Reconciliation of Ontology Mappings to Support Robust Service Interoperability

Asad Masood Khattak[1], Zeeshan Pervez[1], Khalid Latif[2], A. M. Jehad Sarkar[3], Sungyoung Lee[1], Young-Koo Lee[1]
[1] Department of Computer Engineering, Kyung Hee University, Korea,
{asad.masood, zeeshan, sylee}@oslab.ac.kr, yklee@khu.ac.kr

[2] School of Electrical Engineering and Computer Science, NUST, Pakistan,
khalid.latif@seecs.nust.edu.pk

[3] Department of Digital Information Engineering, Hankuk University of Foreign Studies, Korea,
jehad@hufs.ac.kr

***Abstract*— Information on web and in use of web services is increasing enormously even on hourly bases. On semantic web the information is represented in ontology. For system and services to share the information, a sort of mediation (i.e., mappings) is required. Mappings are established between the ontologies (information sources) of web services for resolving the terminological and conceptual incompatibilities. However, with the discovery of new knowledge in the field and accommodating the knowledge in domain ontologies makes the ontology to evolve from one consistent state to another. This consequently makes existing mappings between ontologies unreliable and staled due to the changes in resources. So there is a need for mapping evolution to eliminate discrepancies from the existing mappings. To re-establish the mappings between dynamic ontologies, existing systems restart the complete mapping process which is time consuming. The approach proposed in this paper provides the benefits of mapping reconciliation between the updated ontologies. It takes less time as compared to the existing systems. It only considers the changed resources and eliminates the staleness from the mappings. This approach uses the change history of ontology to drastically reduce the time required for reconciling mappings among ontologies. Experimental results with four different mapping systems using standard data sets are provided that validate our claims.**

I. INTRODUCTION

Information on web is increasing every day and placing heavy computation load on systems for accessing, interpreting, manipulating, maintaining, merging, integrating, inferring, and mining the information [8]. Ontology is *a formal, explicit specification of a shared conceptualization*. Ontology is the main source for the realization of semantic web and its services that help in well define meaning of resources and better understanding of processes between human and computer systems [13]. Service-Oriented Architecture (SOA) and Semantic Web Services Technology are getting mature and are widely adopted [4]. The meaningful and machine interpretable information contained in ontology help semantic web services for automated services discovery, selection, and interoperability [6]. However, since the services are usually provided by autonomous parties, so they have high interface, structural, and semantic heterogeneity among them for information storage and exchange [2, 5, 7, 8, 10, 12, 14, 15, 17, 20, and 22]. To overcome this issue we recognized the value of data and schema mapping [2, 3, 7, 10, 12, 15, 16, 17, and 18].

Use of ontology in systems dealing with information extraction from large complex structured information and web services can give valuable results [1, 2, 3, 7, 10, 16, and 21]. As use of ontology is increasing in Information Systems and Knowledge Sharing Systems, also increases the significance of ontology maintenance [8 and 12]. However, the large and complex structure with decentralized nature of web compels the communities to make their own ontologies for representation of information [5, 8, and 21]. This situation needs mediation among distributed and autonomous sources of information [2, 7, 8, 10, 15, 17, and 22].

As the number of information sources are increasing significantly, and also raise the importance of sophisticated information extraction and management of the heterogeneity among these information sources. So mapping can play a vital role to overcome these issues. Ontology mapping area is relatively a mature area of research to align multiple ontologies (information sources) together for the purpose to share information [2, 10, 15, 17, 19, and 22]. The mapping systems have two main concerns; one is the accuracy and the second is time to produce the mappings. Existing systems such as FOAM [7], Falcon [10], Lily [22], H-Match [2], Prompt [17], and MAFRA [15] are amongst the best matching and mapping systems. These systems consume lots of time when considering large knowledgebases such as; Google Classification[1] and Wiki Classification[2] for mapping. As the data sources evolve independently with flexible structure [9], that makes the already existing mappings among the data sources no more reliable. So there is a need for a system that supports mapping evolution as well for evolving ontologies. Existing systems re-start the complete mapping process and re-establish the mappings among evolved ontologies which is a time consuming process.

[1] http://www.google.com/Top/Reference/Libraries/Library_and_Information_Science/Technical_Services/Cataloguing/Classification/
[2] http://en.wikipedia.org/wiki/Taxonomic_classification

978-0-7695-4462-5/11 $26.00 © 2011 IEEE
DOI 10.1109/SCC.2011.83

Re-establishment of mappings is required for mapped ontologies which are dynamic and ready for change. Mostly, the existing systems take more time for re-establishment of mappings than the first matching process as they start the matching process from scratch; however, the changes in mapped schemas and in the regenerated mediation are not significant [9]. A less time consuming scheme for reconciling ontology mappings (mapping evolution) in dynamic/evolving ontologies is proposed in this research to support reliable and robust service interoperability and dynamic service discovery. Our approach uses the Change History Log (CHL) [11] (i.e., local, centralized, and distributed) for re-establishing the mappings with almost same accuracy and in less time as compared to existing systems. The technique is proposed to reduce the time for re-establishing the mappings between dynamic ontologies. CHL is used to store the changes of dynamic/evolving ontology and later use these stored changes for mapping reconciliation. The CHL helps in reconciliation of mappings in dynamic/evolving web ontologies to overcome the staleness problem of mappings. During reconciliation of ontology mapping, the need is only to update the out-dated mappings which save both time and resources. We have tested Falcon, Lily, FOAM, and Prompt algorithms on different online available data sets and then extended these algorithms with the proposed scheme by incorporating the use of CHL in them. Our proposed extensions are tested on the same data sets and we got drastic reduction in the amount of time required for reconciliation of mappings. Detail experimental results are provided that support our claims.

