching method or inherent problems with the ontologies themselves. This static perspective of most current matching systems will struggle in a dynamic P2P environment. We opt for the *agile development* method in software engineering and suggest the continuity of design-time matching to the entire life-cycle of a peer.

Continuous "design-time" matching is based on the assumption that in P2P environment, interoperability becomes a pertinent issue only when requests are submitted and processed. Once a query has been tackled and the communication channel has been established, data translation becomes the dominant issue till new requests are raised. It is, therefore, reasonable to relocate the interim periods between two consecutive requests to refine and update existing mappings.

The current matching capability normally captures ontologies with a set of features, e.g., names, relations with other entities. Based on the extent to which they take on features from adjacent entities as evidence, matching algorithms can be classified and arranged along an axis of increasing computational complexity [15, 16]. A possible continuous matching scenario might, therefore, be as follows. In the first step, a set of features are selected from a peer's local vocabulary and an initial consensus is defined. We call this the *current consensus*. In subsequent steps, as follow-up interactions broach more features of the ontologies from both the query submitting peer and the query handling peers, candidates in the *current consensus* are re-examined so that non-qualified ones are discarded. In this way, the *current consensus* becomes smaller and more accurate at each recursive step. By keeping a good explanation and backtracking mechanism, it is possible to recover discarded candidates when conflicts are encountered. Let us consider an example. Suppose a peer P is seeking to download annotated mammographic images. When processing a request $asks(mammogram)$, the initial *current consensus* can be established by applying algorithms from the cheapest category, e.g., string-based methods. It might fill the *current consensus* with terms such as $mammogen$, $mammoplasty$, $mammotropin$, and so on, which are obviously unsatisfactory answers to a query regarding $mammogram$. Further interactions gathered from $tells(X$ **is-a** $Xray\_image)$ might reveal the fact that Mammogram is a subconcept of Xray_image based on which $mammogen$, $mammoplasty$ and $mammotropin$

are pruned and the *current consensus* is refined. Such a process continues till a fix-point is reached or a satisfactory answer is accept by the enquirer.

Note that there are three prerequisites of such a continuous approach. Firstly, it should be possible to extract and categorise a set of features, the matching task of which presents a spectrum of increasing accuracy and/or computational complexity. Secondly, a fix point that terminates refinement procedure can be defined. The simplest form of such a fix point can be a "hard" deadline when the matching process has to stop. Finally, candidates in the current consensus are represented in languages understandable and executable by software agents, who are then able to restore the previous execution and combine it with newly discovered knowledge.

Community-driven ontology matching. [17] Below, we discuss one of the community-driven ontology matching approaches, called *public alignment reuse* [91]. A rationale behind the alignment reuse is that many ontologies to be matched are similar to already matched ontologies, especially if they are describing the same application domain [70, 81]. Eventually, once an alignment has been determined, it can be saved and further reused as any other data on the web. Thus, a (large) repository of mappings has the potential to increase the effectiveness of matching systems by providing yet another source of domain specific knowledge. Unlike, e.g., COMA++ [5], which followed a *private* alignment reuse approach (where access to the system is limited to individual users who usually do not know each other, and hence they do not communicate with each other); in a *public* approach, any agent can match ontologies, save the alignments so that these are available for any other agents to reuse, thus, enabling the cross-fertilization between the participating parties and helping to achieve the goals of these parties cooperatively.

Let us consider a simple scenario which involves four researchers from two natural science communities. The researchers are `Mark`, `Michael`, `Jenny`, and `Alexander`. They are represented by roles held in their communities (i.e., end user, knowledge engineer, developer) and web domains/communities where they interact (e.g., biology, chemistry). These researchers have the following profiles:

name	`Mark`
interacts	biology, chemistry web applications
role(s)	end user

name	`Jenny`
interacts	chemistry web application
role(s)	end user, developer

name	`Michael`
interacts	biology, chemistry web applications
role(s)	end user, knowledge engineer

name	`Alexander`
interacts	biology, chemistry web applications
role(s)	end user

Suppose the following two actions take place:

- `Michael` creates an alignment `m` between ontologies coming from `biology` and `chemistry` web applications;
- `Alexander` uses the alignment `m`.

[17] The example under consideration has been taken from [91].

The result of a public alignment reuse is the alignment `m`, which is recommended to `Mark`. After a tool recommends a new mapping to `Mark`, he, as a researcher, can benefit from the extended interoperability between `biology` and `chemistry` web applications without applying any effort to rediscover the new knowledge (already established by `Michael` and validated by `Alexander`). In the proposed scenario, alignment `m` is not recommended to `Jenny`, because she does not use the `biology` web application.

A specific feature of correspondences resulting from this kind of ontology matching is their customization to the user/community and application requirements (*subjective alignments*). All these relations are represented via *annotated mappings* and are to be propagated to the communities associated with the human contributor. Notice that subjective alignments are appropriate for specific tasks in a specific community, but may be inappropriate or even contradicting to the practices of other communities.

Multi-ontology matching. In some applications it is often the case that many agents are involved in it and a consensus is growing from a pool of information. For instance, in e-commerce scenarios, surrogates representing the customers might need to negotiate with more than one service and product provider as well as among the surrogates themselves. In such communications, ternary and quaternary dialogues (and, in turn, correspondences, as opposed to binary ones) abound while each participant might hold his/her own view of the domain of discourse, see, e.g., [17, 45]. To put in line the above example, let us imagine an on-line natural science forum, in which `Jenny` is a registered developer and has her ontology aligned with the rest of the group. In a particular point, `Michael`, `Mark` and `Alexander` log in as new users, each of whom brings a new domain ontology. In such cases, a ternary consensus would allow `Michael`, `Mark` and `Alexander` to "kick-off" discussions immediately on a subtopic of biology that they are all interested in.

