Journal of Computer Science 4 (6): 437-446, 2008
ISSN 1549-3636
© 2008 Science Publications

# Feature Analysis of Ontology Mediation Tools

[1]Ravi Lourdusamy and [2]Gopinath Ganapathy
[1]Department of Computer Science, Sacred Heart College, Tirupattur, Vellore, Tamil Nadu, India
[2]Department of Computer Science, Bharathidasan University, Trichirapalli, Tamil Nadu, India

**Abstract:** Ontology mediation is enabled through interoperability of semantic data sources. It helps data sharing between heterogeneous knowledgebase and reuse by semantic applications. Ontology mediation includes operations such as, mapping, alignment, matching, merging and integration. After briefly describing these operations, this study selectively discusses set of methods, tools and data integration systems. It provides the researchers a comprehensive understanding of methods and tools intended for ontology mediation.

**Key words:** Ontology Mapping, Ontology Alignment, Ontology Merging, Ontology Integration and Ontology Mismatch

## INTRODUCTION

In any semantic solution, data is annotated using ontologies. Ontologies are shared specifications and therefore the same ontologies can be used for the annotation of multiple data sources, like web pages, XML documents, relational databases and so on. Their shared terminologies enable a certain degree of interoperability between the data sources using the same ontologies. To enable such an interoperation, mediation is required between the ontologies.

**Terminologies:** An ontology mapping M is a declarative specification of the semantic overlap between two ontologies $O_S$ and $O_T$. The correspondences between different entities of the two ontologies are typically expressed using some axioms formulated in a specific mapping language. Mapping can be unidirectional or bi-directional. The different phases in the generic mapping process as in[1] is shown in Fig. 1.

**Import of ontologies:** Ontologies can be specified in different languages, which indicate a need to convert them to a common format so that the mapping can be specified. Furthermore, the ontologies need to be imported in the tool, which is used to specify the mapping. Finding Similarities: Many systems use the match operator to automatically find similarities between ontologies. For any two-source ontology, the match operator returns the similarities between ontologies. Specifying Mapping: After similarities between ontologies have been found, the mapping between the ontologies needs to be specified.

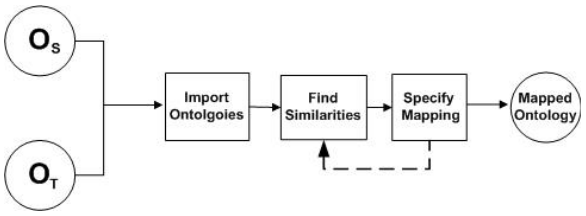


Fig. 1: Mapping Process

The automated or semi-automated discovery of correspondences between two ontologies is called ontology alignment. Ontology alignment is the task of creating links between two original ontologies. Ontology alignment is made, if the sources found to be consistent with each other, but are kept separate or when sources are from the complementary domains. Ontology matching is the process of discovering similarities between two source ontologies. The result of matching operation is a specification of similarities between two ontologies. Ontology matching is carried out through the application of match operator[2].

In ontology merging a new ontology is created which is the union of source ontologies in order to capture all the knowledge from the original ontologies. There are two different approaches in ontology merging. In the first approach, the input of the merging process is a collection of ontologies and the outcome is, one new merged ontology which captures the original ontologies, as given in Fig. 2.

In the second approach the original ontologies are not replaced, but rather a view called bridge ontology is created which imports the original ontologies and specifies the correspondence using bridge axioms as in Fig. 3.

**Corresponding Author:**Ravi Lourdusamy, Department of Computer Science, Sacred Heart College, Tirupattur, Vellore 635 601

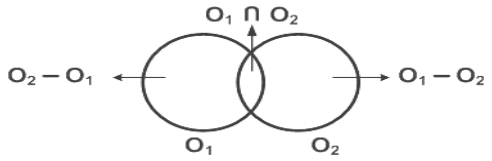


Fig. 2: Output of Merging Process (Approach 1)

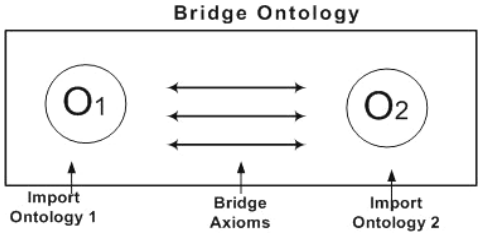


Fig. 3: Output of Merging Process (Approach 2)

Ontology integration is the process of generating a single ontology in one subject from two or more existing and different ontologies in different subjects. The different subjects of the different ontologies may be related. Some change is expected in a single integrated ontology[3].

**Ontology Mismatches:** An important issue in the approaches of ontology mediation is the location and specification of the overlap and the mismatches between concepts, relations, and instances in different ontologies. Based on the work by Klein[4], the mismatches that might occur between different ontologies are Conceptualization mismatches and Explication mismatches. The hierarchy of ontology mismatch is given in Fig. 4.