This paper is arranged as follows: Section II is detail discussion on matching algorithms. Section III presents our proposed time efficient approach for reconciliation of mapping in ontologies. Section IV is discussion on proposed scheme's experimental results in comparison with existing systems performance. Finally we conclude our discussion in Section V.

II. Related Work

Currently many research groups are working on ontology matching/mapping and have developed different systems. Falcon [10] is an overall infrastructure for Semantic Web ontology learning, matching, mapping, and aligning. As an infrastructure for Semantic Web ontologies, it is extensively used by the semantic web community for applications such as, providing fantastic technologies for finding, aligning, and learning ontologies, and ultimately for knowledge discovery. Falcon-AO is one of the prominent components of Falcon. It is an automatic ontology matching component that enables interoperability among Semantic Web applications using related ontologies. It is one of the most practical and popular tools for web ontology matching and mappings that are expressed in RDF(S) and/or OWL. Falcon-AO consists of five main components such as; 1) *Repository,* 2) *Model Pool,* 3) *Alignment Set,* 4) *Matcher Library,* and 5) *Central Controller,* all these components collectively performs the tasks submitted to Falcon-AO.

Prompt [17] is an ontology merging, difference, and alignment tool with a sophisticated scheme of matching terms. Prompt is an open source system developed in Java. It handles ontologies expressed in OWL and RDFS. It takes two ontologies as input and produces the alignments/mappings between them. It is available as a Protégé plug-in. To use the source in user's own application, additional modifications are required on user side.

H-Match [2] is an ontology matching system that takes ontologies as input and produces the results as associations among related ontologies. The results are then used for mappings among these ontologies. H-Match is capable to dynamically configure itself for its adaptation to the semantic complexity of the ontologies to be matched, where the number and type of ontology features are not known in advance. H-Match enforces these dynamic adoption capabilities with the help of syntactic and semantic techniques for ontology matching. In addition it also incorporates a set of four matching models, i.e., *surface, shallow, deep, and intensive*. For its application, H-Match is used for knowledge discovery in the framework of the *Helios* peer-based system.

Lily [22] is a combination of textual and structural techniques for finding out alignments. Lily builds semantic descriptions for each entity of the ontology and then uses lexical similarity and similarity flooding on ontology structure. Its uniqueness is that of using web search engines to overcome semantic heterogeneity. Post processing is conducted for the reason to remove inconsistencies and to make the results more accurate. The framework FOAM [7] is based on heuristically calculated similarity of each resource available in the ontology. Its focus is on the efficiency of alignment generated that also distinguish it from the other systems. Like most of the other systems, FOAM also uses structure of ontology to find out relatedness among entities from participating ontologies.

MAFRA [15], an Ontology MApping FRAmework, mainly developed for distributed ontologies in the Semantic Web. MAFRA provides a conceptual framework with a generic view over the complete distributed mapping process among distributed ontologies. Due to the decentralized nature of Semantic Web there is a significant amount of information redundancy and consistent evolution of ontologies to accommodate the domain knowledge. Thus this changing nature of ontology also needs reestablishment of mapping among the ontologies and MAFRA is considering this issue for the future version.

Falcon-AO [10] and Lily [22] are the most efficient and used tools for ontology matching and mapping with relatively better accuracy. Also when the alignment is to be constructed completely from the scratch, their results accuracy is better than the other existing algorithms [2, 15, and 17]. But the problem like every other system, Falcon and Lily also take much time in establishment of alignments, and they have no support for the reconciliation process of mappings (i.e., unreliable mappings). All the above discussed systems reinitiate the process of mapping between ontologies after they are being updated. This consume lots of time as the changes are usually very simple in type and less in numbers [8, and 9]. In MAFRA [15] and Lily [22], they do mentioned mapping evolution as their future concern but they haven't mentioned

the way of overcoming this issue. Till now, all the existing systems use the complete reestablishment of mappings which is time and resource consuming job.

III. Reconciliation of Ontology Mappings

The proposed scheme for reconciliation of mapping in dynamic/evolving ontologies is time efficient and eliminates staleness from the mappings. It is based on the concept of Change History Log (CHL) [11] that contains all the ontology changes that happens to ontology during ontology evolution. The change log is required for the reason to know which of the mappings are no more reliable due to changes and which resources need re-alignments. For this reason the changes in dynamic ontology need to be maintained for their later use [12]. Basically our proposed scheme (for architecture see Figure 3) has two main components; 1) Change History Log to maintain all the ontology changes and 2) Reconciliation of Mappings in dynamic ontologies.

A. Change History Log

A number of changes, ranging from concepts to properties, can affect the ontology. These changes need to be represented properly to correctly handle explicit and implicit change requirements. For that matter, we have developed Change History Ontology (CHO) [11] to log ontology change, reason for change, and change agents. CHL helps to keep track of the ontology change history.