One of the obstacles in multi-ontology matching is that ontology matching techniques may often be computationally expensive, and therefore, cannot scale to a large number of ontologies. To cope with this issue we envisage a distributed approach. For example, distributed hash tables (DHTs) of [71, 84] provide implementations of self-organizing DHTs on a P2P network. They produce a hash value for the keys of objects that are to be stored, and place them in the peer with ID closest to that hash-value. Thus, a peer is *responsible* for key K when K's hash value lies closest to the ID of that peer.

We use a DHT as a naming system for a method in which peers decide which data is relevant to them, muster more relevant data and repeat. Although this approach can be applied to a variety of techniques and types of ontologies, we will describe a simple case: using folksonomies (i.e., sets of related terms)[18] together with the method provided by Schutze in [77] for automatic word sense discrimination. Each folksonomy is registered in the peers responsible for each of its terms. Therefore, for example, the peer responsible for XML will contain all folksonomies with the term XML. This provides us with an initial clustering of related folksonomies into peers. Obviously, this is the simplest clustering possible. After this, we may use a more computationally complex clustering algorithm to extract the most important terms. Finally, clusters of terms,

[18] See, for example, `http://del.icio.us/`.

for example which are synonyms, result in equivalence correspondences between all the terms from such a cluster.

In [77], terms and their senses are represented in a real-valued high-dimensional space where closeness in the space corresponds to semantic similarity. As input for this method one may use the folksonomies for each term. From the representations of terms in the high-dimensional space it is possible to determine the similarity between terms. This can be done by evaluating when senses are close in the high-dimension space of Schutze, in which case it is likely that they have similar meanings.

Synthesis. Techniques presented in this section are by no means mutually exclusive. In fact, a closer look at the above discussions might reveal that some techniques might be the prerequisite or outcomes of and, in many cases, the trigger for the applications of others. For instance, *continuous "design-time" matching* might rely on initial approximate or partial matching and aims to provide sufficiently "good" *approximate*/*partial* matching results at the end of each revision; *interactive matching* might underpin the *continuous "design-time" matching* by gradually unveiling the domain of discourse through communications; a public alignment towards a set of common ontologies would certainly allows multi-ontology matching to emerge.

A possible combinatory scenario might be as follows. Let us suppose that a query $buy(canonS500)$ is submitted to a group of peers, G, *public alignment reuse* method facilitates the retrieval of known mappings from previous interactions. The peers in G then evaluate each known mapping within the context of the current interaction and gather possible mappings that might answer the query into the initial *partial matching* consensus M_c. If previous mappings are not trustworthy, G populates M_c by mappings that are obtained with relatively "cheap" methods, e.g., *partial matching* and/or "*shallow matching*" [16] or approximate matching, e.g., classifying $canonS500$ as $camera$, $photocopier$, $printer$, that are products of Canon Inc. While sending the responses drawn from M_c back to and awaiting the feedback from the *enquirer*, peers in G *continuously* work on M_c by applying more expensive algorithms or consulting external peers.

Having received messages from G, the *enquirer* might be in one of the following states: (i) it might consider that the answers from G are satisfactory and terminate the interactions with G; (ii) it might relax the query to include *approximate* answers; (iii) it might refine the query with the new knowledge discovered in the answers from G. Further *interactions* between G and the *enquirer* would help to shape a more accurate M_c that allows the latter to obtain satisfactory answers. For instance, new evidence might suggest a refinement according to the known model code "$S500$" or the fact that the *enquirer* is asking information about a camera to pin down the type of the referred product.

5 Systems and evaluation

5.1 State of the art prototypes

At present there exists around fifty matching prototypes, see, e.g., [9, 20, 22, 31, 38, 51, 58, 60, 68]. Most of them have been already discussed in the previous surveys addressing the ontology matching topic, see, e.g., [8, 21, 49, 53, 67, 70, 81, 88]. Thus, in this section we revisit only those (several) systems (e.g., QOM [25]) which are of particular interest for us from the dynamic ontology matching perspective. It is worth also noting that ontology matching is an area which evolves quite fast itself. Therefore, we have considered here a number of recent prototypes which try to deal with dynamic applications.

DCM framework (U. of Illinois at Urbana-Champaign). MetaQuerier [17] is a middleware system that assists users to find and query multiple databases on the web. It exploits the DCM (dual correlation mining) matching framework to facilitate, e.g., source selection according to a user's search keywords [45]. The system takes as input multiple schemas and returns mappings between all of them. DCM automatically discovers complex mappings (e.g., {author} corresponds to {first name, last name}) between attributes of the web query interfaces in the same domain of interest (e.g., books). As the name DCM indicates, schema matching is viewed as *correlation mining*. The idea is that co-occurrences of patterns often suggest the complex matches. That is, *grouping attributes*, such as first name and last name, tend to co-occur in query interfaces. Technically, this means that those attributes are positively correlated. Contrary, attribute names which are synonyms, e.g., quantity and amount, rarely co-occur, thus representing an example of negative correlation between them. Matching is performed in two phases. During the first phase (matching discovery), a set of matching candidates is generated by mining first positive and then negative correlations among attributes and attribute groups. Also, some thresholds and a specific correlation measure (H-measure) are used. During the second phase (matching construction), by applying some strategies of ranking (e.g., scoring function, rules) and selection (iterative greedy selection), the final alignment is produced.

