Ontology-based Change Propagation in Shareable Health Information Applications

Anny Kartika Sari
Dept. of Computer Science and Electronics
Gadjah Mada University
Yogyakarta, Indonesia

Wenny Rahayu
Dept. of Computer Science and
Computer Engineering
La Trobe University
Melbourne, Australia

***Abstract*—One of the most important challenges to be addressed when establishing an integrated smart health environment is the availability of shareable health data and knowledge which standardize the interoperability of components within the environment. Health ontologies are commonly utilized to enable interoperability between applications in such environment. However, the dynamic nature of health knowledge causes the need for frequent changes in health ontologies which then must be propagated to the relevant applications. A change propagation method that can efficiently streamline the change management from an ontology to all the applications which reference to it is proposed. A component called a *mapper* is used to manage the mapping between application terms and ontology concepts. The mapper is aimed to maintain the applications' access to the most up-to-date ontology concepts and to improve the semantic mapping between the application terms and the ontology concepts. Some rules are developed for the change propagation process. The evaluation of the method shows that the mapper can improve the mapping list in terms of: (i) correctness, by proposing a new mapping entry to substitute an existing one which is not valid anymore because ontology concept is deleted or changed; (ii) currency maintenance, by recommending a better mapping between an application term and a new ontology concept based on the similarity value between the term and the new concept.**

Keywords*—*health information system; ontology-based application; ontology evolution

I. Introduction

Current developments in the health domain require patients' data to be exchanged between information systems of different health care providers. Today, people are highly mobile, thus they can access health care treatment from different providers who can be geographically separated. Specialization in the health care domain also requires flexibility in the exchange of patient data. However, it requires the availability of shareable health data and knowledge which standardize the interactions of components within the environment.

Semantic interactions in the environment can be standardized using health ontologies. Several health ontologies such as SNOMED CT (Systematized Nomenclature of Medicine–Clinical Terms), UMLS (Unified Medical Language System) and LOINC (Logical Observation Identifiers Names) have been established to achieve semantic interoperability between different health providers in the environment. While an ontology-based health information system application must refer to the most current ontology, health ontologies constantly change due to changes in the knowledge of the health domain. The frequent changes in health ontologies may become a problem in the effort to ensure the applications refer to the most up-to-date ontology concepts.

In the notion of ontology evolution, there is a phase which is related to the effort of maintaining the currentness of the ontology-based applications with regard to the ontology to which they refer. This phase is named *ontology change propagation*. The goal of this phase is to bring the changes of the ontology to the depending artifacts such as other ontologies or the applications based on it. Since ontologies and applications have different characteristics, the ontology change propagation process is classified into two types: *ontology-to-ontology* change propagation and *ontology-to-application* change propagation. In this work, the focus is on the change propagation from a base ontology to the applications.

The complexity of change propagation process depends on how the ontology bound to the applications. The process will be easier for applications in which the concepts in the ontology are not tightly bound to them. This means that the ontology components are not hard-coded/embedded in the applications. Such applications only access the ontology 'on the fly', that is, the applications only need it when they are executing a process. For instance, in an ontology-based decision support system, an ontology is needed during the reasoning process to support decision making. In this type of application, there is no continuous direct binding between ontology concepts and application terms. The applications need the ontology as a whole, not only particular components. Once the ontology changes, the applications can be immediately directed to the new ontology. The impact of the ontology changes to the applications is not significant because the ontology components are not hard-coded in the applications.

For applications where ontology components are embedded, different approaches should be used. In this type of application, ontology components are used continuously, and may be hard-coded in the applications. For instance, in applications which utilize ontology to achieve interoperability between different health information systems, there may be one-to-one mapping between each term in the application and an ontology concept. In this type of application, changes in ontology should be propagated to applications straight away so that the applications always refer to the current ontology and interoperability is maintained. However, direct propagation may affect the validity of data or cause inconsistency. Furthermore, sometimes the nature of the applications make

it impossible to make frequent changes to the applications because they may raise some technical issues. In this type of application, a change propagation process which does not directly affect the applications would be more appropriate. To the best of our knowledge, there is no existing work on ontology-to-application change propagation which considers this matter.