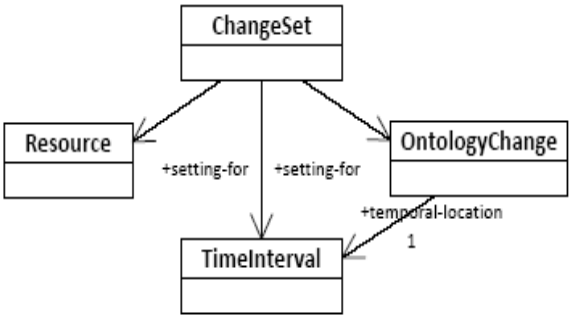

Figure 1, Reification of time-indexed participation of ontology changed resource. *ChangeSet* is a setting for change of a time interval [11].

The core elements of CHO are the *OntologyChange* and *ChangeSet* classes. The *OntologyChange* class has sub-class as *AtomicChange* that represent all the class, property, and individual level changes at atomic level expressed in Figure 1. On the other hand the *ChangeSet* bundles all the changes of specific time interval in a coherent manner shown in Figure 1. The *ChangeSet* is responsible for managing all the ontology changes and arranges them in time indexed fashion. This time indexing also classifies the *ChangeSet* as Instant type and Interval type. Instant type *ChangeSet* holds only one change occurred at some time instant, while Interval type *ChangeSet* holds the changes occurred in a starched time interval [11].

Corresponding to the CRUD interfaces in databases (excluding read), there are three categories in the proposed ontology representing the operations or the change types: Create (such as *ClassAddition, PropertyAddition,* and *IndividualAddition*), Update (such as *ClassRenaming, PropertyRenaming,* and *IndividualRenaming*), and Delete (such as *ClassDeletion, PropertyDeletion,* and *IndividualDeletion*). There are three categories in the ontology to represent different components of the ontology being subject to change. These categories include: *ClassChange*, *PropertyChange,* and *IndividualChange* [8 and 11]. Based on these categories we derive instances of class *OntologyChange*, represented with the symbol Δ, using the following axioms:

$$R_{\Delta} \equiv \exists\ \text{ChangeTarget}.(\text{Class} \sqcup \text{Property} \sqcup \text{Individual} \sqcup \text{Ontology})$$

$$\Delta \equiv R_{\Delta} \sqcap \forall \text{changeType}.(\text{Create} \sqcup \text{Update} \sqcup \text{Delete}) \sqcap \exists \text{changeAgent}.(\text{Person} \sqcup \text{SoftwareAgent}) \sqcap =1\,\text{changeReason}$$

For instance, the following snippet (i.e., Figure 2) represents the class addition scenario including the corresponding *ChangeSet* instance information as well.

```
log:Interval
a                                   cho:ChangeSetType ;
cho:hasChangeSetTypeValue           "Interval" .

log:ChangePerson_Instance_1982
a                                   cho:ChangePerson ;
cho:hasAuthorName                   "Administrator" .

log:ChangeSet_Instance_2474557
a                                   cho:ChangeSet ;
cho:hasChangeAuthor                 log:ChangePerson_Instance_1982 ;
cho:hasChangeSetType                log:Interval ;
cho:hasChangeBeginTime              00:00:46 ;
cho:hasChangeEndTime                00:03:21 ;
cho:hasChangeReason                 "User Request" ;
cho:hasOntology                      http://www.uclab.khu.ac.kr/human.owl .

log:ClassAddition_Instance_1224702057078
a                                   cho:ClassAddition ;
cho:hasChangedTarget                human:NCI_C12801 ;
cho:hasTimeStamp                    1224702057078 ;
cho:isPartOf                        log:ChangeSet_Instance_2474557 ;
cho:isSubClassOf                    owl:Thing .
```

Figure 2, Changes in Human (nci_anatomy) ontology stored in CHL

In Figure 2, *log* is prefix for Change History Log, *cho* is prefix for Change History Ontology, and *human* is prefix for *Human* (*nci_anatomy*) ontology. The above snippet depicts an instance of *ClassAddition* class which is defined as a sub-class of *ClassChange* as elaborated below.

$$\text{ClassAddition} \sqsubseteq \text{ClassChange} \sqcap \exists \text{changeType.Create}$$

With reference to relational databases, our methodology recon on logging techniques to persistently store the changes. This later on helps undo/redo, ontology recovering, query reformulation, temporal traceability of ontology changes, and reconciliation of ontology mappings. The changes are preserved in time-indexed manner in a triple store using the schema provided by CHO [11]. Upon a request for any of the above mentioned purpose, this Change History Log (CHL) containing all the changes is accessed for the required changes. Each entry in the log is an instance of either *ChangeSet* or *OntologyChange* class from the CHO. The log also preserves the provenance information about the change, such as who made these changes and when; and also what was the reason for these changes.