FOAM: QOM (U. Karlsruhe). FOAM is a Framework for Ontology Alignment and Mapping. It includes such matching systems as NOM, QOM, and APFEL [25–27]. In the context of this deliverable we are interested primarily in the QOM system. However, we briefly revisit also NOM in order to facilitate understanding of QOM.

NOM (Naive Ontology Mapping) [27] combines a set of elementary matchers based on rules, exploiting explicitly codified knowledge in ontologies, such as information about super- and sub-concepts, super- and sub-properties, etc. At present the system supports 17 rules. For example, rule (R5) states that if super-concepts are the same, the actual concepts are similar to each other.

QOM (Quick Ontology Mapping) [25] is a successor of the NOM system. The approach is based on the idea that the loss of quality in matching algorithms is marginal (to a standard baseline); however improvement in efficiency can be tremendous. This fact

allows QOM to produce mapping elements fast, even for large-size ontologies. QOM is grounded on the matching rules of NOM. However, for the purpose of efficiency the use of some rules has been restricted, e.g., R5. Also, QOM avoids the complete pair-wise comparison of trees in favor of a (n incomplete) top-down strategy, thereby focusing only on promising matching candidates. The similarity measures produced by matching rules are aggregated by using a sigmoid function, thereby emphasizing high individual similarities and de-emphasizes low individual similarities. Finally, with the help of thresholds, the final alignment is produced.

OntoBuilder (Technion Israel Institute of Technology). OntoBuilder is a system for information seeking on the web [64]. A typical situation the system deals with is, for example, when a user is searching for a car to be rented. Obviously, the user would like to compare prices from multiple providers in order to make an informed decision. Thus, the same input information has to be typed in many times. OntoBuilder operates in two phases, namely: (i) ontology creation (the so called *training* phase) and (ii) ontology adaptation (the so called *adaptation* phase). During the training phase an initial ontology (in which a user's data needs are encoded) is created by extracting it from a visited web-site of, e.g., *AVIS* car rental company. The adaptation phase includes on-the-fly matching and interactive merging operations of the related ontologies with the actual (initial) ontology. Ontology creation is out of the scope of this deliverable. Hence, we concentrate only on the ontology adaptation phase. During the adaptation phase the user suggests the web sites (s)he would like to further explore, e.g., the *Hertz* car rental company. Each such a site goes through the ontology extraction process, thus, resulting in a candidate ontology, which is then merged into the actual ontology. To support this, the best match for each existing term in the actual ontology (to terms from the candidate ontology) is selected. Selection strategy employs thresholds. The matching algorithm works in a term to term fashion. It sequentially executes a number of matchers. Some examples of the matchers used here are removing noisy characters and stop terms and substring matching. If all else fails, a thesaurus look-up is performed. Finally, mismatched terms are presented to the user for manual matching. Some further matchers such as those for precedence matching were introduced in a later work in [34]. Also Top-K mappings as alternative for single best matching (i.e., top-1 category) was proposed in [33].

ORS (University of Edinburgh). The Ontology Refinement System (ORS) [59] deals with the problem of syntactic mismatch between similar first-order ontologies in agent settings. Its ultimate goal is slightly different compared with the other systems discussed in this section, namely it is concerned with the repair of ontologies. The system has a built-in grammar of potential mismatches - for example, for first order relations these may be mismatches in the predicate name or in the number, ordering or types of arguments - and diagnoses mismatches using algorithms that pinpoint the potential mismatch with reference to this grammar. For example, a mismatch in the predicate name exist between $price(Item, Amount)$ and $cost(Item, Amount)$. However, $price(Item, Amount)$ would also be mismatched with $price(Item, Amount, PaymentMethod)$, $price(Amount, Item)$ and $price(Book, Amount)$ in terms, respec-

tively, of number, order of type of the arguments. Unlike many ontology matching systems, ORS does not require full access to two (or more) external ontologies between which it acts as a third-party interpreter. Instead it is designed as a matching process that can be attached to an agent that is, in the course of its other activities, attempting to interpret the locutions of other agents. Thus ORS requires access to one complete ontology (the ontology of its agent, to which these locutions must be mapped) and only the small parts of other ontologies that are revealed in locutions. Matching entire ontologies is not the goal; only those parts that are central to the comprehension of the interaction that is currently taking place are relevant. Since, in a large-scale, multi-user system, it is not possible to predict which agents will be interacted with and what locutions these interactions contain, it is neither practical nor possible to do this matching off-line; instead these locutions must be matched during the interactions as the need for them is revealed. ORS particularly focuses on mismatches caused by abstractions, where ontological objects have been made more general, or refinements, where ontological objects have been made more specific. This is because ontologies are often changed since they have been found to contain unnecessary detail or to be lacking in necessary detail. Examples of such changes are adding or removing arguments from relations so that they contain more or less information; changing the types of the arguments to more or less general types (sub- or super-types); changing the relation itself into a more or less general relation.