Conceptualization mismatches are mismatches of different conceptualization of the same domain. Conceptualization mismatches fall in two categories; namely a scope mismatch and a mismatch in the model coverage and granularity. A scope mismatch occurs when two classes have some overlap in their extensions (the set of instances), but the extensions are not exactly the same. There is a mismatch in the model coverage and granularity, if there is a difference in (a) the part of the domain that is covered by both ontologies (for example, the ontologies of university employees and the students), or (b) the level of detail with which the model is covered (for example, one ontology might have one concept 'person', whereas another ontology distinguishes between young person, middle-aged person and old person).

Explication mismatches are mismatches in the way of specifying a conceptualization. Explication

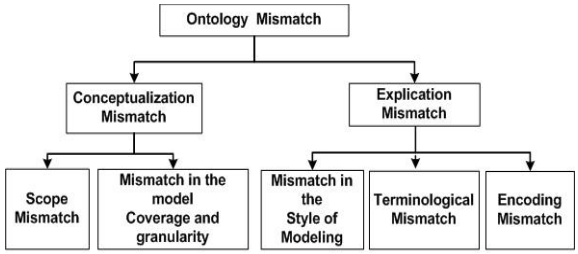


Fig. 4: Hierarchy of Ontology Mismatch

mismatches fall in three categories namely mismatch in the style of modeling, terminological mismatch and encoding mismatch. A mismatch in the style of modeling occurs if either (a) the paradigm used to specify a certain concept is different (for example, time specified in intervals is different from the time specified in points in time), or (b) the way the concept is described differs (for example, using subclasses versus attributes to distinguish groups of instances). A terminological mismatch occurs when two concepts are equivalent, but they are represented using different names (Synonyms) or when the same name is used for different concepts (Homonyms). An encoding mismatch occurs when values in different ontologies are encoded in a different way (for example, distance measure specified in kilometers and miles).

**A comparison on ontology mediation tools and systems:** A specific framework does not exist for comparison of ontology mediation tools[5] nor direct comparison of ontology mediation tools be possible[6]. But set of criteria to compare the ontology mediation tools is proposed as in[1,3]. The comparison of tools on ontology mediation is made on the following criteria, namely input and output requirements, level of user interaction, ontology language, mapping concept, automation support, and the level of implementation.

The ontology mediation approaches are grouped into three categories as in[1], namely methods and tools, data integration systems and specific techniques. Methods and tools approach describes the special purpose methods and tools. In this approach the tools from ontology matching (GLUE)[7] to ontology merging (PROMPT)[8] and ontology mapping (MAFRA)[9] are discussed. Data integration systems approach describes the tools ONION[10] and OBSERVER[11]. These integration systems are all comprehensive in the sense that they typically have different types of functionality. Specific techniques approach describes a specific technique. In this approach, FCA Merge[12],

Ontomorph[13] and QOM[14] are discussed in this research.

## MATERIALS AND METHODS

**PROMPT:** The PROMPT[8] suite contains of a set of tools that have had an important impact in the area of merging, aligning and versioning of ontologies. The different tasks in multiple ontology management are closely interrelated and share several components and heuristics. Thus, tools for supporting some of the tasks in the context of multiple ontology management can benefit greatly from their integration with others. The key components of the PROMPT suite have been developed as extensions (plug-ins) of the Protégé 2000 ontology development environment. The following components are distinguished in PROMPT suite as in Fig. 5. The suite includes an ontology merging tool (iPROMPT, formerly known as PROMPT), an ontology tool for finding additional points of similarity between ontologies for other tools like iPROMPT (Anchor PROMPT), an ontology versioning tool (PROMPT Diff) and a tool for factoring out semantically complete subontologies (PROMPTFactor).

iPROMPT is an interactive ontology merging tool, which helps users in the ontology merging task by providing suggestions about which elements can be merged, by identifying inconsistencies and potential problems and suggesting possible strategies to resolve these problems and inconsistencies. Anchor PROMPT extends the performances of tools like iPROMPT determining additional points of similarities between ontologies that are not identified by iPROMPT. PROMPTDiff compares two versions of ontology and identifies structural differences between different versions of the same ontology. PROMPTFactor is a tool that enables users to create a new ontology, factoring out part of an existing ontology.

One of the major contributions to the developments of PROMPT suite was the identification of an important overlap in the functionality of its tools and the implementation of an integrated approach where all these tools benefit from each other. The PROMPT knowledge model is frame-based and it is designed to be compatible with OKBC[15]. Since this knowledge model is extremely general, and many existing knowledge representation systems have knowledge models compatible with it, the solutions to merging and alignment produced by PROMPT can be applied over a variety of knowledge representation systems.