In this paper, a method to handle ontology changes in an ontology-based application is proposed. The main focus of this work is the ontology-to-application change propagation process, specifically from the ontology to the depending applications which are constantly bound to the ontology concepts. A component, referred to as the *mapper*, is responsible for managing the mapping list such that the application terms can always be bound to the current ontology concepts. The main advantage of the mapper is that the binding between the application terms and the ontology concepts can be done outside the application so that the ontology changes can be handled without the need to modify the application. Some rules are developed as guidelines for the mapper to perform its task. The mapper has two important roles. Firstly, it can propose a new mapping entry to substitute an existing one which is not current due to a deletion or change of the ontology concept listed in the entry. Secondly, the mapper can propose a better mapping of an application term because a new concept in the ontology is found to be more semantically similar to the term than the existing concept previously bound to the term. In this way, the mapping list can be kept up-to-date, while its quality is improved.

The rest of the paper is structured as follows. Section 2 outlines related work. Section 3 presents the connection mechanism between ontology and the applications. The main focus of this work is discussed in Section 4, which explains the method in propagating the changes from the ontology to the applications and managing the mapping list when the ontology changes. Section 5 discusses the evaluation of the method. Section 6 concludes the work.

II. Related Work

Previous work on the management of ontology evolution has been proposed. Some of the important frameworks are CONCORDIA ([1], [2]), CREAM ([3]), OntoView ([4]), KAON ([5], [6]), CHAO ([7]), Evolva ([8], [9]), GOMMA ([10]), COnto-Diff ([11]) and CHO (Change History Ontology) ([12], [13], [14]). However, only a few of the frameworks address the change propagation phase, such as [2], [6], [7], [15], [16] and [17]. most of them only discusses the ontology-to-ontology change propagation method, not the ontology-to-application method. Our previous work in [18] and [19] also discusses an ontology-to-ontology approach with the focus on the change propagation from a base ontology to a sub-ontology. While the approach in [18] is only based on the change operations provided by the release of the health ontologies, the method proposed in [19] considers the semantic of the change operations.

The work on ontology evolution in ontology-based applications which is related to ontology-to-application change propagation method is summarized in Table I. The proposed approach differs from the existing work summarized in the table in at least two issues. Firstly, the focus is on health ontologies which have some specific characteristics as follow: (i) they have been standardized; (ii) they change very frequently; (iii) their size is very large, and; (iv) the domain is very critical. The ontologies which have become the focus of the existing work do not have these characteristics. Secondly, the main interest of the change propagation in this work is the change propagation to the applications which use the concepts constantly in the direct binding between the concepts and the application terms. To the best of our knowledge, there has not been any existing work which focuses on this issue. In most of the existing work on the ontology-to-application change propagation, including the work listed in I, the applications utilize ontologies in only two ways: instances and queries. In [24], an ontology-based method to handle terminology changes is proposed. The work focuses on International Classification of Diseases (ICD-9-CM) terminology, which is one of the standardized terminologies commonly used in medical area. However, the work does not consider the change propagation process from the terminology to the applications.

III. The Connection between Ontology and Applications

Figure 1 shows a description of the framework used in distributed health provider systems. It consists of a main ontology, a main ontology manager and several different health provider systems, each of which contains an ontology suitable for the system which is referred to as the *referred ontology*. Each health provider system also contains some health information system applications. The ontology manager is the key component in the framework. It manages the ontologies by doing two tasks: 1) keeping the ontologies up-to-date by propagating the changes which occur in the main ontology to the relevant referred ontologies and; 2) giving notification to each health provider system whenever there are changes in its referred ontology as a consequence of the changes in the main ontology. In this paper, only the second task is discussed because it is related to the mapping mechanism between ontology concepts and application terms.

As previously mentioned, continuous access to ontology concepts by the applications will include direct reference or mapping between the ontology concepts and the application terms. To make the references more well-structured, a *mapping list* is used. It includes the mapping between the terms used in the applications and the concepts included in the referred ontology. A component referred to as the *mapper* is also used to manage the mapping list and to handle the changes when the ontology evolves. Since the mapper and the mapping list are not part of the applications, the ontology components do not need to be hard-coded in the applications. By separating the mapping list and the mapper from the applications, management of the mapping when the ontology changes will be easier. The application does not need to do the adjustment every time the ontology changes because the mapping in the mapping list has been adjusted by the mapper.

