Knowledge-Based Systems 35 (2012) 369–374

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Short Communication

Time efficient reconciliation of mappings in dynamic web ontologies

Asad Masood Khattak, Zeeshan Pervez, Khalid Latif, Sungyoung Lee *

Department of Computer Engineering, Kyung Hee University, South Korea
School of Electrical Engineering and Computer Science (SEECS), NUST, Pakistan

ARTICLE INFO

Article history:
Received 25 February 2011
Received in revised form 14 April 2012
Accepted 15 April 2012
Available online 21 April 2012

Keywords:
Ontology change
Ontology evolution
Ontology mapping
Mapping evolution
Mapping reconciliation

ABSTRACT

Mappings are established among ontologies for resolving the terminological and conceptual incompatibilities among information networks and information systems. Accommodating new knowledge in domain ontology causes the ontology to change from one consistent state to another. This consequently makes existing mappings among ontologies unreliable and stale due to the changes in resources. Mapping evolution eliminates discrepancies in the existing mappings. The proposed approach offers the benefits of re-establishing mappings among the updated ontologies in less time than is required with existing systems. It only considers the changed resources and eliminates staleness from the mappings. This approach uses the change history to drastically reduce the time required for reconciling mappings among ontologies, as shown in the results.

1. Introduction

The amount of information on the web is increasing rapidly, placing a heavy computational load on systems for accessing, interpreting, manipulating, maintaining, merging, integrating, inferring, and mining the information [1]. Ontology is *a formal, explicit specification of a shared conceptualization*, and use of ontology in systems dealing with information extraction from large complex structured information can provide valuable results [2]. The increasing use of ontology in information systems also increases the significance of ontology maintenance [3].

The increase in the number of information sources also raises the importance of sophisticated information extraction and management [1,4]. Ontology mapping is a relatively mature area of research used for aligning two or more ontologies (information sources) for the purpose of sharing information and overcoming heterogeneity issues [1,2,4–8]. The terms mapping and matching are often used interchangeably. However, matching is considered to be a prerequisite of mapping [1]. Matching is the process of determining semantic relatedness between two entities. On the other hand, mapping is the process of finding the data transformation based on the semantic relatedness for a given instance of a source entity that will produce an instance of a target entity [8,9]. Falcon [2], MAFRA [5], H-Match [6], Lily [10], and TaxoMap [11] are amongst the best matching and mapping systems. These systems require substantial computational resources to consider large ontologies, such as *Google Classification*,[1] *Wiki Classification*,[2] *ACM Classification Hierarchy*,[3] and *MSC Classification Hierarchy*[4] for mapping. On the other hand, when one or both of the mapped ontologies change from one state to another, then the existing mappings are no longer reliable. There is a need for a system that supports mapping evolution. Existing systems, i.e., Falcon [2], MAFRA [5],H-Match [6], Lily [10], and TaxoMap [11], do not support mapping evolution but re-start the entire process for re-establishing mappings, which is a time-consuming process.

Re-establishment of mappings is required for dynamic mapped ontologies. Re-establishing mappings takes more time than the original mapping process due to the changes introduced. However, the changes in mapped ontologies and regenerated mediation are not significant [12]. We propose a less time-consuming scheme for reconciling ontology mappings (mapping evolution) in dynamic/evolving ontologies. Our approach uses the Change History Log (CHL) [13] method for mapping reconciliation. The use of CHL in ontology mapping helps to reconcile mappings in dynamic/evolving web ontologies in order to overcome the staleness problem and reduce the time for reconciliation of mappings. During reconciliation of ontology mapping, only the out-dated

* Corresponding author. Address: Room No. 313, Department of Computer Engineering, Kyung Hee University, 446-701 Yong-in, South Korea. Tel.: +82 31 201 2514; fax: +82 31 202 2520.
E-mail address: sylee@oslab.khu.ac.kr (S. Lee).