Figure 6 illustrates the PROMPT ontology merging and ontology alignment algorithm. PROMPT takes two ontologies as input and guides the user in the creation of a merged ontology as output. The gray boxes indicate the actions performed by PROMPT. The white box indicates the actions performed by the user. First,

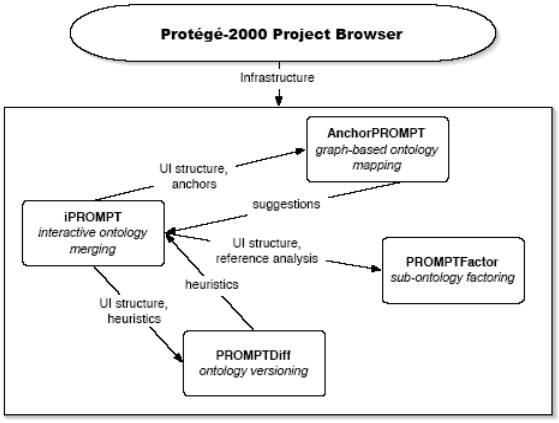


Fig. 5: The PROMPT suite infrastructure and interactions between tools

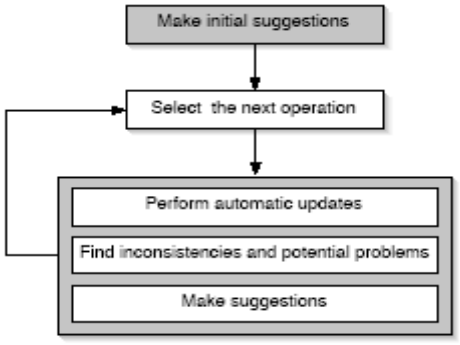


Fig 6: The flow of the iPROMPT Algorithm

PROMPT creates an initial list of matches based on class names. Then the following iterative cycle happens: The user triggers an operation by either selecting one of PROMPT's suggestions from the list or by using an ontology-editing environment to specify the desired operation directly, and PROMPT automatically executes additional changes based on the type of the operation, generates a list of suggestions for the user based on the structure of the ontology around the arguments to the last operation, and determines conflicts that the last operation introduced in the ontology and finds possible solutions for those conflicts. The important features of PROMPT are setting the preferred ontology, maintaining the user's focus, providing feedback to the user, logging and reapplying the operations.

The goal of AnchorPROMPT is to augment the results of methods that analyze only local context in ontology structures, such as Chimaera and iPROMPT, by finding additional possible points of similarity between ontologies. To do this, AnchorPROMPT requires that the other tool or the user provide an initial set of related terms. Following a graph perspective, the tool establishes a set of paths that connects the terms of ontology that are related with the terms of the other one. The third element of the suite is PROMPTDiff, which is used to compare the structure of two versions of a particular ontology and which identifies the frames (classes, slots or instances) that have no changes, frames with only changes in their properties, and frames that have also changed in other parts of their definitions. Tools like CVS influence the name of the tool, PROMPTDiff, which is a version control system that is used to maintain the history of program source code files. This tool includes facilities to discover changes between versions of a document.

The last element of the PROMPT suite is the tool PROMPTFactor that allows users to extract from the larger ontology the elements that the user is interested in, in a way that also copies all the terms required for preserving the semantics of the description. The authors of the tool call this process "factoring subontologies". During the analysis of the PROMPT suite, it is found that the tool has some limitations in the area of ontology evolution and versioning. PROMPTDiff only detects differences between two versions using a structural difference and the description of the differences between two versions of an ontology that PROMPTDiff offers is limited.

**MAFRA:** MAFRA (MApping FRAmework for distributed ontologies)[9] is a framework defined for mapping distributed ontologies on the semantic web. MAFRA has been implemented as a plug-in of KAON[16] and introduces two new concepts namely, semantic bridges and service-centric approaches.

A semantic bridge is defined as a declarative representation of a semantic relation between source and target ontologies entities[17]. A semantic bridge provides the necessary mechanisms to transform instances and property fillers of source ontology into instances and property fillers of target ontology. Semantic bridges are similar to the notion of articulation structures (the point of linkage between two aligned ontologies) in[4] and articulation ontologies in ONION[18,19]. Each semantic bridge has an associated transformation service that determines the transformation procedure and the information the user must provide to the transformation engine. Each service

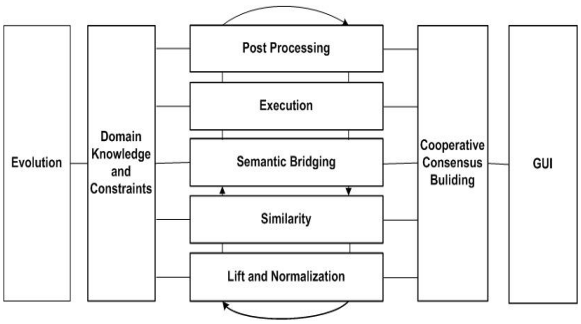


Fig. 7: MAFRA Conceptual Architecture

is characterized by a set of arguments, which in turn are characterized by name, type, optionality and location. Services are not only responsible for the transformation capabilities but also for the validation of argument values and semi-automatic mapping. The service-oriented approach complements the semantic bridges mechanisms providing the transformation services necessary to perform the mapping transformations.