PowerMap (Open University). PowerMap is a matching algorithm that has been developed in the context of the PowerAqua question answering system [55]. As such, it was designed to be integrated in an open environment, such as the semantic web, where concepts extracted from a user query are matched to on-line ontologies at run-time and the selected ontologies are used for answering the question. PowerMap is a hybrid matching algorithm comprising linguistic and structural matching techniques with the assistance of a thesaurus, e.g., WordNet. The algorithm takes as input a triple or a set of triples and returns semantically equivalent ontology entities (classes, instances, properties) for each of the terms of the triples. PowerMap has two major phases.

The first phase identifies a set of candidate mappings for the query terms: given the terms of the triples, this step returns mappings to entities in on-line ontologies that are lexically related to these terms. This *terminological* phase concentrates on element-level matching and identifies candidate classes and instances from different ontologies to be matched to each query term. This step relies on lemmatization, acronyms (for instances), string-based comparison (e.g., edit distance) and thesauri look-up for synonyms, hypernyms and hyponyms.

The second phase takes as input the candidate mappings identified in the previous step and filters out those that are indeed relevant. For example, if the term capital is matched to concepts with identical labels in a geographical and a financial ontology, the system needs to decide which mapping is relevant for the sense of the query term. This step uses *structural* matching over the class hierarchy. It computes semantic similarity between the user query terms and candidate ontology class terms, and between the candidate ontology classes themselves. Similarity between classes is measured once the meaning of the ontology class (its sense) is made explicit by an interpretation not

only of its label but also of its WordNet sense determined by its position in the ontology taxonomy (in particular, the sense of an ontology class is determined by the sense of its ascendant/descendant in the ontology). Similarity between classes is measured by the distance (depth) and common subsumers between the two concept/senses in the WordNet "IS-A" hierarchy (as in hierarchy distance based matchers).

This method avoids a global interpretation of the matched ontologies and aims to provide partial mappings. Not all the concepts of the ontology are interpreted, only those that seem similar to the mapping terms (i.e., those identified by the first step of the algorithm). The obtained mappings are stored in a semantic data repository (Sesame) in order to enable their future reuse.

S-Match (University of Trento). S-Match [36–38] is a schema-based matching system. It takes two graph-like structures (e.g., XML schemas, classifications) and returns semantic relations (e.g., equivalence, subsumption) between the nodes of the graphs that correspond semantically to each other. The relations are determined by analyzing the *meaning* (concepts, not labels) which is codified in the elements and the structures of schemas/ontologies. In particular, labels at nodes, written in natural language, are translated into propositional formulas which explicitly codify the label's intended meaning. This allows for a translation of the matching problem into a propositional unsatisfiability problem, which can then be efficiently resolved using (sound and complete) state of the art propositional satisfiability deciders. S-Match was designed and developed as a platform for semantic matching, namely a highly modular system with the core of computing semantic relations where single components can be plugged, unplugged or suitably customized. At present, S-Match libraries contain 13 element-level matchers, see [40], and 3 structure-level matchers (e.g., SAT4J [10]).

S-Match has been used so far mostly in traditional applications. Based on the recent optimizations and evaluations, see [41], S-Match has significantly improved its performance characteristics. Thus, it demonstrates a good potential also for the emerging applications.

5.2 Evaluation methodology

Evaluation problem. In general, we think of evaluation as an assessment of performance or value of a system, process, product, or policy. As [75] indicates, an evaluation activity requires:

- A *system* or its representation such as prototype, product, etc.; together with its underlying *process* (algorithm, simulation, etc.);
- *Criteria* representing the objectives of the systems;
- *Measures* based on the criteria;
- *Measuring instruments* to register (or compute) the measures;
- *Methodology* for obtaining the measurements and conducting the evaluation (i.e., set of tools and methods applied in order to obtain the experimental results).

Let us review these general requirements one by one from the matching perspective. The *system* and *process* in this case include the test cases and their associated processing under given algorithms and procedures.

The major criterion exploited in the matching evaluation is *relevance*, i.e., how relevant are the mappings produced by a matching system with respect to the end user. Let us consider measures of relevance that are well known in information retrieval, such as *Precision*, *Recall*, and *F-measure*, which were adapted to the matching domain, e.g., [19]. Calculation of these measures is based on the comparison of the mappings produced by a matching system (R) with the reference mappings considered to be correct (C).

- *Precision* varies in the [0,1] range; the higher the value, the smaller the set of wrong mappings (false positives) which have been computed. Precision is a correctness measure. It is computed as follows:

$$Precision = \frac{|C \cap R|}{|R|} \tag{1}$$

- *Recall* varies in the [0,1] range; the higher the value, the smaller the set of correct mappings (true positives) which have not been found. Recall is a completeness measure. It is computed as follows:

$$Recall = \frac{|C \cap R|}{|C|} \tag{2}$$

- *F-measure* varies in the [0,1] range. It is global measure of the matching quality, which increases if the matching quality increases. The version presented here is computed as a harmonic mean of Precision and Recall:

$$F - measure = \frac{2 * Recall * Precision}{Recall + Precision} \tag{3}$$

The other important criterion is *efficiency*, i.e., how fast the mappings are produced. Usually the *execution time* is taken as a measure of efficiency.

In computation of the relevance based measures the *experts* play the role of measuring instruments, since they define the relevance of the mappings. For execution time the *clock* is considered as a measuring instrument.

Finally, the testing methodology includes design, manner and techniques used to obtain and analyze the evaluation result. These, in turn, also need to be evaluated for their validity and reliability.