[1] http://www.google.com/Top/Reference/Libraries/Library_and_Information_Science/Technical_Services/Cataloguing/Classification/.
[2] http://en.wikipedia.org/wiki/Taxonomic_classification.
[3] http://www.acm.org/about/class/1998/.
[4] http://www.math.niu.edu/~rusin/known-math/index/index.html.

0950-7051/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2012.04.016

mappings are updated which saves both time and resources. We have tested Falcon [2], H-Match [5], Lily [10], and TaxoMap [11] algorithms on eight different data sets available online and then extended these algorithms with the proposed scheme by incorporating the CHL. The experimental results show a drastic decrease in the computational time required for mapping reconciliation using the proposed extensions compared with those of Falcon, H-Match, Lily, and TaxoMap on data sets of *Mouse* and *Human* ontology, *Brinkman* and *GTT* ontology, and *GEMET* and *NALT* ontology.

2. Reconciliation of ontology mapping

The proposed scheme for reconciliation of mapping in dynamic ontologies is efficient in terms of computation time and eliminates staleness from the existing established mappings when one or both of the mapped ontologies evolve. The scheme is based on the concept of CHL [13] which contains changes in ontology during evolution. The change log is required to know which of the mappings are staled due to changes and what resources need realignment. The proposed scheme has two main components: (1) CHL to maintain ontology changes and (2) reconciliation of mappings in dynamic ontologies.

2.1. Change History Ontology

A number of changes, ranging from concepts to properties, can affect the ontology. The changes need to be represented properly to correctly handle explicit and implicit change requirements. To address this, we have developed Change History Ontology (CHO) [13] to log ontology change, reason for change, and change agents that help to keep track of the change history.

The core elements of CHO are the *OntologyChange* and *ChangeSet* classes. The *OntologyChange* class has the sub-class *AtomicChange* that represent all the class and property level changes at the atomic level, as expressed in Fig. 1. On the other hand the *ChangeSet* bundles all the changes in a specific time interval in a coherent manner (see Fig. 1). The rationale is that the individual changes are not performed in isolation and are usually part of a particular change session. 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 or Interval type. Instant type *ChangeSet* contains only one change during a particular time instant, while the Interval type *ChangeSet* contains the changes that occur during a starched time interval [13].

In previous approaches [1,15], ontology changes are stored sequentially without preserving their dependencies or interlinking them with other changes. In CHL, the *ChangeSet* instance is used for grouping the changes to preserve the coherence of all the ontology changes at the atomic level. One ontology resource participates in a change event per time instance. Fig. 1 shows a diagrammatic depiction of this pattern. The listing of all ontology changes is maintained in the CHL in conformance to the CHO.

Corresponding to the CRUD interfaces in databases, the proposed ontology has three categories (excluding read) representing the change types: Create, Update, and Delete. To represent the changes in CHL, there are four categories in the ontology to represent different components of the ontology that are subject to change (i.e., *ClassChange*, *PropertyChange*, *IndividualChange*, and *OntologyChange* [13]). Based on the above mentioned categories, we derive instances of class *OntologyChange*, represented with the symbol Δ, using the following axioms, for details see [3] and [13]:

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

A single change representation with a corresponding *ChangeSet* containing relevant information and changes in *Human* ontology is shown in Fig. 2, where *log* is the prefix for Change History Log, *cho* is the prefix for Change History Ontology, and *human* is the prefix for *Human* (*nci_anatomy*) ontology. The snippet depicts an instance of the *ClassAddition* class which is defined as a sub-class of *ClassChange*.

Each entry in the log is an instance of either *ChangeSet* or *OntologyChange* class from the CHO. Identifying the set of changes resulting from ontology evolution is very important for accurately extracting the required changes from CHL for mapping reconciliation. It is necessary to avoid the risks of mapping reconciliation based on changes which have resulted from older ontologies instead of the current version. The idea of interval type *ChangeSet* is to bundle the changes corresponding to a particular change session and then to use the exact set of changes for mapping reconciliation. Moreover, the time ordering of an individual element level change is also followed for each step of the reconciliation procedure. Collectively, both *ChangeSet* and element level *Ontolo-*