Figure 7 outlines the conceptual architecture of MAFRA system. In this architecture, a set of modules is organized along horizontal and vertical dimensions. Horizontal modules correspond to five fundamental phases namely, lift and normalization, similarity, semantic bridging, execution and post-processing. The vertical modules correspond to four phases; namely, evolution, domain knowledge and constraints, cooperative consensual building and GUI. The vertical modules interact with the horizontal modules during the entire ontology mapping process. In the lift and normalization phase, lifting refers to the process of importing the ontologies in a formal uniform representation (RDF-Schema) for facilitating later operations. The next step after lifting is the normalization of the vocabulary of the ontologies by eliminating lexical and semantic differences (like special characters, uppercase letters and acronyms). In similarity phase, similarities between ontology entities are calculated as a support for mapping discovery. In semantic bridging phase after identifying the similarities between entities from different ontologies, the similar entities are semantically bridged; i.e. correspondences are established between them for enabling the transformation of instances of one source entity to instances of one target entity. MAFRA includes a formal representation to specify the mappings. The formalism that is used to describe the semantic bridges is based on DAML+OIL ontology, the so called, Semantic Bridging Ontology (SBO). The

result is close to the notion of articulation ontology in ONION. A mapping is a set of instances of the semantic bridges described by this ontology.

In the execution phase, when the mappings (bridges) are specified, the next step is to exploit them in a meaningful way. MAFRA addresses only the task of instance transformation. The execution module actually transforms instances from the source ontology representation into the target ontology representation by evaluating the transformation functions associated with the bridges defined in the previous stage. In the post-processing step based on the execution results, the mapping specification is again analyzed in order to improve the quality of the instance transformation task. In the evolution step the changes in the source and target ontologies are synchronized with the semantic bridges defined by the semantic bridge module. In the cooperative consensus-building phase from the multiple alternative possible mappings, the tool helps to setup a consensus between the various proposals of people involved in the mapping task. In the domain constraints and background knowledge phase, the tool allows users to include extra information in order to improve the quality of the mapping. In the GUI, visualization of the elements of the source and target ontologies makes the mapping task easier.

**GLUE:** GLUE[7] is a system, which employs machine-learning technologies to semi-automatically create mappings between heterogeneous ontologies, where an ontology is seen as taxonomy of concepts. GLUE focuses on finding 1-to-1 mappings between concepts in taxonomies. The similarity of two concepts A and B in the two taxonomies $O_1$ and $O_2$ is based on the sets of instances that overlap between the two concepts. In order to determine whether an instance of concept B is also an instance of concept A, first a classifier is built using the instances of concept A as the training set. This classifier is now used to classify the instances of concept B. The classifier then decides for each instance of B, whether it is also an instance of A or not. Based on these classifications, four probabilities are computed. These four probabilities are used to compute the joint probability distribution for the concepts A and B, which is a user-supplied function. Two possible functions for the joint probability distribution are Jaccard Coefficient and Most Specific Parent (MSP). In Jaccard Coefficient, the similarity measure is computed by dividing the probability that an instance is in the intersection of two concepts by the probabilit that an instance is in the union of the concepts (P $(A \cap B)/P(A \cup B)$), which intuitively corresponds to the function of relevant instances, which are both in A and B. In most-specific parent, the similarity measure is positive for any parent B of A and it is

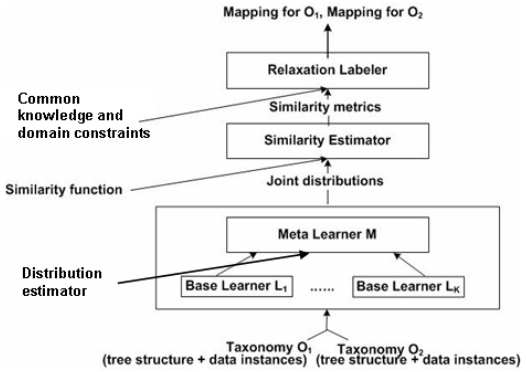


Fig. 8: The GLUE Architecture

the highest for the most specific parent, i.e., the concept $B_{MSP}$ that represents the smallest superset of A. The general architecture of GLUE system as in Fig. 8 consist of three phases namely the distribution estimator, the similarity estimator and the relaxation labeler.

The distribution estimator takes as input the two taxonomies $O_1$ and $O_2$ together with their instances and applies machine learning to compute the four probabilities. Currently, the distribution estimator uses a content learner, which learns a classifier, based on the textual context of the instances and a name learner, which learns a classifier based on the name of the instance.