Figure 2 shows the detail of the health provider system components and the processes which occur when the referred ontology changes. The tasks of the main ontology manager are to provide the referred ontology with the ontology components and to notify the mapper when there is an ontology change. It

TABLE I: Some existing works on ontology-to-application change propagation

Article	Ontology	Goal	Application
[20]	Legal Ontology	To discover inconsistencies in a semantic web service description, whose repairing improves the agreement of the ontology with the business rules	e-gov change management system
[3], [5]	not specified	To enable consistency in the annotations of knowledge sources in the case of changes in the domain ontology	CREAM, a semantic annotation framework
[21]	The Ontology of Professional Judicial Knowledge (OPJK)	To provide ontology managers and users with a tool that helps to detect effects of changes in ontologies and select versions based on their properties	MORE (Multi-version Ontology REasoner)
[22], [23]	CIDOC Conceptual Reference Model (CRM) ontology	Ontology-based system could provide continuous and unchanged services to the end-users	Applications based on CIDOC CRM ontology

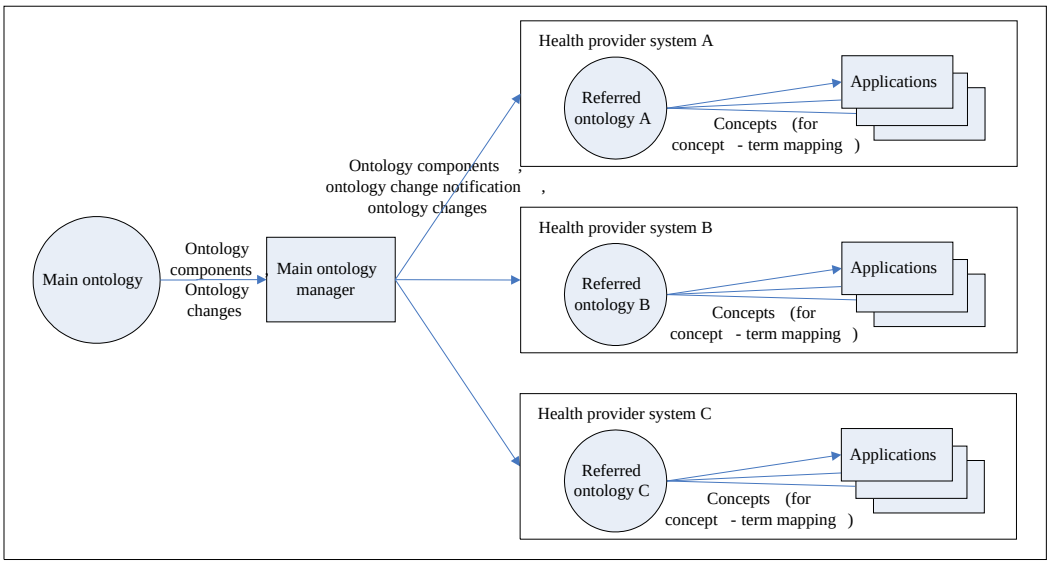

Fig. 1: Framework of the distributed health provider systems

also provides the mapper with a list of the change operations applied to the referred ontology. Based on the list of the change operations, as well as the term-concept mapping data provided by the mapping list, the mapper do the adjustment to the mapping list. The applications, or to be more specific the user/administrator of the applications, can accept or reject the adjustment.

At any given time, the health provider keeps the valid and up-to-date ontology, which is consistent with the main ontology, to be referred to by the applications. However, due to technical issues or data loss, sometimes there are applications which still refer to the concepts from the previous version of the ontology which are not included in the current version because they have been deleted or changed. To anticipate this situation, the old concepts are included as an extension of the valid ontology, which is referred to as the *extended concepts*. These extended concepts, together with the valid and up-to-date ontology, construct the referred ontology. Figure 3 shows the contents of a referred ontology. In the figure, 'Ontology' refers to the valid and up-to-date ontology. The extended concepts are not components of the valid ontology. Each of the extended concepts is annotated with the information of the concept in the valid referred ontology which is related to it, that is, the new concept which replaces or represents the extended concept. Moreover, information on the version of the main ontology in which it was originally included is also available. The formal definition of the annotation for the extended concept ce follows.