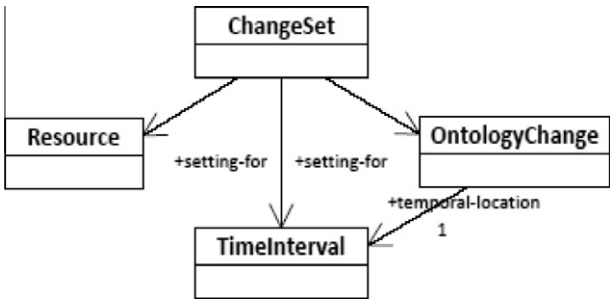

Fig. 1. Reification of time-indexed participation of an ontology changed resource. ChangeSet is a setting for a change in the time interval [13].

```
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
```

Fig. 2. Changes in Human (nci_anatomy) ontology stored in CHL.

gyChange information, resulting from incremental changes or drastic changes in ontology; help to achieve the overall objective of mapping reconciliation.

2.2. Time efficient reconciliation

Systems used in previous research [2,5,6,10,11] establish ontology mappings with high accuracy; however, these are still time-consuming. Our idea is to use the Change History Log [13] entries to reconcile mappings among ontologies in less computational time and to eliminate staleness from established mappings. This approach is most suitable for large ontologies with hundreds and thousands of resources, such as *Google Classification* and *Wiki Classification*, or *ACM Classification Hierarchy* and *MSC Classification Hierarchy*. The larger is the size of the ontology, the better and more time efficient is the approach compared with the existing algorithms.

Consider that two ontologies are mapped and exchange information based on the established mappings, and that one or both of the ontologies are changed (evolved) to another state. In this case, the existing mappings are not reliable and became stale. The mappings between these two ontologies need to evolve with the evolving ontologies to keep mappings between ontologies up to date. The scenario is discussed in two cases: (1) one of the mapped ontologies evolves and (2) both the ontologies evolve from one consistent state to another. In both cases, the mappings also need to evolve to accommodate the new mappings for the changed resources and to eliminate the staleness from the established mappings.

To reconcile the mappings in a time efficient manner and to remove the staled mappings, we propose to use the CHL entries for both ontologies by identifying the changed resources in both ontologies. Mappings are then established only for the changed resources and to update the existing mappings. The previous mappings between these two ontologies are updated at the completion of the proposed algorithm (see Algorithm-1) execution. We need to extend the method by providing appropriate resources for calculating Semantic Affinity (*SA*) using the change information from CHL. The signature for *SA* is given below.

$$\boldsymbol{SA}(\boldsymbol{C}_1, \Delta_1, \boldsymbol{C}_2, \Delta_2, \psi)\begin{cases} C_1 \text{ Resource from Ontology } O_1 \\ \Delta_1 \text{ Change information from CHL of Ontology } O_1 \\ C_2 \text{ Resource from Ontology } O_2 \\ \Delta_2 \text{ Change information from CHL of Ontology } O_2 \\ \psi \text{ User defined threshold for resource match} \end{cases}$$

where Δ_1 and Δ_2 are changes in both ontologies contained in CHL. For calculating semantic affinity, these changes are required and extracted from CHL using the SPARQL query given below. To determine the latest changes, the *ChangeSet* instances are sorted in descending order of their timestamp defined in CHO, and the topmost *ChangeSet* instance is selected. 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)
Δ_x: ← SELECT ?changedTarget ?isSubClassOf WHERE {Resource docLog:hasChangedTarget ?changedTarget. Resource docLog: isSubClassOf ?isSubClassOf}

Algorithm 1.