It is possible to plug in different learners for different aspects using a meta-learner, which uses a certain function to incorporate the predictions from all learners into an overall prediction. The similarity estimator applies a user-supplied function, such as the Jaccard Coefficient or MSP and computes a similarity value for each pair of concept ($A \in O_1$, $B \in O_2$). The relaxation labeler takes as input the similarity values for the concepts from the taxonomies and searches for the best mapping configuration, exploiting user supplied domain specific constraints and heuristics.

**Data Integration Systems:**

**ONION:** ONION (ONtology compositION)[18,19] is an architecture based on a sound formalism to support a scalable framework for ontology integration that uses a graph-oriented model for the representation of ontologies. The special feature of this system is that it separates the logical inference engine from the representation model (the graph representation) of the ontologies. This allows for the accommodation of different inference engines in the architecture.

In ONION, there are two types of ontologies, individual ontologies, referred as source ontologies and

articulation ontologies, which contain the concepts and relationships expressed as articulation rules (rules that provide link across domains). Articulation rules are established to enable knowledge interoperability, and to bridge the semantic gap between heterogeneous sources. They indicate which concepts individually or in conjunction, are related in the source ontologies. The SKAT (Semantic Knowledge Articulation Tool)[19] uses the structure of these graphs together with term-matching and other rules to propose articulation rules for the articulation ontologies. The source ontologies are reflected in the system by the use of wrappers. The mapping between ontologies is executed by ontology algebra[18,20]. Such algebra consists of three operations, namely, intersection, union and difference. The objective of ontology algebra is to provide the capability for interrogating many largely semantically disjoint knowledge resources, given the ontology algebra as input.

The intersection produces on ontology graph, which is the intersection of the two source ontologies with respect to a set of articulation rules, generated by an articulation generator function. The intersection determines the portion of knowledgebase that deal with similar concepts. The union operator generates a unified ontology graph comprising the two original ontology graphs connected by the articulation. The union presents a coherent connected and semantically sound unified ontology, if the original ontologies are consistent. The difference operator, to distinguish the difference between two ontologies ($O_1$-$O_2$) is defined as the concepts and relationships of the first ontology that have not been determined to exist in the second. This operation allows a local ontology maintainer to determine the extent of one's ontology that remains independent of the articulation with other domain ontologies so that it can be independently manipulated without having to update the articulation.

ONION tries to separate as much as possible the logical inference engine from the representation model of the ontologies, allowing the accommodation of different inference engines. It also uses articulation of ontologies to interoperate among ontologies. These articulation ontologies can be organized in a hierarchical fashion. The ontology mapping is based on the graph mapping and at the same time, domain experts can define a variety of fuzzy matching.

The ONION architecture depicted in Fig. 9 consists of four components namely data layers, viewer, query system and articulation engine. The data layer contains the wrappers for the external sources and the

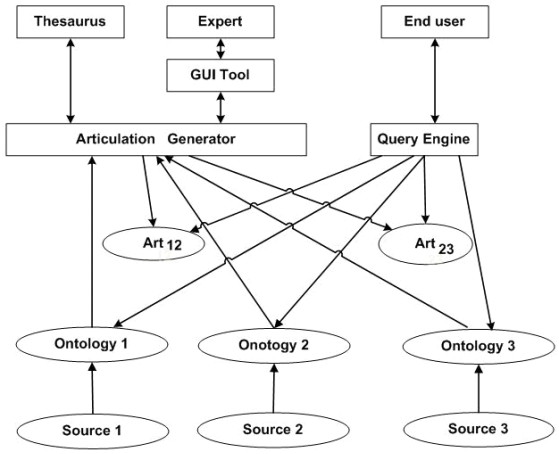


Fig. 9: The Component of the ONION System

articulation ontologies that form the semantic bridges between the sources. The viewer is the user interface, which visualizes both the source and the articulation ontologies. The query system translates queries formulated in term of articulation ontology into a query execution plan and executes the query. The articulation engine takes articulation rules proposed by the SKAT and generates sets of articulation rules, which are forwarded to the expert for confirmation.

**OBSERVER:** OBSERVER (Ontology Based System Enhanced with Relationships for Vocabulary hEterogeneity Resolution)[11] is a system, which is intended to overcome problems with heterogeneity between distributed data repositories by using component ontologies and the explicit relationships between these components. OBSERVER presents on architecture consisting of component nodes, each of which has its own ontology and the Inter-ontology Relationship Manager (IRM), which maintains mappings between the ontologies at the different component nodes. Besides the ontology, each component node contains a number of data repositories along with mappings to the ontology, to enable semantic querying of data residing in these repositories. When other components need to be queried, the IRM provides mappings to ontologies of other component nodes in order to enable querying. The user views the data in the system through its own local ontology, located at the user's component node.

OBSERVER uses a component-based approach to ontology mapping. It provides brokering capabilities across domain ontologies to enhance distributed ontology querying thus avoiding the need to have a

global schema or collection of concepts. OBSERVER uses multiple pre-existing ontologies to access heterogeneous, distributed and independently developed data repositories. Each repository is described by means of one or more ontology expressed using the Description Logic (DL) system CLASSIC. The information requested from OBSERVER is expressed according to the user's domain ontology, also expressed using DL. DL allows matching the query with the available relevant data repositories, as well as translating it to the ontologies used in the local repositories.