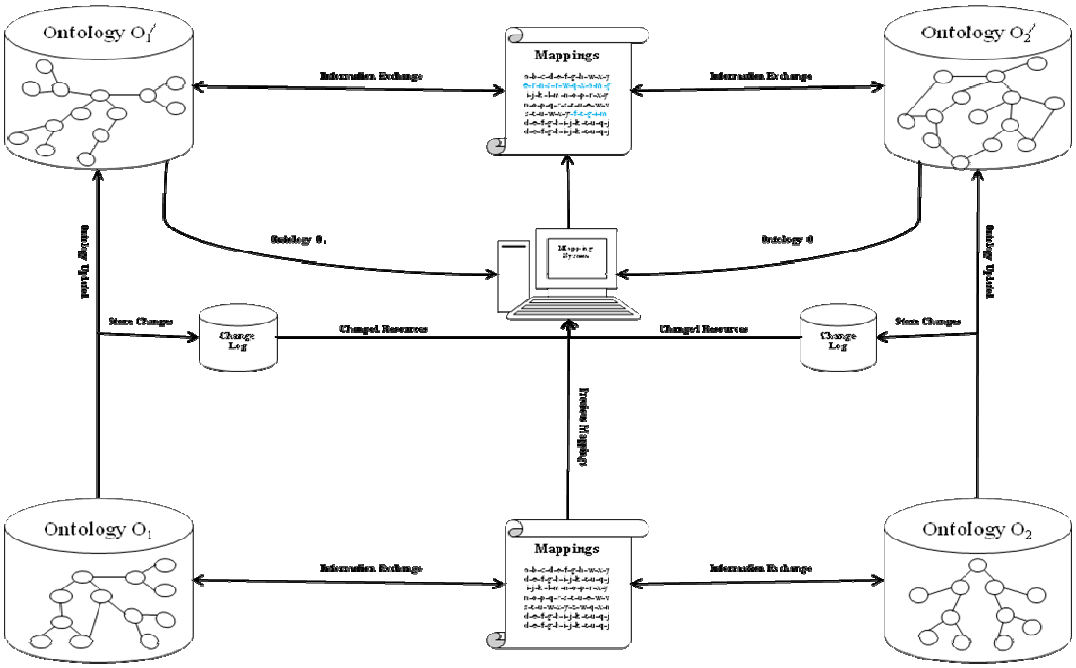

Figure 3, Shows the framework for reconciliation of mappings in dynamic/evolving ontologies.

Now a days, the use of ontology is not restricted to only local use, but they may be centralized or even distributed among different remote nodes, so the same goes for CHL. CHL can also be used in all the three contexts (i.e., local CHL, centralized CHL, and distributed CHL). To manage CHL consistency in these different situations, we have also provided the provision to import OWL-Time ontology in CHO to eliminate any possible type of change ordering conflicts.

B. Reconciliation Procedure

As discussed above, there are different algorithms available to establish mappings between ontologies [2, 10, 15, 17, and 22]. Our idea is to use the entries of Change History Log [11] for reconciliation of mappings between ontologies, which not only help to eliminate staled mappings but also take less time to reconcile mappings. This approach is most suitable for large size ontologies having hundreds and thousands of resources, for example; when reconciling mappings among *Mrinkman, GTT, GEMET, NALT, Google Classification, Wiki Classification*, *ACM Classification Hierarchy,* and *MSC Classification Hierarchy*. The larger the size of ontology the better and time efficient the approach is against any of the above discussed algorithms. Detail procedure is given below.

Re-establishing Mappings*:* Looking at the scenario given in Figure 3, where two ontologies are mapped and they exchange information based on the established mappings. Now consider that one or both the ontologies are changed (evolved) to another state (see Figure 3). In this case the already existing mappings are of no more use as they are not reliable and also became stale in this situation. So the mappings between these two ontologies need to evolve with the evolving ontologies to be up to date. To discuss this scenario we take two different cases.

1. If one of the ontologies evolve from one state to another then its mapping with other ontologies are not reliable as there will be changes in the resources mapped with the other ontology. To reconcile the mappings between these ontologies, we propose that instead of completely reinitializing the mapping process from the scratch, which is a time consuming process, use the CHL entries to figure out the changed resources in the evolved ontology. Then use only these changed resources in the mapping process to map it with the other ontology and simply update the previous mappings with the new ones and remove the staled mapping entries. In this case we only need to extend the method for calculating the Semantic Affinity (*SA*) by incorporating the change information from CHL. So the modified method for *SA* including parameters is given below;

$$SA(C, C', \Delta', \psi)$$

- C Resource from Ontology O_1
- C' Resource from changed Ontology O_2
- Δ' Change information from CHL of Ontology O_2
- Ψ User defined threshold for resource match

2. Now consider that both the ontologies evolved from one consistent state to another as demonstrated in Figure 3 and it is also the worst case scenario. In this case the mapping also needs to evolve to accommodate the mappings for the new changed resources and eliminate the staleness from the already established mappings. Again, we don't need to completely reestablish the mappings between both the ontologies like existing

systems which is a time and resource consuming process. As given in Figure 3, both ontologies O_1 and O_2 have evolved. To reconcile the mappings between the evolved ontologies in time efficient manner and remove the staled mappings, we use the CHL entries for both the ontologies to identify the changed resources from both ontologies. Based on the identified changes, reconcile mappings for these changed resources, update the old mappings, and remove the unreliable (staled) mappings from the previous mappings. This is not only time efficient technique but also eliminate the staleness from the mappings that needs to be updated for reliable communication and exchange of information between systems and/or services.