Time efficient reconciliation algorithm for ontology mapping using ontology changes

Algorithm ReconMapping ():
A resource matching threshold is defined as $\psi = 0.70$.
Input: Ontologies O_1 and O_2 for mapping reconciliation. 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 which is then updated in the original mappings file.
1. /* Check for change of resources in CHL offer both the mapped ontologies and read the changes in Δ from CHL */
2. **If** $\exists \Delta \sqcap \Delta.O_1.CHL.NewChange$ **then**
3. /* Read the changes in ontology O_1 from CHL in Δ_1 */
4. $\Delta_1 \leftarrow \{x| < \text{CHL}_\Delta, x > \text{Change}\}$
5. **Endif**
6. **If** $\exists \Delta \sqcap \Delta.O_2.CHL.NewChange$ **then**
7. /* Read the changes in ontology O_2 from CHL in Δ_2 */
8. $\Delta_2 \leftarrow \{x| < \text{CHL}_\Delta, x > \text{Change}\}$
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 from CHL of both the changed ontologies */
13. R-Map [][]← SemanticAffinity($C_1 \in O_1$, Δ_1, $C_2 \in O_2$, Δ_2, ψ)
14. **Else–If** $\exists \Delta_1.Change \sqcup \exists \Delta_2.Change$ **then**
15. /* Calculate semantic affinity using changed resources of one of the changed ontologies represented as Δ^l */
16. R-Map [][]← SemanticAffinity ($C_1 \in O_1$, $C_2 \in O_2$, Δ^l, ψ)
17. **Endif**
18. /* Update the original mapping file with the reconciled mappings for the changed resources, this step also removes the stale mappings */
19. Execute.update(MappingsFile, R-Map[][])
20. **End**

3. Discussion

Systems tested in previous studies [5,10,14,15] support mapping evolution. However, some of these have different focuses while the others are not mature enough in their approaches. The system discussed in [14] mainly focuses on schema-based mapping evolution to support Local as View and Global as View approaches [16] that support query reformulation in data integration applications. Our proposed approach is different from [14], as schema and ontologies are fundamentally different [15,17,18]. In [15], the authors proposed a mapping evolution algorithm for mappings between a schema and the schema's annotations. The focus of the algorithm is to maintain the consistency of mapping between the schemas and their corresponding annotations. Both the systems discussed in [14] and [15] are different from our proposed system as [14] focuses on schema level mapping evolution and [15] focus on mapping evolution between the schema and annotations for the schema. System discussed in [18] proposed s-XML, a mapping scheme to bridge XML and relational database for effective storage, extraction, and exchange of large data.

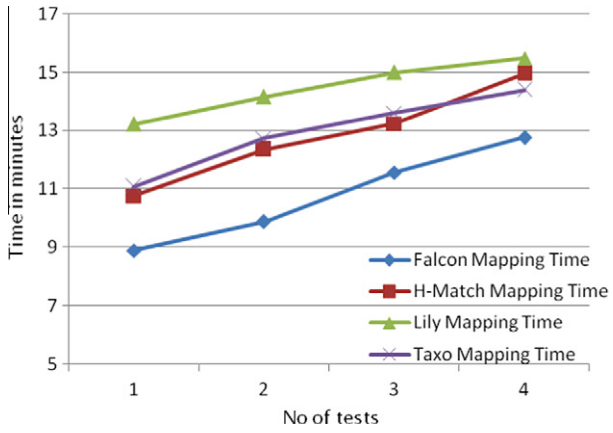

Fig. 3. Falcon [2], H-Match [6], Lily [10], and TaxoMap [11] mapping and re-establishment of mapping results with respect to time for Mouse and Human ontologies.

MAFRA [5] and Lily [10] are the two mapping systems that in addition to mapping generation between two ontology versions also focus on the evolution of mappings when at least one of the mapped ontologies evolves from one state to another. However, neither MAFRA [5] nor Lily [10] has a concrete methodology to support mapping evolution for evolving ontologies. For the testing and discussion of the proposed system, the authors made extensions to the existing systems in order to support the mapping reconciliation procedure instead of redeveloping the whole mapping system.

The discussion in this section is based on the mapping procedure times of Falcon [2], H-Match [6], Lily [10], and TaxoMap [11] and then on a time comparison with our extensions using CHL. The data sets used in these experiments are *Mouse, Human, Mrinkman, GTT, GEMET, and NALT* ontologies, available online at http://oaei.ontologymatching.org/. The changes are considered at both complex and atomic levels [1,4]. Complex change consists of several atomic level changes, e.g., deletion of a super class will result in complex change that consists of the deletion of all the subclasses of that super class. Atomic change is a simple change, e.g., renaming a resource. These experiments are all conducted using complex changes.