Definition 1. Annotation for ce
$A(ce) \equiv < c, v >$ *is the annotation for the extended concept ce where c is the concept in the current referred ontology related to ce and v is the version of the referred ontology in which ce was included.*

The mapping list contains several mapping entries, each of which connects a term used in the application to a concept in the referred ontology. The formal definition of the mapping list is as follows.

Definition 2. Mapping list
$L \equiv \{l_1, l_2, ..., l_n\}$ *is the mapping list with* $l_i \equiv < a_i, t_i, c_i, v_i, df_i, dt_i, s_i, r_i >$ *is the mapping entry.*

In the definition, L is the mapping list which is a set of mapping entries l_i. l is the mapping entry in the mapping list and has the following structure:

<application_id, term_id, concept_id, version, date-from, date-to, status, reason >

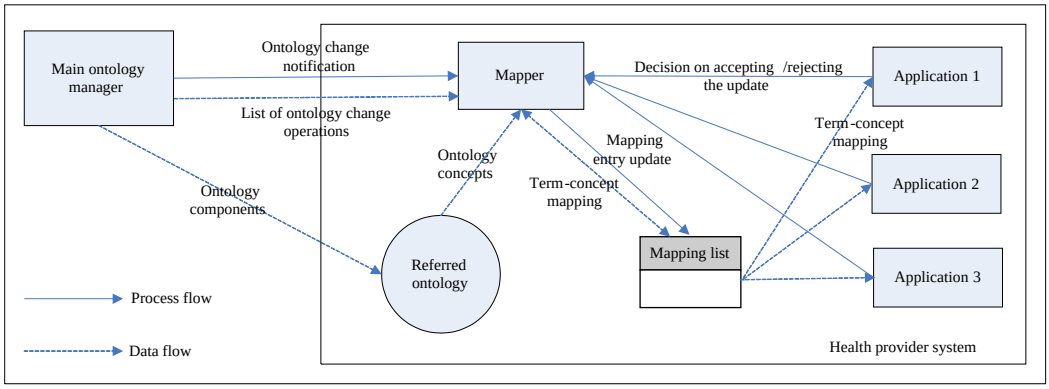

Fig. 2: The components inside a health provider system and the process and data flows when the main ontology changes.

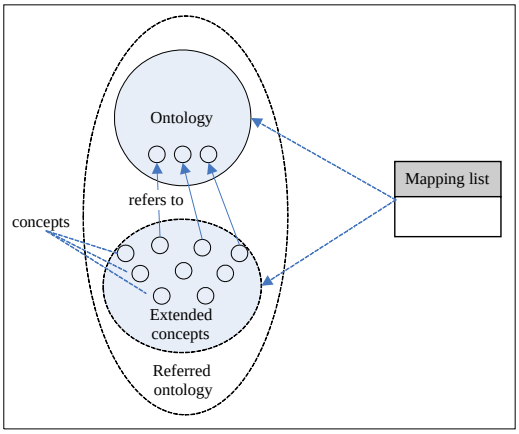

Fig. 3: The valid and up-to-date ontology and the extended concepts construct the referred ontology.

Application_id (a) is the ID of the application where the term is used. *Term_id* (t) is the ID of the term which refers to the ontology concept. *Concept_id* (c) is the ID of the concept in the referred ontology. Each concept which ID is currently used in the mapping list is referred to as a *referred concept*. *Version* refers to the most current version of the referred ontology where the concept is included. *Date-from* (df) is the date when the mapping is created. *Date-to* (dt) is the date when the mapping becomes obsolete due to ontology changes or application changes. *Status* (s) is the status of the mapping which can be: (i) 'valid', which means that the mapping is still used; (ii) 'invalid', which means that the mapping is not used anymore; (iii) 'new', which means that the mapping is produced by the mapper due to finding a new version of the referred ontology; or (iv) 'obsolete', which means that the mapping should be checked for its validity due to the change in the referred ontology. The *reason* (r) explains why a mapping entry is not valid anymore.

During the change propagation process, the mapping list can be changed. We define two operations which are related to the change of the mapping entries in the mapping list: *update* operation and *addition* operation. Deletion operation is not defined because the history of the mapping entries should be maintained in the mapping list. The formal definition of the update and addition operations is as follows.