Evaluation efforts. To the best of our knowledge (coordinated international) ontology matching evaluation efforts have emerged only recently [19], see [2, 4, 86]. These are:

I3CON [20]. In I3CON of 2004, five ontology matching systems have been evaluated on eight test cases taken from various domains. The ontologies were taken from the web and matched against their modifications (e.g., adaptation to the concerned topic, language translation). The biggest matching problem was constructed from

[19] `http://oaei.ontologymatching.org/`

[20] `http://www.atl.external.lmco.com/projects/ontology/i3con.html`

the ontologies with hundreds of classes. The reference mappings were produced by consensus of an external group of students. Precision, Recall and F-measure have been calculated for all the test cases and compared among the systems. The best systems demonstrated 0.7-0.8 Precision and 0.7-0.8 Recall, which produced a 0.6-0.75 F-Measure. Efficiency of the systems was not considered in the evaluation.

EON [21]. In the EON contest of 2004, four ontology matching systems have been evaluated on a test bed consisting of 20 matching problems. In particular, the initial ontology taken from the bibliography domain was matched against its 16 modifications obtained in a (semi) automatic way (e.g., flattened hierarchy, no instances) and 4 ontologies of the same domain developed by the different institutions. These ontologies contained tens of classes. The reference mappings were known by participants in advance, which allowed them to tune their systems. The tools for mappings management and evaluation of the matching quality measures were provided [29]. The qualitative measures (such as Precision and Recall) were calculated for all the matching problems and the analysis of results were performed by the authors of the matching systems. The best Precision results were in 0.9-0.99 range while Recall slightly exceeded 0.8. Efficiency of the systems was not considered in the evaluation.

OAEI-2005 [22]. Ontology Alignment Evaluation Initiative (OAEI) of 2005 aimed at performing comparative evaluation of the matching quality of seven ontology matching systems. As from [4], these are: OMAP, CMS, Dublin20, Falcon, FOAM, OLA, and ctxMatch2. The evaluation consisted of two parts:

- A systematic test set composed from ontologies of bibliographic domain (tens of nodes). This is a slightly modified version of the EON-2004 test set.
- Two large scale real world matching tasks, namely: (i) the first matching task was extracted from Google, Looksmart,Yahoo web directories and (ii) the second task was matching the FMA [74] and GALEN [72] medical ontologies (tens of thousands of nodes). For the task of matching medical ontologies no reference mappings were known in advance, and therefore, no matching quality measures can be estimated. For the web directories an incomplete reference mappings set was provided. Notice that this incomplete reference mappings set allows the evaluation of only Recall of matching solutions.

Nearly all the systems participating in the evaluation demonstrated high or very high results in systematic tests. For example, five systems out of seven have shown Precision higher than 0.8 and Recall higher than 0.7. At the same time web directory matching task turned out to be much harder. In fact, the best Recall value was about 0.3 which is significantly lower that previously reported results. Medical ontologies matching task was exploited as a scalability test for the matching systems and only a few of them (namely, CMS and Falcon) were capable of providing the results for this kind of matching task. Finally, it is worth noting that efficiency of the systems was not considered in the evaluation.

Let us now discuss two evaluation efforts performed individually by matching systems developers. In [25] (the QOM system), three ontology matching systems were

[21] http://oaei.ontologymatching.org/2004/Contest/

[22] http://oaei.ontologymatching.org/2005/

evaluated on 3 pairs of ontologies composed from hundreds of entities. The matching quality and efficiency indicators were computed. A trade off between efficiency and quality have been demonstrated for two of the systems. The highest F-measure (for the tests under consideration) of 0.8 has been reported. The most efficient matching solution solved the task in 10 seconds and demonstrated F-Measure of 0.6.

In [41] (the S-Match system), the efficiency of four matching solutions was evaluated on 2 artificially produced and 4 real world matching problems. The size of the matching tasks varied from tens up to thousands of nodes. The evaluation results showed that the fastest matching system were able to solve the matching tasks with tens of nodes in less than a second. The matching tasks with hundreds of nodes were solved in 10-200 seconds. The biggest matching tasks involving thousands of nodes were solved in 10 minutes.

Evaluation summary. At present, nearly all state of the art matching systems suffer from the lack of evaluation. Till very recently there was no comparative evaluation and it was quite difficult to find two systems evaluated on the same dataset. Often authors artificially synthesize datasets for empirical evaluation but they rarely explained their premises and assumptions. Also, most of the current evaluation efforts were devoted to ontologies with tens of nodes and only some works (see, e.g., [25]) present the evaluation results for the matching problems with hundreds of nodes. At the same time, industrial sized matching problems may contain up to tens of thousands of nodes. This is the case mostly due to lack of large scale annotated mapping datasets obtained from real world matching tasks.

Given the fact that coordinated international evaluation of matching approaches in general is still in its infancy, unfortunately, dynamic applications as providers of matching tasks are largely ignored for the time being.

6 Open issues and challenges

In this section we discuss some open issues and challenges which have to be addressed in order to realize the dynamic ontology matching.

Lack of background knowledge. Recent industrial-strength evaluations of matching systems, see, e.g., [6, 30, 39], show that lack of background knowledge, most often domain specific knowledge, is one of the key problems of matching systems. In fact, most state of the art systems, for complex real world matching tasks, produce much lower values of *recall* ($\sim$30%) than for *toy* examples, where the recall was most often around 80-90%. This problem has a particular significance in dynamic settings. In fact, the matching input is often much more shallow, being a set of (sometime, random) terms, and therefore, it incorporates fewer clues to result in plausible mappings.