The system contains a number of component nodes, one of which is the user node. Each node has an ontology server that provides definitions for the terms in the ontology and retrieves data underlying the ontology in the component node. If the user wants to expand his query over different ontology servers, the original query needs to be translated from the vocabulary of the user's ontology into the vocabulary of another's component ontology. A method for evaluating the information loss for the case of inexact translations was developed, with the purpose of enabling the system to choose the best among alternative translations. The loss of information threshold is used by the query processor, which discards queries exceeding the threshold. The information loss is computed based on the commonly encountered metrics in information retrieval, precision and recall.

An IRM provides the translations between the terms among the different component ontologies. The IRM effectively contains a one-to-one mapping between any two-component ontologies. This module is able to deal with Synonym, Hyponym, Hypernym, Overlap, Disjoint and Covering inter-ontology relationships. Furthermore, the IRM is also able to deal with extensional relationships between ontologies through the use of transformer functions. The user submits a query to the query processor in its own component node. Each component node has a query processor. The query processor first uses the local ontology server to translate the query into queries on the local data repositories and then execute them, after which the user can choose to incrementally increase the query to multiple ontologies. The query processor then uses the IRM to translate the query into terms used by the other component ontologies and retrieves the results from the ontology servers at the other component nodes.

The ontology server can be queried in two ways. Information about the ontology itself can be retrieved and the ontology server can answer queries formulated

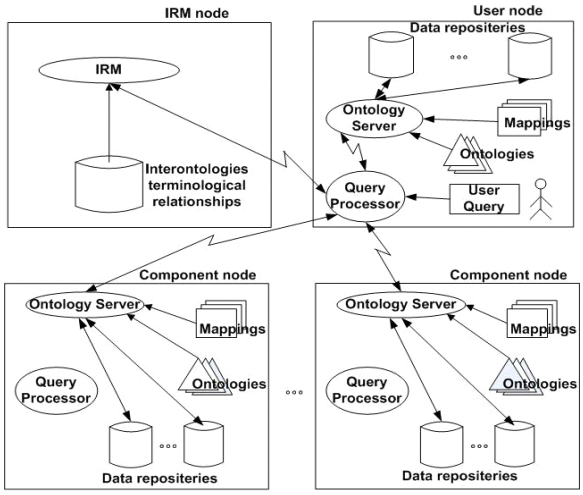


Fig 10: The general OBSERVER architecture

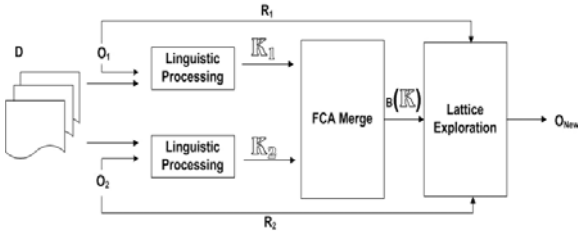


Fig 11: The Ontology Merging Method

over an ontology using the mappings to the different data sources and the wrappers available for each data source. The query capabilities of each data source are consulted by the ontology server, which creates a query plan and invokes the wrappers to retrieve the data from the sources. In principle, only the local ontology server is queried initially. The user can then choose to incrementally expand the query over multiple ontologies in order to retrieve more results for the query. The OBSERVER architecture depicted in Fig. 10 consists of a number of component nodes and the IRM node.

**Specific Techniques:**

**FCA MERGE:** FCA Merge[12] is a bottom up approach for ontology merging. FCA Merge tool is based on application specific instances of the two given ontologies $O_1$ and $O_2$ that are to be merged. The overall process of merging two ontologies is depicted in Fig 11. The process of FCA Merge consists of three steps, namely (i) instances extraction and computing of two formal contexts $K_1$ and $K_2$, (ii) the FCA Merge core algorithm that derives a common context and computes

a concept lattice and (iii) the generation of the final merged ontology based on the concept lattice.

FCA Merge tool takes as input data the two ontologies and a set D of natural language documents. The documents have to be relevant to both ontologies, so that the documents are described by the concepts contained in the ontology. The documents may be taken from the target application, which requires the final merged ontology. Instances are extracted from the document in D. This automatic knowledge acquisition step returns, for each ontology, a formal context indicating which ontology concept appear in which documents. The extraction of the instances from documents is necessary because there are usually no instances, which are already classified by both ontologies. However, if this situation is given one can skip the first step and use the classification of the instances directly as input for the two formal contexts. The second step of ontology merging approach comprises the FCA Merge core algorithm. The core algorithm merges two contexts and computes a concept lattice form the merged context using FCA techniques. More precisely, it computes a pruned concept lattice, which has same degree of detail as the two source ontologies. Instance extraction and the FCA Merge core algorithm are fully automatic.