The inputs for this module are; (also shown in Figure 3) both the evolved ontologies O_1 and O_2, CHL entries for both the ontologies Δ_1 and Δ_2 for ontology O_1 and for Ontology O_2 respectively. The pervious mappings between these two ontologies are updated at the end of proposed algorithm (see Algorithm-1) execution. *SA* is calculated by incorporating the change information from CHL. So the modified method including parameters will be like;

$SA(C_1, \Delta_1, C_2, \Delta_2, \psi)$ {
C_1 *Resource from ontology* O_1
Δ_1 *Change information from CHL of Ontology* O_1
C_2 *Resource from Ontology* O_2
Δ_2 *Change information from CHL of Ontology* O_2
ψ *User defined threshold for resource match*

Δ_1 and Δ_2 are changes of both the ontology contained in CHL. For calculating *SA*, these changes are required and extracted from CHL using SPARQL query given below. To get the latest changes, first the *ChangeSet* instances are sorted in descending order of their timestamp defined in CHO and the top most *ChangeSet* instance is selected. Afterwards all the changes corresponding to the selected *ChangeSet* instance are retrieved from CHL.

```
Resource: ← SELECT ?changes ?timeStamp WHERE {?changes
docLog:isPartOf changeSetInstance . ?changes docLog:hasTimeStamp
?timeStamp } ORDER BY DESC(?timeStamp)
```

```
Δx: ← SELECT ?changedTarget ?isSubClassOf WHERE {Resource
docLog:hasChangedTarget ?changedTarget . Resource
docLog:isSubClassOf ?isSubClassOf }
```

After the process of reconciliation, the staled part of mapping is removed from the overall mappings. It is updated with the new changed mappings as shown in Figure 3 with color blue in the mappings. This process of reconciliation of mapping not only eliminates the staleness from the mappings but also is more time efficient as it just focuses on changed resource that makes it more suitable for systems and services dealing in information exchange.

IV. IMPLEMENTATION AND RESULTS

In this section we discuss in detail about the results we have achieved with our proposed extensions to the existing mapping systems[3] i.e., Falcon [10], FOAM [7], Lily [22], and Prompt [17]. The results verify that the amount of time required for reconciliation of mapping using proposed extensions is far less than the existing systems. The data sets used in these experiments are; *Mouse, and Human* ontologies, available online at (http://oaei.ontologymatching.org/). Other data sets such as *Health* and *Food*[4] ontology and *People+Pets*[5] ontology are also used.

The experiments are all conducted on machine with 2.66 GH Quad Core and 3 GB of primary memory. The experiments are carried out for 2nd case (worst case) i.e., both the mapped ontologies evolve from one state to another. These experiments are by no means the comparison of existing systems, but the comparison of each individual system with our proposed extensions to that individual system. The experiments are conducted on changes at atomic level, while changes can occur in two modes i.e., changes are considered at

Algorithm ReconMapping ( ):
A resource matching threshold is defined as $\psi = 0.70$.
Input: Ontologies O_1 and O_2 for mapping reconciliation.
Input: Ontology change information (i.e., Δ_1 and Δ_2) from CHL of both ontologies, i.e., $\Delta_1 \in O_1$ and $\Delta_2 \in O_2$.
Output: Set of mappings for the changed resources and then updated in the original mappings file..

1. /* Check for change of resources in CHL of both the mapped ontologies and read the changes in Δ */
2. **If** $\exists \Delta \sqcap \Delta.O_1.CHL.NewChange$ **then**
3. /* Read the changes in Δ_1 */
4. $\Delta_1 \leftarrow$ {x | < CHL_Δ, x > Change}
5. **Endif**
6. **If** $\exists \Delta \sqcap \Delta.O_2.CHL.NewChange$ **then**
7. /* Read the changes in Δ_2 */
8. $\Delta_2 \leftarrow$ {x | < CHL_Δ, x > Change}
9. **Endif**
10. /* Start mapping reconciliation procedure by calculating semantic affinity */
11. **If** $\exists \Delta_1.Change \sqcap \exists \Delta_2.Change$ **then**
12. /* Calculate semantic affinity using changed resources of both the ontologies */
13. R-Map [][] ← SemanticAffinity($C_1 \in O_1$, Δ_1, $C_2 \in O_2$, Δ_2, ψ)
14. **Endif**
15. **If** $\exists \Delta_1.Change \sqcup \exists \Delta_2.Change$ **then**
16. /* Calculate semantic affinity using changed resources of one changed ontology */
17. R-Map [][] ← SemanticAffinity($C_1 \in O_1$, $C_2 \in O_2$, Δ_2, ψ)
18. **Endif**
19. /* Update the original mapping file with the reconciled mappings for the changed resources */
20. Execute.update(Mappings, R-Map[][])
21. **End**

Algorithm-1. Time efficient reconciliation algorithm for ontology mapping using ontology changes

[3] The mapping systems selected for use and comparison in this paper are those which showed best performance in OAEI'05, OAEI'07, and OAEI'09.
[4] http://aims.fao.org/
[5] http://www.atl.lmco.com/projects/ontology/

complex and at atomic level [12]. Complex change is a change that consists of several atomic level changes e.g., deletion of super class will result in complex change that includes the deletion of all the subclasses of that super class. Atomic change is a simple change e.g., renaming a resource. In these experiments, changes are mostly the introduction of new resources in the domain ontologies. Figure 4 is the validation of our claim and it uncovers the limitations of existing systems by not focusing on mapping evolution and its effects. The existing systems take almost same amount of time or even more for regeneration of mappings. Figure 4 shows the experimental results of FOAM [7], Falcon [10], Lily [22], and Prompt [17] on four different versions of *Human* and *Mouse* ontologies. The number of changes between different versions of these ontologies is listed in Table 1. The reason is that the mapping and mapping re-establishment procedure in existing systems always starts from the scratch.