The goal of our study is to uncover the limitations of existing systems that do not focus on mapping evolution and its effects. In Fig. 3, for every 25 new changes, the time for re-establishment of mapping (using existing systems) increases. In existing systems, the mapping procedure is restarted between ontologies with 25 new changes (most of the changes are addition of new resources). Table 1 shows the time consumed for establishing mappings among different ontologies of Falcon, H-Match, Lily, and TaxoMap and also with our extensions to Falcon, H-Match, Lily, and TaxoMap. The experiments are conducted on a computer with 2.66 GHz Quad Core and 4 GB of primary memory. As visible from the first four rows of Table 1 for all mappings, all the systems require more than 8.50 min. In some cases, such as *Food* ontology, Falcon needs 5.75 h to generate mappings (refer to [2]).

Each time 25 random changes occur (mostly new resource additions) in each of the mapped pair ontologies listed in Table 1, evolution to another state occurs. We reapplied the algorithms to re-establish mappings and then implemented them with our extensions. Existing algorithms start from scratch and take more time than the previous test, as shown in the second group of four row combinations in Table 1. Our extensions to existing algorithms using Change History Log (CHL) [13] only consider the changed resources and reconcile mappings. The proposed technique saves significant computation time (shown in the last group of four rows of Table 1). In the second set of experiments, the proposed extensions are then tested with 25 random changes introduced into both mapped ontologies. Table 2 shows the mapping times of the original systems and the mapping times of proposed extensions to the existing system. Even with the changes in both the mapped ontologies, the proposed system result for mapping reconciliation is far better than those of the existing systems. The results show that our extensions using CHL drastically reduced the computation time required for reconciliation of mappings in dynamic ontologies. Due to its smaller size and association with the evolving ontology, the

Table 1
Computation time analysis of Falcon [2], H-Match [6], Lily [10], and TaxoMap [11] for mapping, re-mapping, and reconciliation of mappings with our extensions using the Change History Log when one of the mapped ontologies changes.

Mapping system	Mouse onto & Human onto (m)	Brinkman onto & GTT onto (m)	GEMET onto & NALT onto
Falcon mapping time	8.89	32.40	51.33 m
H-Match mapping time	10.76	39.13	1.12 h
Lily mapping time	13.22	34.03	52.76 m
TaxoMap mapping time	11.08	33.41	52.65 m
Falcon re-mapping time	9.17	33.36	52.43 m
H-Match re-mapping time	11.86	40.47	1.14 h
Lily re-mapping time	14.15	35.51	52.91 m
TaxoMap re-mapping time	12.73	34.09	53.76 m
Extended Falcon re-mapping time	**1.08**	**3.11**	**5.36 m**
Extended H-Match re-mapping time	**1.42**	**2.78**	**7.31 m**
Extended Lily re-mapping time	**1.93**	**4.08**	**6.73 m**
Extended TaxoMap re-mapping time	**1.87**	**3.64**	**5.92 m**

Table 2
Computation time analysis of Falcon [2], H-Match [6], Lily [10], and TaxoMap [11] for mapping, re-mapping, and reconciliation of mappings with our extensions using the Change History Log when both mapped ontologies change.

Mapping system	Mouse onto & Human onto (m)	Brinkman onto & GTT onto (m)	GEMET onto & NALT onto
Falcon mapping time	8.89	32.40	51.33 m
H-Match mapping time	10.76	39.13	1.12 h
Lily mapping time	13.22	34.03	52.76 m
TaxoMap mapping time	11.08	33.41	52.65 m
Falcon re-mapping time	9.87	34.63	53.71 m
H-Match re-mapping time	12.35	41.55	1.17 h
Lily re-mapping time	15.43	37.20	54.97 m
TaxoMap re-mapping time	13.21	35.93	55.36 m
Extended Falcon re-mapping time	**2.36**	**5.06**	**9.48 m**
Extended H-Match re-mapping time	**2.96**	**4.88**	**12.39 m**
Extended Lily re-mapping time	**3.45**	**6.75**	**10.37 m**
Extended TaxoMap re-mapping time	**2.97**	**6.09**	**10.18 m**