Definition 3. Mapping entry update operation
$Given\ the\ mapping\ entry\ l_1 \equiv < a_1, t_1, c_1, v_1, df_1, dt_1, s_1, r_1 >$.
$UpdateMap(< a_1, t_1, c_1, v_1, df_1, dt_1, s_1, r_1 >, < a_2, t_2, c_2, v_2, df_2, dt_2, s_2, r_2 >) \equiv (a_2 \leftarrow a_1) \wedge (t_2 \leftarrow t_1) \wedge (c_2 \leftarrow c_1) \wedge (v_2 \leftarrow v_1) \wedge (df_2 \leftarrow df_1) \wedge (dt_2 \leftarrow dt_1) \wedge (s_2 \leftarrow s_1) \wedge (r_2 \leftarrow r_1)$

Definition 4. Mapping entry addition operation
$Given\ mapping\ list\ L \equiv \{l_1, l_2, ..., l_n\}$.
$AddMap(l_{n+1}, L) \equiv L \leftarrow L \cup \{l_{n+1}\}\ with\ l_{n+1} \equiv < a_{n+1}, t_{n+1}, c_{n+1}, v_{n+1}, df_{n+1}, dt_{n+1}, s_{n+1}, r_{n+1} >$.

Basically, the mapping entry update operation is used to update the field values of an existing mapping entry. One or more of the field values can be changed to the new and correct one. The mapping entry addition operation is applied when a new mapping entry needs to be added to the mapping list. The field values of the new mapping entry have been defined before the addition of the mapping entry.

IV. Change Propagation Process

The change operations in the ontology can be applied to all components of an ontology which can be concepts, relationships, description and description mappings. However, since generally the applications of the health provider system only use the concepts to be referred to, in this paper, only the changes related to concepts are considered. There are four types of concept change operations in the ontology:

1) $AddCon(c)$: adds a new concept c to the ontology. The concept can be a leaf or non-leaf concept.
2) $DelCon(c)$: deletes an existing concept c from the ontology. No new or existing concept is proposed to represent the meaning of c.
3) $ChangeCon(c_1, c_2)$: changes concept c_1 into c_2, and then c_1 is deleted from the ontology. This means that c_2 can be used to represent the meaning of c_1.
4) $MovCon(c)$: moves concept c to another branch in the ontology graph. Movement of a concept does not influence the entries in the mapping list because the concept still exists in the ontology.

Figure 4 shows the processes which occur in the mapper upon changes to the referred ontology. A list of the ontology change operations is provided by the main ontology manager. It contains the list of concepts which are deleted, added, moved or changed. When the main ontology has been changed, the main ontology manager performs the 'notify ontology changes' process. This process triggers the processes performed by the mapper. Three processes are performed by the mapper. The first process is the 'check mapping entries'. This process examines each mapping entry with a 'valid' status and checks whether the concept in that entry is listed in the list of the ontology change operations provided by the main ontology manager. The concepts which are not included in the mapper but are referred to by any extended concepts through their annotation are also examined. The second process is the 'update mapping list'. In this process, the mapper updates the mapping list according to the rules for the mapping list changes which will be discussed later. The updates, which can be editing or addition of entries, are proposed to the user for agreement. The last process is the 'update mapping list according to the user's decision'. In this process, the mapper updates the mapping list based on the user's decision on the proposed mapping list produced by the earlier process. We consider that a decision by the user (administrator, engineer or expert) is needed in this process because the user might not want to use the evolved version of the referred ontology or he may reject some of the new entries proposed by the mapper due to technical issues or other reasons. After the mapping list is changed in accordance with the user's decision, the referred ontology should be adjusted to suit the decision of the user. The adjustment is especially needed when the user chooses not to change the mapping entry, which specifically influences the extended concepts part of the referred ontology.

When the mapper receives notification from the main ontology manager that the referred ontology has been changed, it automatically searches for mapping entries with 'valid' or 'not used' status. If a concept in a mapping entry is also listed in the list of changes, or if a concept in the list of changes is referred to by an extended concept included in the mapper, the mapper will do the adjustment to the mapping entry by performing two actions:

1) It changes the status field from 'valid' to 'obsolete'. The reason field is also set to either 'deleted concept', 'changed concept', 'new child' or 'new parent', which is chosen according to the type of change operation corresponding to the entry.
2) As a replacement, it proposes new mapping entries with the same application_id and term_id values, but the value of the concept_id field is different, which is determined by several rules. The status field is set to 'new'.