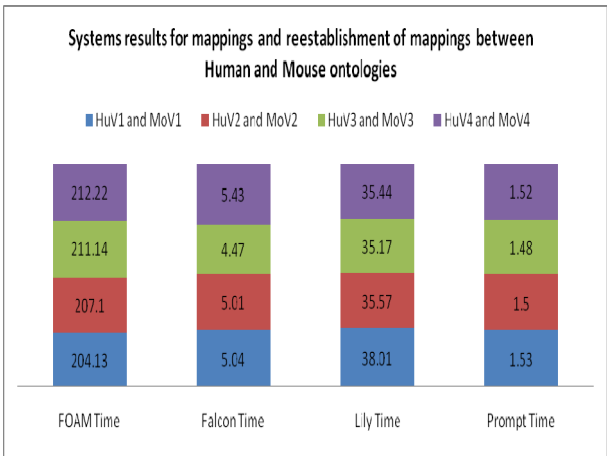

Figure 4, Mapping and Reestablishment of mapping results with respect to time for Mouse and Human ontology using FOAM [7], Falcon [10], Lily [22], and Prompt [17].

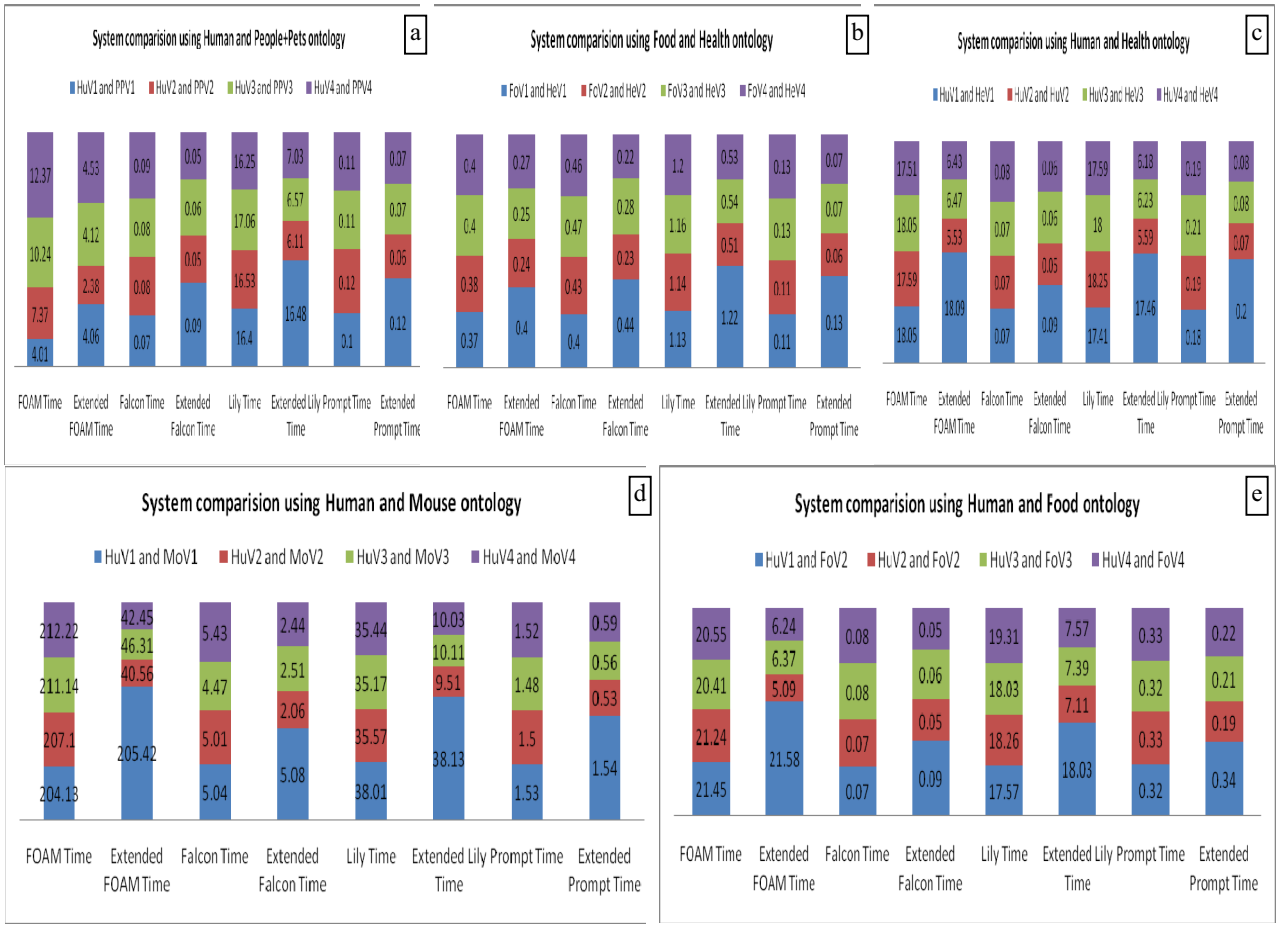

Figure 5, Detail comparisons of proposed extensions against Falcon [10], FOAM [7], Lily [22], and Prompt [17] on combination of 5 different data sets are given. Each graph i.e., a, b, c, d, e, shows the existing systems results in comparison to our proposed extensions. X-axis is the existing systems and proposed extensions tests on data sets. Y-axis of the graphs in-general shows the time. Each packet of every bar in the graphs with different colors show the execution time consumed by existing systems and proposed extensions for reconciliation of mappings between various versions of ontology. In these graphs; *Hu=Human, Mo=Mouse, Fo=Food, He=Health,* and *PP=People+Pets* are used as shortcut for ontology names where *V* represent the version with number on it e.g., *HuV2* represent *Human* ontology and its 2nd version.