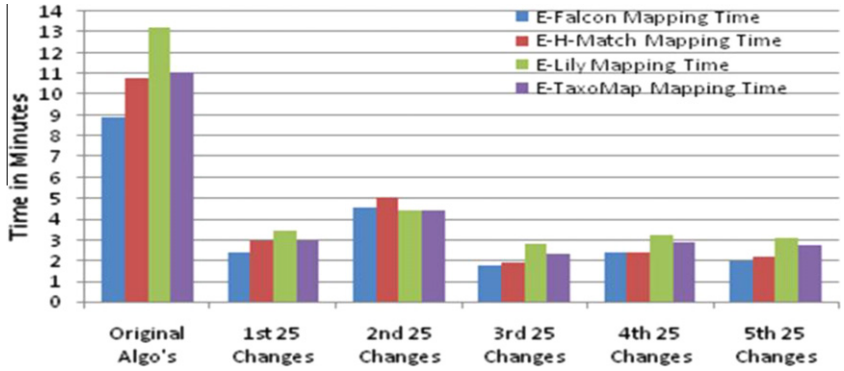

Fig. 4. Mapping and re-establishment (reconciliation) of mapping results for Mouse and Human ontologies. The first bar combination is the result of the original Falcon [2], H-Match [6], Lily [10], and TaxoMap [11], while remaining bar combinations are the results of our proposed approach to these existing systems.

Table 3
Space consumption analysis of Falcon [2], H-Match [6], Lily [10], and TaxoMap [11] against the systems with proposed extensions using CHL during the mapping reconciliation procedure. Memory size is represented in KBs, whereas all the memory usage values are the peak values recorded during systems execution.

Ontology	Falcon space usage	Extended Falcon space usage (KB)	H-Match runtime space usage (KB)	Extended H-Match runtime space usage (KB)	Lily runtime space usage (KB)	Extended Lily runtime space usage (KB)	TaxoMap runtime space usage (KB)	Extended TaxoMap runtime space usage (KB)
Original human vs. Original mouse	187,003	187,320	193,241	193,551	129,393	129,544	107,061	107,392
Human V1 vs. Mouse V1	189,934	**93,698**	194,091	**87,814**	134,319	**81,802**	107,801	**44,061**

CHL instance containing sets of changes corresponding to their *ChangeSets* (change sessions) is maintained in the same directory as the original evolving ontology. When the mapping reconciliation procedure is activated, the required *ChangeSet* with corresponding changes is fetched from the CHL residing in the same directory. This eliminates the need to maintain a separate lookup table for the corresponding CHL location.

The time to reconcile mappings between ontologies using our proposed extensions depends on the types of changes made. A single change may have cascading effects on existing resources or may result in several induced changes [1]. In our approach, computational time (time complexity) is directly proportional to the number of changes in ontology and/or the effects of the changes on existing ontology resources. With more changes in the ontology, mapping time complexity will increase but will still be less than the original algorithms. Mostly, the cascade effects and induced changes are due to the change in the higher level hierarchy. This sort of change is less frequent once the domain ontology matures [1,12]. One such case (showing the variation in time complexity) is also visible in the third comparison of Fig. 4. The *x*-axis of Fig. 4 shows the number of tests, and the *y*-axis is the time in minutes required for mapping reconciliation using the original algorithms and the proposed extensions. The first bar combination is the original time of the existing systems (Falcon, H-Match, Lily, and TaxoMap) for establishing the mappings between *Human* and *Mouse* ontologies while the remaining bars are the time results for the reconciliation of mapping with our proposed extensions using *CHL*. Even with the cascaded effects and induced changes, our proposed approach requires less mapping computation time than the original algorithms.