Performance. Following the above mentioned examples from the industrial-strength evaluations, besides the effectiveness of the results, there is an issue of performance. Performance is of prime importance in many dynamic applications, for example, where a user can not wait too long for the system to respond. *Execution time* indicator shows

scalability properties of the matchers and their potential to become industrial-strength systems. Also, referring to the above mentioned evaluations, the fact that some systems ran out of memory on some test cases, although being fast on the other test cases, suggests that their performance time was achieved by using a large amount of main memory. Therefore, usage of *main memory* should also be taken into account.

Interactive (automated) approaches. In traditional applications, as the above mentioned results of evaluations indicate, automatic ontology matching usually cannot deliver a high quality results, especially on huge datasets. Thus, for traditional applications, semi-automatic matching is a plausible way to improve the effectiveness of the results. Dynamic applications in this sense have a specific feature which traditional applications do not. In fact, since there are multiple parties (agents) involved in the process, mismatches (mistakes) could be negotiated (corrected) in a fully automated way. Obviously, during interaction the full context of ontological objects used will not be available. However, this can involve negotiation between the agents as to what these objects are referring to and how they may be matched into each other's ontologies or some shared understanding. Along the lines of interactions, argumentation schemas the agents use are becoming important, since this is the way that agents may build a consensus. Therefore, *explanations* of matching [82], being an argumentation schema, become crucial. In fact, if two agents are going to trust the fact that two terms may have the same meaning, then they need to understand the reasons leading a matching system to produce such a result.

Social aspects. The impact of social networks, web communities and direct involvement of humans (in a distributed fashion) on dynamic ontology matching has to be analyzed and distilled. As the public alignment reuse approach (discussed previously) indicates, once an alignment has been determined, it can be saved, annotated, and further reused as any other data on the web. A first open issue here is what should be succinctly included in the annotation (codifying social aspects) of a mapping element in order to be practically useful in the future. On the one hand, a (large) repository of mappings has the potential to increase the effectiveness of matching systems by providing yet another source of domain specific knowledge [91]. On the other hand, users can publish different and even contradicting alignments. Hence, one of the open problems here is how to manage the contradictory mappings in the repositories.

Self-configuration and customizing technology. In dynamic settings, it is natural that applications are constantly changing their characteristics. Therefore, approaches that attempt to tune and adapt automatically matching solutions to the settings in which an application operates are of high importance. This includes automation of the combination of individual matchers and libraries of matchers, namely which matchers to use and where, what are the most appropriate thresholds, weights, coefficients, and so on.

Evaluation. There are at least two challenges in evaluation of dynamic ontology matching. The first one concerns automated dataset construction methodologies, e.g., how to acquire the reference mappings automatically in dynamic settings. The second challenge concerns new quality measures, which are application specific; that is, how to assess whether the result of matching is good enough for the application.

7 Conclusions

In this deliverable we have extended the notion of ontology matching to dynamic ontology matching. We have considered three scenarios operating in dynamic settings from various applications, namely biomedicine, emergency, open browsing/query answering, and collected the requirements they pose towards a matching solution. Based on these requirements we have identified the main difference between dynamic matching and conventional ontology matching. In particular, this difference lies in a number of functional and non-functional requirements. These functional requirements include: (i) approximate, (ii) partial and (iii) run-time matching. Non-functional requirements include: (i) good enough quality of matching results and (ii) matching performance constraints. We have discussed five general matching directions which we believe can appropriately address those requirements. These are: approximate and partial ontology matching; interactive ontology matching; continuous "design-time" ontology matching; community-driven ontology matching; and multi-ontology matching. We overviewed state of the art matching systems as well as their evaluation principles from the dynamic ontology matching perspective. Finally, the key open issues and challenges towards realizing dynamic ontology matching were discussed, thereby providing a vision for the major future activities.

References

1. JXTA project. see http://www.jxta.org.
2. The information interpretation and integration conference. http://www.atl.external.lmco.com/projects/ontology/i3con.html, 2004.
3. S. Acharya, P. B. Gibbons, and V. Poosala. Aqua: A fast decision support systems using approximate query answers. In *Proceedings of VLDB*, pages 754–757, 1999.
4. B. Ashpole, J. Euzenat, M. Ehrig, and H. Stuckenschmidt, editors. *Integrating Ontologies*. Proceedings of the Integrating Ontologies workshop at K-CAP, 2005.
5. D. Aumüller, H. H. Do, S. Massmann, and E. Rahm. Schema and ontology matching with COMA++. In *Proceedings of SIGMOD, Software Demonstration*, 2005.
6. P. Avesani, F. Giunchiglia, and M. Yatskevich. A large scale taxonomy mapping evaluation. In *Proceedings of ISWC*, pages 67 – 81, 2005.
7. J. Barwise and J. Seligman. *Information Flow: The Logic of Distributed Systems*. Cambridge University Press, 1997.
8. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.
9. S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, (28(1)):54–59, 1999.
10. D. Le Berre. A satisfiability library for Java. http://www.sat4j.org/.
11. P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. Contextualizing ontologies. *Journal of Web Semantics*, (26):1–19, 2004.
12. D. Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. Technical report, World Wide Web Consortium (W3C), http://www.w3.org/TR/rdf-schema/, 2004.
13. D. Brickley and L. Miller. FOAF Vocabulary Specification. Technical report, http://xmlns.com/foaf/0.1/, 2005.