The final step of deriving the merged ontology from the concept lattice requires human interaction. Based on the pruned concept lattice and the sets of relation names $R_1$ and $R_2$, the ontology engineer creates the concepts and relations of the target ontology. Graphical means of the ontology-engineering environment is offered using OntoEdit for supporting this process.

**ONTOMORPH:** The OntoMorph[13] system aims to facilitate ontology merging and the rapid generation of knowledge base translators. It combines two powerful mechanisms to describe KB transformations. The first of these mechanisms is syntactic rewriting via pattern-directed rewrite rules that allow the concise specification of sentence-level transformations based on pattern matching, and the second mechanism involves semantic rewriting which modulates syntactic rewriting via semantic models and logical inference. The integration of ontologies can be used on any mixture of syntactic and semantic criteria. In the syntactic rewriting process, input expressions are first tokenized into lexemes and then represented as syntax trees, which are represented internally as flat sequences of tokens and their structure only exists logically. OntoMorph's pattern language and execution model is

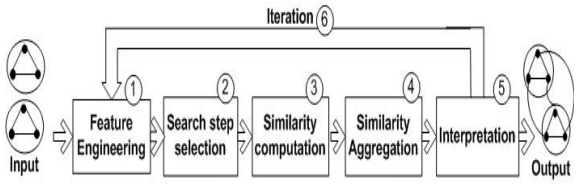


Fig. 12: Mapping Procees

strongly influenced by PLISP. The pattern language can match and de-structure arbitrarily nested syntax trees in a direct and concise fashion. Rewrite rules are applied to the execution model. For the semantic rewriting process OntoMorph is built on top of the Power Loom knowledge representation system which is a successor to the Loom system. Using semantic import rules, the precise image of the source KB semantics can be established within power Loom.

**QOM:** The QOM (Quick Ontology Mapping) tool[14] represents an approach that considers both the quality of mapping results as well as the run-time complexity. The hypothesis is that mapping algorithms may be streamlined such that the loss of quality is marginal, but the improvement of efficiency is so tremendous that it allows for the ad-hoc mapping of large-size, light weight ontologies. To substantiate the hypothesis a number of practical experiments were performed. The steps of process model is shown in Fig 12 define QOM. The process starts with two ontologies which are going to be mapped onto one another, as its input. Feature Engineering transforms the initial representation of ontologies into a format usable for the similarity calculations. For instance, the subsequent mapping process may only work on a subset of RDFS primitives. Selection of Next Search steps: The derivation of ontology mappings takes place in a search space of candidate mappings. This step may choose, to compute the similarity of a restricted subset of candidate concepts pairs $\{(e, f)|e\in O_1, f\in O_2\}$ and to ignore others. *Similarity computation* determines similarity values between candidate mappings (e, f) based on their definitions in $O_1$ and $O_2$ respectively. *Similarity Aggregation:* In general there may be several similarity values for a candidate pair of entities (e, f) from two ontologies $O_1$ and $O_2$. These different similarity values for one candidate pair must be aggregated into a single

Table 1 Comparative Matrix of Ontology Mediation Tools and Systems

| Tool / criteria | PROMPT | MAFRA | GLUE | ONION | OBSERVER | FCA-Merge |
|---|---|---|---|---|---|---|
| Input | Two ontologies | Two ontologies | Two taxonomies with their data instances in ontologies | Terms in two ontologies | Two ontologies | Two input ontologies and a set of documents of concepts |
| Output | Merged ontology | Mappings of the two ontologies | A set of pairs of similar concepts | Sets of articulation rules | Inter Relationships Manager | Merged ontology |
| User Interaction | The user accepts or rejects or adjusts system's suggestion | The domain expert interface with the similarity and semantic bridging modules and it has graphical interface | User defined mappings for training data, Similarity measure, Setting up the learner weight, and analyzing system's match suggestion | A human expert chooses or deletes or modifies suggested matches using a GUI tool | Query based interface | The domain expert interface with background knowledge |
| Ontology language | RDFS, OWL | RDFS | Taxonomies | Directed labeled graphs and Horn Clauses | Description logics ( CLASSIC ) | - |
| Mapping Concept | Heuristic based analyzer | Semantic Bridging Ontologies(SBO) | Similarity measures | Articulation rules | Extended relational algebra for mapping ontology DB and DL and transformer functions for mapping | Linguistic analysis and algorithm for computation for pruned concept lattice |
| Automation Supports | Name and structural Matching | Lexical and structural matching and semi-automatic creation of mappings | Multi-strategy machine learning approach | Term and structural matching using Semantic (SKAT ) Annotation Tool (SKAT ) | Query Processing | |
| Implement Status | Version 2.1.1 | Two prototypes have been implemented | Research prototype | Research prototype for the unification of heterogeneous ontologies | Research prototype for the access of distributed heterogeneous data source in the area of bibliographic data | Research prototype |

aggregated similarity value. Interpretation uses the individual or aggregated similarity values to derive mappings between entities from O1 and O2. Some mechanisms here are to use thresholds for similarity mappings to perform relaxation labeling or to combine structural and similarity criteria. Iteration: Several algorithms perform iteration over the whole process in order to bootstrap the amount of structural knowledge.