The proposed algorithm is both time efficient and space efficient because the mapping reconciliation procedure loads ontology from one side and changes from another side. The changes are smaller than with the original ontology. The proposed system's runtime memory usage is compared with the existing system's memory usage and the results (see Table 3) show that the proposed system memory consumption is less than that of the existing system. Moreover, as the proposed system require lesser time for mapping reconciliation, so the memory consumption is for shorter interval of time in comparison with the existing systems. Table 3 shows the results of memory consumption with the proposed extensions compared with the existing systems' traditional approach. The original *Human* and *Mouse* ontologies as well as the changed versions used in the computation time analysis shown in Table 2 are compared. Efficient memory utilization using proposed extensions for mapping reconciliation in changed ontologies is highlighted in Table 3.

The proposed extensions reduce the amount of time required for the mapping reconciliation; however, it is also important to test their effects on the accuracy of the reconciled mappings. Most of the mapping systems developed mainly focuses on the accuracy of the mappings. The accuracy of the mapping is more critical when the services or information systems deal with information from the healthcare domain. To investigate the accuracy of reconciled mappings with proposed extensions, healthcare domain ontologies, i.e., *HL7 Classes ontology* and *openEHR Classes ontology* were used with their two different versions for mapping and mapping reconciliation. These ontologies were tested using Falcon [2], H-Match [16], Lily [10] and TaxoMap [11], and their results were compared with results from proposed extensions to these systems (see Table 4). The changes used in these tests were 25 random changes introduced in different versions of the ontologies and are listed in Table 4, which shows fewer mappings after the reconciliation procedure than those found by the original systems; however, the difference is not large.

In the future, the focus will be on the missing mappings and the reasons for the missing mappings in order to optimize the proposed system for mapping accuracy. Two points must be addressed to improve accuracy. (1) Increase the level of information with the changes, such as with every class change (except class deletion),

Table 4
Mapping accuracy results of proposed extensions to the mapping systems compared to the original mapping systems using HL7 Classes ontology (http://web.science.mq.edu.au/~borgun/Software.html) and openEHR Classes ontology (http://trajano.us.es/~isabel/EHR/). In these tests, only two versions of the ontologies are used. For the mapping process Falcon [2], H-Match [6], Lily [10], and TaxoMap [11] are used.

Mapping system	Ontology 1	Ontology 2	Ontology changes	Mapping time (min)	Number of mappings found
Falcon mapping	HL7 V1	openEHR V1	Original	0.58	18
Falcon re-mapping	HL7 V2	openEHR V2	25 vs. 25	1.18	20
Ext- Falcon re-mapping	**HL7 V2**	**openEHR V2**	**25 vs. 25**	**0.26**	**19**
H-Match mapping	HL7 V1	openEHR V1	Original	2.04	17
H-Match re-mapping	HL7 V2	openEHR V2	25 vs. 25	2.53	17
Ext-H-Match re-mapping	**HL7 V2**	**openEHR V2**	**25 vs. 25**	**1.26**	**16**
Lily mapping	HL7 V1	openEHR V1	Original	1.45	17
Lily re-mapping	HL7 V2	openEHR V2	25 vs. 25	2.09	19
Ext-Lily re-mapping	**HL7 V2**	**openEHR V2**	**25 vs. 25**	**0.49**	**18**
TaxoMap mapping	HL7 V1	openEHR V1	Original	1.63	18
TaxoMap re-mapping	HL7 V2	openEHR V2	25 vs. 25	2.09	20
Ext-TaxoMap re-mapping	**HL7 V2**	**openEHR V2**	**25 vs. 25**	**0.64**	**19**

provide additional information, i.e., its super class and sub classes. Similarly, with every property change (excluding property deletion), additional information on the domain and range are needed. This also improves accuracy. However, this additional information increases the mapping reconciliation time. (2) Expert intervention should be used to resolve semantic conflicts.