14. D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Logical foundations of peer-to-peer data integration. In *Proceedings of PODS*, pages 241–251, 2004.
15. S. Castano, A. Ferrara, and S. Montanelli. Matching ontologies in open networked systems: Techniques and applications. In *Journal of Data Semantics V*, pages 25–63, 2006.
16. S. Castano, A. Ferrara, S. Montanelli, and G. Racca. Semantic information interoperability in open networked systems. In *Proceedings of ICSNW, in cooperation with ACM SIGMOD*, pages 215–230, 2004.
17. K. C.-C. Chang, B. He, and Z. Zhang. Toward large scale integration: Building a metaquerier over databases on the web. In *Proceedings of CIDR*, 2005.
18. F. Corrêa da Silva and J. Agustí. *Knowledge Coordination*. John Wiley & Sons, 2003.
19. H. H. Do, S. Melnik, and E. Rahm. Comparison of schema matching evaluations. In *Proceedings of the workshop on Web and Databases*, 2002.
20. H. H. Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In *Proceedings of VLDB*, pages 610–621, 2002.
21. A. Doan and A. Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine, Special Issue on Semantic Integration*, 2005.
22. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map ontologies on the semantic web. In *Proceedings of WWW*, pages 662–673, 2003.
23. J. Dunn and M. Hardegree. *Algebraic Methods in Philosophical Logic*. Oxford University Press, 2001.
24. M. Dzbor, J. Domingue, and E. Motta. Magpie - towards a semantic web browser. In *Proceedings of ISWC*, pages 690–705, 2003.
25. M. Ehrig and S. Staab. QOM: Quick ontology mapping. In *Proceedings of ISWC*, pages 683–697, 2004.
26. M. Ehrig, S. Staab, and Y. Sure. Bootstrapping ontology alignment methods with APFEL. In *Proceedings of ISWC*, pages 186–200, 2005.
27. M. Ehrig and Y. Sure. Ontology mapping - an integrated approach. In *Proceedings of ESWS*, pages 76–91, 2004.
28. H. Enderton. *A Mathematical Introduction to Logic*. Academic Press, second edition, 2001.
29. J. Euzenat. An API for ontology alignment. In *Proceedings of ISWC*, pages 698–712, 2004.
30. J. Euzenat, H. Stuckenschmidt, and M. Yatskevich. Introduction to the ontology alignment evaluation 2005. In *Proceedings of Integrating Ontologies workshop at K-CAP*, 2005.
31. J. Euzenat and P. Valtchev. Similarity-based ontology alignment in OWL-lite. In *Proceedings of ECAI*, pages 333–337, 2004.
32. A. Farquhar, R. Fikes, and J. Rice. The Ontolingua Server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6):707–728, 1997.
33. A. Gal. Managing uncertainty in schema matching with top-k schema mappings. *Journal on Data Semantics*, 2006.
34. A. Gal, G. Modica, H. Jamil, and A. Eyal. Automatic ontology matching using application semantics. *AI Magazine*, 26(1):21–32, 2005.
35. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1997.
36. F. Giunchiglia and P. Shvaiko. Semantic matching. *The Knowledge Engineering Review*, 18(3):265–280, 2003.
37. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-match: An algorithm and implementation of semantic matching. In *Proceedings of ESWS*, pages 61–75, 2004.
38. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. Semantic schema matching. In *Proceedings of CoopIS*, pages 347–365, 2005.
39. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. Discovering missing background knowledge in ontology matching. In *Proceedings of ECAI*, 2006.

40. F. Giunchiglia and M. Yatskevich. Element level semantic matching. In *Proceedings of Meaning Coordination and Negotiation workshop at ISWC*, 2004.
41. F. Giunchiglia, M. Yatskevich, and E. Giunchiglia. Efficient semantic matching. In *Proceedings of ESWC*, pages 272–289, 2005.
42. F. Giunchiglia and I. Zaihrayeu. Making peer databases interact - a vision for an architecture supporting data coordination. In *Proceedings of the International workshop on Cooperative Information Agents (CIA)*, pages 18–35, 2002.
43. J. Goguen and R. Burstall. Institutions: abstract model theory for specification and programming. *Journal of the ACM*, 39(1), 1993.
44. A. Halevy, Z. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The piazza peer data management system. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):787–798, 2004.
45. B. He and K. C.-C. Chang. Automatic complex schema matching across web query interfaces: A correlation mining approach. *ACM Transactions on Database Systems*, 31(1):1–45, 2006.
46. P. Hitzler et al. Integrated view and comparison of alignment semantics. Deliverable D2.2.5, KnowledgeWeb EU Network of Excellence, November 2005.
47. I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: a semantic web rule language combining OWL and RuleML. Technical report, http://www.daml.org/rules/proposal/, 2004.
48. Y. Kalfoglou and M. Schorlemmer. IF-Map: An ontology-mapping method based on information-flow theory. *Journal on Data Semantics I*, pages 98–127, 2003.
49. Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review Journal (KER)*, (18(1)):1–31, 2003.
50. K. Kamijo et al. HUP-ML: Human proteome markup language for proteomics database. *Journal of the Mass Spectrometry Society of Japan*, 51(5):542–549, 2003.
51. J. Kang and J. F. Naughton. On schema matching with opaque column names and data values. In *Proceedings of SIGMOD*, pages 205–216, 2003.
52. V. Kashyap and A. Sheth. Semantic and schematic similarities between database objects: a context-based approach. *The VLDB Journal*, 5(4):276–304, 1996.
53. J. A. Larson, S. B. Navathe, and R. Elmasri. A theory of attributed equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, 1989.
54. D. Lenat. CYC: a large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.
55. V. Lopez, E. Motta, and V. Uren. PowerAqua: Fishing the Semantic Web. In *Proceedings of ESWC*, 2006.
56. V. Lopez, M. Pasin, and E. Motta. Aqualog: An ontology-portable question answering system for the semantic web. In *Proceedings of ESWC*, pages 546–562, 2005.
57. S. MacLane. *Categories for the Working Mathematician*. Springer, second edition, 1998.
58. J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proceedings of VLDB*, pages 49–58, 2001.
59. F. McNeill. *Dynamic Ontology Refinement*. PhD thesis, University of Edinburgh, 2006.
60. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm. In *Proceedings of ICDE*, pages 117–128, 2002.
61. E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. Observer: An approach for query processing in global information systems based on interoperability between pre-existing ontologies. In *Proceedings of CoopIS*, pages 14–25, 1996.
62. A. Miles and D. Brickley. SKOS core vocabulary. Technical report, World Wide Web Consortium (W3C), http://www.w3.org/TR/2005/swbp-skos-core-spec, 2005.