To test the existing systems and with our proposed extensions, we used standard data sets and their changes at atomic level. Table 1 shows different versions of data sets and the number of atomic level changes between their different versions. Then these versions and the mentioned changes from CHL are used for experiments using existing systems and the proposed extensions in the existing systems.

TABLE 1. Ontology versions and the number of atomic changes applied to one version that transforms ontology to another version. All the ontologies listed in 1st row. Numeric values are the number of changes between two different versions of ontology.

Ontology Versions	Human	Mouse	Health	Food	People+Pet
Version1	Original	Original	Original	Original	Original
Version2 = Version1 + No of Changes	283	166	169	122	120
Version3 = Version2 + No of Changes	112	201	153	161	172
Version4 = Version3 + No of Changes	123	198	145	114	109

The existing systems i.e., Falcon [10], FOAM [7], Lily [22], and Prompt [17] and proposed extensions are tested on the data sets shown in Table 1. A constant similarity value of 0.70 is kept in all the tests as a matching threshold. Numbers of iterations in most of the systems are kept as default but in case of FOAM [7] it is set to 7 iterations per execution, however, it does not affect the results as systems are not compared with one another for accuracy. The execution time of these systems varies against one another (see Figure 5) due to different matching scheme used in their implementation. These algorithms start from scratch so mostly they take more time than the previous test as shown in Figure 5. Our extension to existing algorithms using Change History Log (CHL) [11], only consider the changed resources and reconcile mappings for the changed resources. Proposed technique helped in saving large amount of the computation time (shown in Figure 5). Execution time shown in Figure 5 is all in minutes and fractions of minutes.

Each graph of Figure 5 shows the results of existing systems and proposed extensions on a particular data set with its different versions. Each graph of Figure 5 consists of 4 pairs, making 8 bars in total. Each alternative pair is the results comparison of proposed system against existing system. 1st bar of each pair shows the execution time of existing system on each version (differentiated using colors) of ontology, while the 2nd bar of each pair shows the execution time for our proposed extensions. One very obvious pattern visible in each graph of Figure 5 is that the execution time of proposed extensions on starting versions of ontologies is always same or a fraction greater than the existing systems. It is because, if the ontologies are matched for the first time, in this case, proposed system carryout complete mapping procedure in addition to looking for the changes from CHL and existing mappings. The detailed experimental results shown in Figure 5 validate that our extensions drastically reduce the time required for reconciling ontology mappings. This facilitates the process of interoperability and information exchange between web services and the services are not suspended for longer durations.

The time for reconciliation of mappings between ontologies depends on the types of changes made. A single change introduced may have cascading effects on existing resources or may result in several induced changes [8]. Our approach depends on number of changes, the more the number of changes in an ontology higher will be the mapping time for our approach but still less time than the original algorithms. Mostly, the cascade effects and induced changes are due to the change at higher level of hierarchy and are less frequent once domain ontology gets matured [9 and 8]. One of such case is also visible in Figure 6 (x-axis = number of tests, y-axis = minutes) in bar third comparison. First bar is the original time for existing systems to establish the mappings between *Human* and *Mouse* ontology. The remaining bars are the time results for the reconciliation of mapping with our proposed extensions using CHL. In this case, we used the same sets of versions used for initial experimental results and with the same number of changes given in Table 1. Even with the cascade effects and induced changes, our proposed approach takes less mapping computation time than the original algorithms.

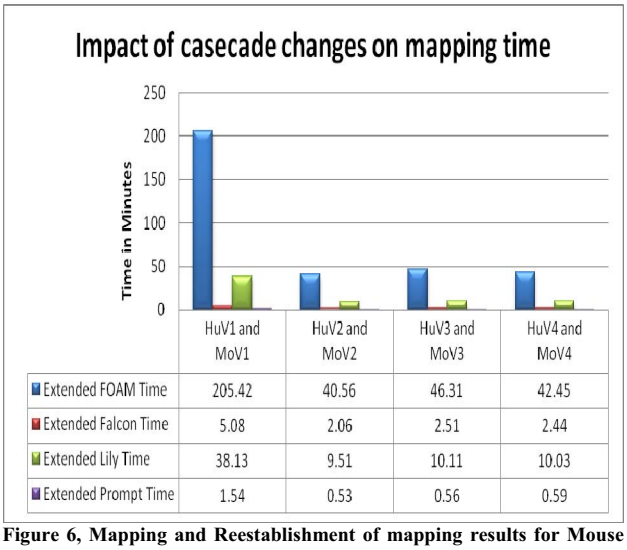

Figure 6, Mapping and Reestablishment of mapping results for Mouse and Human ontology. First bars combination and first column in the table is the result of original FOAM [7], Falcon [10], Lily [22], and Prompt [17]. The remaining bars combinations are the results of our proposed extension. 3rd combination of bars and 3rd column show the time increase due to cascading effects of changes.

V. Conclusions and Future Work

Information exchange and interoperability is the key research issue in different research groups. Mapping between two information sources (i.e., ontologies) of web services is the key of sharing information and achieving interoperability. Systems exist that generate mappings between ontologies to support information exchange and interoperability. Due to autonomous nature of services and discovery of new

knowledge in the field, that makes the domain ontologies to evolve which consequently make the existing mappings unreliable. In this situation, mapping reconciliation is required to keep the services functioning for information exchange. We proposed extensions to existing systems by introducing CHL that enable the reconciliation of mappings in these systems. We found a drastic decrease in amount of time required for reconciling ontology mappings among dynamic ontologies, compared to the already existing systems that reinitiate the complete process.