Iteration may stop when no new mappings are proposed. Eventually, the output returned is a mapping table representing the relation map of O1 and O2.

## CONCLUSION

The comparison matrix presented in table 1 captures the important features of the tools as per the framework that is discussed in this research. Through this survey and analysis, the research provides a comprehensive understanding of ontology mediation and points to various research aspects specific to the roles on ontology mediation.

## REFERENCES

1. Livia, P., F. Cristina, S. Francois and Jos de Bruijn, Francisco Martin-Recuerda, Dimitar Manov, Marc Ehrig, D4.2.2 State-of-

the-art survey on Ontology Merging and Aligning V2, 2006, Digital Enterprise Research Institute, University of Innsbruck, pp: 1-121.
2. Erhard, R. and A.B. Philip. 2001. A survey of approaches to automatic schema matching, VLDB J. Very Large Databases, 10: 334-350.
3. Namyoun, C., I.I. Yeol Song and H. Hyoil, 2006. A Survey on Ontology Mapping, SIGMOD Record, Vol 35, No 3, Sep 2006 , pp : 34-41
4. Klein, M., Combining and relating Ontologies: An Analysis of Problems and Solutions, In workshop on Ontologies and information sharing, IJCAI'01, August 4-5, 2001, Seattle, USA.
5. Yannis, K. and S. Marco, 2003. Ontology mapping: The state of the art, The knowledge Engineering Review, 18: 1-31
6. Natalya, F. N. and A. M. Mark, 2002. Evaluating Ontology-mapping tools: Requirements and experience, Proceedings of the workshop on evaluation of ontology at EKAW'02(EOEN 2002), Siguenza, Spain.
7. Doan, A., J. Madhavan, P. Domingos and A. Halevy, 2002. Learning to map between Ontologies on the Semantic Web, Proceedings of the 11th International WWW Conference [WWW 2002], May 7-11, 2002, Honolulu, Hawaii, USA.
8. Noy N.F. and M.A. Muser, 2003. The PROMPT suite: Interactive tools for Ontology merging and mapping, Int. J. Human Comput. Stud, 59: 983–1024
9. Alexander, M., M. Boris, S. Nuno and V. Raphael , 2002. MAFRA A Mapping Framework for Distributed Ontologies, Proceedings of 13th European conference on knowledge Engineering and knowledge management EKAW-2002, Madrid, Spain.
10. Mitra, P., G. Wiederhold and M. Kersten, 2000. A Graph Oriented Model for Articulation of Ontology Interdependencies, In proceedings Conference on Extending Database Technology (EDBT 2000), Konstanz, Germany.
11. Mena, E., A. Illarramendi, V. Kashyapand and A. Sheth, 2000. Observer: An approach for query processing in global information systems based on inter operation across pre-existing Ontologies, Distributed and parallel Databases, An int. J, 8: 223-271.
12. Stumme, G. and A. Maedche, FCA-Merge: Bottom-up Merging of Ontologies, In seventh international Conference on Artificial Intelligence (IJCAI'01), seattle, WA, USA, 200, pp: 225-230
13. Hans, C. 2000. OntoMorph : A translation system for symbolic knowledge, Proceedings of the 7th international conference on Principles of Knowledge Representation and Reasoning, Brecknridge, Colorado, USA, Pp: 471-482,
14. March E. and S. Steffen, 2004. Quick ontology mapping, Hiroshima, Japan
15. Chaudhri, V.K., A. Farquhar, R. Fikes, P.D. Karp and J.P. Rice, 1998. OKBC: A Programmatic Foundation for knowledge Base Interoperability, Proceedings of the 15th National conference on AI(AAAI-98), Madison, Wisconsin, Pp: 600-607.
16. http://kaon.semanticweb.org, KAON ontology management tool, university of karlsruhe.
17. Nuno S. and R. Joao, Ontology mapping for interoperability in semantic web, Proceedings of the IADIS International Conference www/Internet(ISWI 2003), 2003, Portugal, USA
18. Prasenjit M. and Gio Wiederhold, An Algebra For Semantic Interoperability Of Information Sources. In IEEE International Conference on Bio informatics and Biomedical Engineering, 2001, Pp: 174-182.
19. Prasenjit M, Gio wiederhold and J. Jan, Semi-Automatic Integration of Knowledge Sources. Proceedings of Fusion 99, California, USA, July 1999.
20. Gio wiederhold, An Algebra for Ontology Composition, Proceedings Monterey workshop on formal methods, US naval Post Grduate School, Monterey, CA,1994 Pp: 56-61.