63. D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-Peer computing. Technical report, HP Laboratories, Palo Alto, March 2002.
64. G. Modica, A. Gal, and H. Jamil. The use of machine-generated ontologies in dynamic information seeking. In *Proceedings of CoopIS*, pages 433–448, 2001.
65. U. Nambiar and S. Kambhampati. Answering imprecise queries over web databases. In *Proceedings of VLDB*, pages 1350–1353, 2005.
66. I. Niles and A. Pease. Towards a standard upper ontology. In *Proceedings of FOIS*, pages 2–9, 2001.
67. N. Noy. Semantic Integration: A survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70, 2004.
68. N. Noy and M. Musen. Anchor-PROMPT: using non-local context for semantic matching. In *Proceedings of the workshop on Ontologies and Information Sharing at IJCAI*, pages 63–70, 2001.
69. P. G. Pedrioli et al. A common open representation of mass spectrometry data and its application to proteomics research. *Nature Biotechnology*, 22:1459–1466, 2004.
70. E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, (10(4)):334–350, 2001.
71. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proceedings of ACM SIGCOMM*, pages 161–172, 2001.
72. A. L. Rector, A. Gangemi, E. Galeazzi, and A. J. Glowinski. The GALEN CORE model schemata for anatomy: Towards a re-usable application-independent model of medical concepts. In *Proceedings of MIE*, 1994.
73. David Robertson. A lightweight coordination calculus for agent systems. In *Proceedings of DALT*, pages 183–197, 2004.
74. C. Rosse and J. L. Mejino. A reference ontology for biomedical informatics: the foundational model of anatomy. *Journal of Biomedical Informatics*, (36(6)):478–500, 2003.
75. T. Saracevic. Evaluation of evaluation in information retrieval. In *Proceedings of ACM SIGIR*, pages 138–146, 1995.
76. M. Schorlemmer and Y. Kalfoglou. Progressive ontology alignment for meaning coordination: An information-theoretic foundation. In *Proceedings of AAMAS*, pages 737–744, 2005.
77. H. Schutze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.
78. J. Seidenberg and A. Rector. Web ontology segmentation: Analysis, classification and use. In *Proceedings of WWW*, 2006. to appear.
79. A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.
80. P. Shvaiko and J. Euzenat. Ontology matching: virtual thematic information resource. www.OntologyMatching.org, September 2005.
81. P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics*, (IV):146–171, 2005.
82. P. Shvaiko, F. Giunchiglia, P. Pinheiro da Silva, and D. McGuinness. Web explanations for semantic heterogeneity discovery. In *Proceedings of ESWC*, pages 303–317, 2005.
83. M.K. Smith, C. Welty, and D.L. McGuinness. OWL web ontology language guide. Technical report, World Wide Web Consortium (W3C), http://www.w3.org/TR/2004/REC-owl-guide-20040210/, February 10 2004.
84. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM*, pages 149–160, 2001.

85. H. Stuckenschmidt. Approximate information filtering with multiple classification hierarchies. *International Journal on Computational Intelligence Applications*, 2002.
86. Y. Sure, O. Corcho, J. Euzenat, and T. Hughes, editors. *Evaluation of Ontology-based Tools*. Proceedings of the international workshop on Evaluation of Ontology-based Tools (EON), 2004.
87. M. Uschold. Achieving semantic interoperability using RDF and OWL - v4, 2005.
88. H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Huebner. Ontology-based integration of information - a survey of existing approaches. In *Proceedings of the workshop on Ontologies and Information Sharing at IJCAI*, pages 108–117, 2001.
89. H. Yu and E. Agichtein. Extracting synonymous gene and protein terms frombiological literature. *Bioinformatics*, 19:340–349, 2003.
90. I. Zaihrayeu. *Towards Peer-to-Peer Information Management Systems*. PhD thesis, International Doctorate School in Information and Communication Technologies, UNITN, March 2006.
91. A. V. Zhdanova and P. Shvaiko. Community-driven ontology matching. In *Proceedings of ESWC*, pages 34–49, 2006.