Currently, we are testing our technique on larger data sets in different combinations to verify our claims. Variable mapping accuracy in results of our technique is also one of the concerns. We are also planning to use CHL for change prediction and ontology consistency after change.

ACKNOWLEDGMENT

This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency)" (NIPA-2011-(C1090-1121-0003) and by Basic Science Research Program Through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (2010-0016042).

REFERENCES

[1] V. Bicer, O. Kilic, A. Dogac, G. B. Laleci, “Archetype-based semantic interoperability of web service messages in the health care domain”, International Journal of Semantic Web and Information Systems (IJSWIS), 1(4), pages 1–23, October 2005.

[2] S. Castano, A. Ferrara, and S. Montanelli. “Matching ontologies in open networked systems”. Techniques and applications, Journal on Data Semantics (JoDS), vol. V, pp. 25-63, 2006.

[3] N. Choi, I. Song, and H. Han, “A survey on ontology mapping”, SIGMOD Rec. Vol. 35, No. 3. Pages 34 – 41, 2006.

[4] F. Chen,Z. Zhang, J. Li, J. Kang, and H. Yang, “Service Identification via Ontology Mapping”, In Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference, pages 486--491, USA, 2009.

[5] S. Dietze, N. Benn, J. Domingue, A. Conconi, and F. Cattaneo, “Two-Fold Service Matchmaking - Applying Ontology Mapping for Semantic Web Service Discovery”, In Proceedings of Asian Semantic Web Conference, pages 246-260, China, 2009.

[6] S. Dietze, A. Gugliotta, J. Domingue, H. Q. Yu, and, M. Mrissa, “An automated approach to Semantic Web Services Mediation” Journal on Service Oriented Computing and Applications, pages 261-275, volume (4), issue (4), 2010.

[7] M. Ehrig and Y. Sure. “Foam - framework for ontology alignment and mapping; results of the ontology alignment initiative”, In Proc. of the Workshop on Integrating Ontologies, 2005.

[8] G. Flouris, D. Manakanatas, H. Kondylakis, D. Plexousakis, and G. Antoniou. “Ontology Change: Classification and Survey,” Knowledge Engineering Review (KER), 23(2), pages 117-152, 2008.

[9] A. Y. Halevy, Z. G. Ives, M. Jayant, P. Mork, D. Suciu, and I. Tatarinov, “The Piazza peer data management system”, IEEE Transactions on Knowledge and Data Engineering, volume (16), pages 787 - 798, July 2004.

[10] W. Hu, and Y. Qu, “Falcon-AO: A practical ontology matching system.” Journal of Web Semantics. 6, 3, pages 237-239, 2008.

[11] A. M. Khattak, K. Latif, S. Khan, N. Ahmed, "Managing Change History in Web Ontologies," Semantics, Knowledge and Grid, International Conference on, pp. 347-350, 2008.

[12] A. M. Khattak, K. Latif, S. Y. Lee, Y. K. Lee, “Ontology Evolution: A Survey and Future Challenges”, The 2nd International Conference on u- and e- Service, Science and Technology (UNESST 09), Korea, 2009.

[13] T. B. Lee, J. Hendler, and O. Lassila, “The Semantic Web: a new form of web content that is meaningful to computers will unleash a new revolution of possibilities”, Sci. Am 5(1), 2001.

[14] J. Madhavan, P. Bernstein and E. Rahm, “Generic Schema Matching with Cupid”. 27th International Conference on Very Large Data Bases, 2001.

[15] A. Maedche, B. Motik, N. Silva, and R. Volz, “MAFRA - A MApping FRAmework for Distributed Ontologies.” In Proceedings of the 13th international Conference on Knowledge Engineering and Knowledge Management. ontologies and the Semantic Web, pages 235-250, London, 2002.

[16] M. Nagarajan, K. Verma, A. P. Sheth, J. Miller, J. Lathem, "Semantic Interoperability of Web Services: Challenges and Experiences," Proc. 4th IEEE International Conference on Web Services, IEEE CS Press, pages 373–382, 2006.

[17] N. Noy and M. Musen. “The PROMPT Suite: Interactive tools for ontology merging and mapping”, International Journal of Human-Computer Studies, (59(6)): pages 983–1024, 2003.

[18] M. Paolucci, N. Srinivasan, K. Sycara, “Expressing WSMO Mediators in OWL-S”, In proceedings of Semantic Web Services (ISWC), 2004.

[19] P. Shvaiko and J. Euzenat, “ Ten Challenges for Ontology Matching”, In proceedings of the 7th International Conference on Ontologies, Darabases, and Applications of Semantics (ODBASE), 2008.

[20] A. P. Sheth and J. A. Larson, “Federated database systems for managing distributed, heterogeneous, and autonomous databases”, ACM Computing Surveys, 1990.

[21] P.S. Tan, A.E.S. Goh, and S.S.G. Lee, “An Ontology to Support Context-Aware B2B Services”, IEEE International Conference on Services Computing (SCC), pages 586-593, USA, 2010.

[22] P. Wang and B. Xu, “Lily: Ontology alignment results for oaei 2009”. Ontology Matching (OM), 2009.