4. Conclusions

Mapping between two information sources (i.e., ontologies) is very important for sharing information and achieving interoperability. New discoveries in the field and their presence in domain ontologies result in evolution to another state. The evolution of ontology from one state to another consequently makes the existing mappings between ontologies unreliable. To handle this, we proposed extensions to existing systems by introducing CHL to reconcile mappings in these systems. We found a drastic decrease in the amount of time required for reconciling ontology mappings among dynamic ontologies compared to the already existing systems that reinitiate the entire process.

Currently, we are testing our technique on larger data sets in different combinations in order to verify our claims. Variable mapping accuracy following our technique is a future concern.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MEST) (No. 2011-0030823).

References

[1] G. Flouris, D. Manakanatas, H. Kondylakis, D. Plexousakis, G. Antoniou, Ontology change: classification and survey, Knowledge Engineering Review (KER) 23 (2) (2008) 117–152.
[2] W. Hu, Y. Qu, Falcon-AO: a practical ontology matching system, Journal of Web Semantics 6 (3) (2008) 237–239. doi=http://dx.doi.org/10.1016/j.websem.2008.
[3] A.M. Khattak, K. Latif, S.Y. Lee, Y.K. Lee, Ontology evolution: a survey and future challenges, in: The 2nd International Conference on u- and e-Service, Science and Technology (UNESST 09), Jeju, Korea, December 10–12, 2009.
[4] N. Choi, I. Song, H. Han, A survey on ontology mapping, SIGMOD Record 35 (3) (2006) 34–41.
[5] A. Maedche, B. Motik, N. Silva, R. Volz, MAFRA – a MApping FRAmework for distributed ontologies, in: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, London, 2002, pp. 235–250.
[6] S. Castano, A. Ferrara, S. Montanelli, Matching ontologies in open networked systems. Techniques and applications, Journal on Data Semantics (JoDS) V (2006) 25–63.
[7] P. Shvaiko, J. Euzenat, Ten challenges for ontology matching, in: Proceedings of the 7th International Conference on Ontologies, Darabases, and Applications of Semantics (ODBASE), 2008.
[8] M.C. Valiente, E.G. Barriocanal, M.A. Sicilia, Applying an ontology approach to IT service management for business-IT integration, Knowledge-Based Systems 28 (2012) 76–87.
[9] B. Philip, E. Eiman, F. Wenfei, F. Michael, Putting context into schema matching, in: Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), Seoul, Korea, 2006, pp. 307–318.
[10] P. Wang, B. Xu, Lily: ontology alignment results for oaei 2009, Ontology Matching (OM) 15 (2009).
[11] F. Hamdi, B. Safar, N.B. Niraula, C. Reynaud, Taxomap alignment and refinement modules: results for OAEI 2010, in: Proceedings of the 5th International Workshop on Ontology Matching (OM-2010) Collocated with the 9th International Semantic Web Conference (ISWC-2010), CEUR-WS, 2010, pp. 212–220.
[12] A.Y. Halevy, Z.G. Ives, M. Jayant, P. Mork, D. Suciu, I. Tatarinov, The Piazza peer data management system, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 787–798.
[13] A.M. Khattak, K. Latif, S. Khan, N. Ahmed, Managing change history in web ontologies, in: Semantics, Knowledge and Grid, International Conference on Semantic, Knowledge and Grid (SKG), 2008, pp. 347–350.
[14] R. Fagin, P. Kolaitis, L. Popa, G. Tan, Schema mapping evolution through composition and inversion, Schema Matching and Mapping, Data-Centric Systems and Applications (2011) 191–222.
[15] Y. An, T. Topaloglou, Maintaining semantic mappings between database schemas and ontologies, Semantic Web, Ontologies and Databases, Lecture Notes in Computer Science (2008) 138–152.
[16] M. Lenzerini, Data integration: a theoretical perspective, in: Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, New York, USA, 2002, pp. 233–246.
[17] N. Noy, M. Klein, Ontology evolution: not the same as schema evolution, Journal of Knowledge Information Systems 6 (4) (2004) 428–440.
[18] S. Subramaniam, S.C. Haw, P.K. Hoong, s-XML: an efficient mapping scheme to bridge XML and relational database, Knowledge-Based Systems 27 (2012) 369–380.