Not all 'valid' mapping entries are affected by the ontology changes. Rules are defined to determine which mapping entries must be updated due to the corresponding change operation. The rules are built based on the type of change operation, as described above. In the following description, the impact of each change operation type is explained and underlies each rule. As previously mentioned, the $MoveConc$ operation does not influence the mapping list, thus the rule for the operation is not defined. Hence, there are only three rules which correspond to the three types of change operations.

1) Impact on the concept deletion operation ($Delcon(c_1)$)
When the concept_id field of a mapping entry refers to a concept to be deleted in the ontology, or when a concept to be deleted from the ontology is referred to by an extended concept contained in a valid mapping entry, the mapping entry must be updated. The value of the concept_id field must be changed to the ID of the *candidate concept*, i.e. the most similar child or parent concept of the deleted concept. The formal definition of the rule follows.

Rule 1: Propagation of $DelCon(c_1)$ operation
$\forall\ DelCon(c_1) \mid \exists\ l_1 \in L,\ l_1 \equiv < a_1,\ t_1,\ c_x,\ v_1,\ dd/mm/yyyy,\ null,\ 'valid',\ null > \ \wedge (c_x = c_1\ \vee\ c_x = ce \mid ce\ is\ an\ extended\ concept\ \wedge A(ce) \equiv < c_1,\ v_{ce} >) :$
$c_i\ is\ the\ candidate\ concept\ \rightarrow\ UpdateMap(l_1, < a_1,\ t_1,\ c_x,\ v_1,\ dd/mm/yyyy,\ null,\ 'obsolete',\ deletion' >) \wedge\ AddMap(< a_1,\ t_1,\ c_i,\ v_{current},\ null,\ null,\ 'new',\ 'null' >, L)$

The candidate concept is determined in the following way. The similarity between a concept c_1 and its parent c_2 is defined as $sim_{cp}(c_1, c_2)$ or $sim_{cp}(c_2, c_1)$, and the value is calculated using the child/parent pair similarity formula proposed in [25]. If c_1 is the deleted concept, while $c_2, ..., c_n$ are the set of parent or child concepts of c_1 in the previous referred ontology, $sim_{cp}(c_1, c_2), sim_{cp}(c_1, c_3), ..., sim_{cp}(c_1, c_n)$ are calculated. If there is c_i with $c_i \in c_2, ..., c_n$ where $sim_{cp}(c_1, c_i) = max\{sim_{cp}(c_1,\ c_2), ..., sim_{cp}(c_1, c_n)\}$, then c_i is referred to as the *candidate concept*. If there are $c_i, c_{i+1}, ..., c_{i+x} \in c_2, ..., c_n$ which have the same child/parent similarity value to c_1, MESH (Medical Subject Heading) vocabulary is used to find which of the names of $c_i, c_{i+1}, ..., c_{i+x}$ are in the same descriptor record to the name of c_1, in which the root words of the names of those concepts are used. If c_i is the only concept whose name is in the same descriptor record to the name of c_1 in MESH, c_i is the candidate concept. Otherwise, Jaro-Winkler distance is used to find the candidate concept whose name is the most similar to the name of c_1. Jaro-Winkler distance is adequate to find the candidate concept because using two level of assessment, i.e. child/parent similarity and MESH descriptor containment, basically the concepts have the same similarity value to the deleted concept. Hence, they are differentiated by their names.

2) Impact on the concept change operation ($ChangeCon(c_1, c_2)$)
When the concept_id field of a mapping entry refers to a concept to be changed in the referred ontology, that mapping entry must be updated. The concept_id field of the new proposed mapping entry must refer to c_2, which is the concept to which c_1 is changed in the ontology. The formal definition of this rule follows.

Rule 2: Propagation of $ChangeCon(c_1, c_2)$ operation
$\forall\ ChangeCon(c_1, c_2) \mid \exists\ l_1 \in L,\ l_1 \equiv < a_1,\ t_1,\ c_x,\ v_1,\ dd/mm/yyyy,\ null,\ 'valid',\ null > \ \wedge (c_x = c_1\ \vee\ c_x = ce \mid ce\ is\ an\ extended\ concept\ \wedge A(ce) \equiv < c_1,\ v_{ce} >) :$

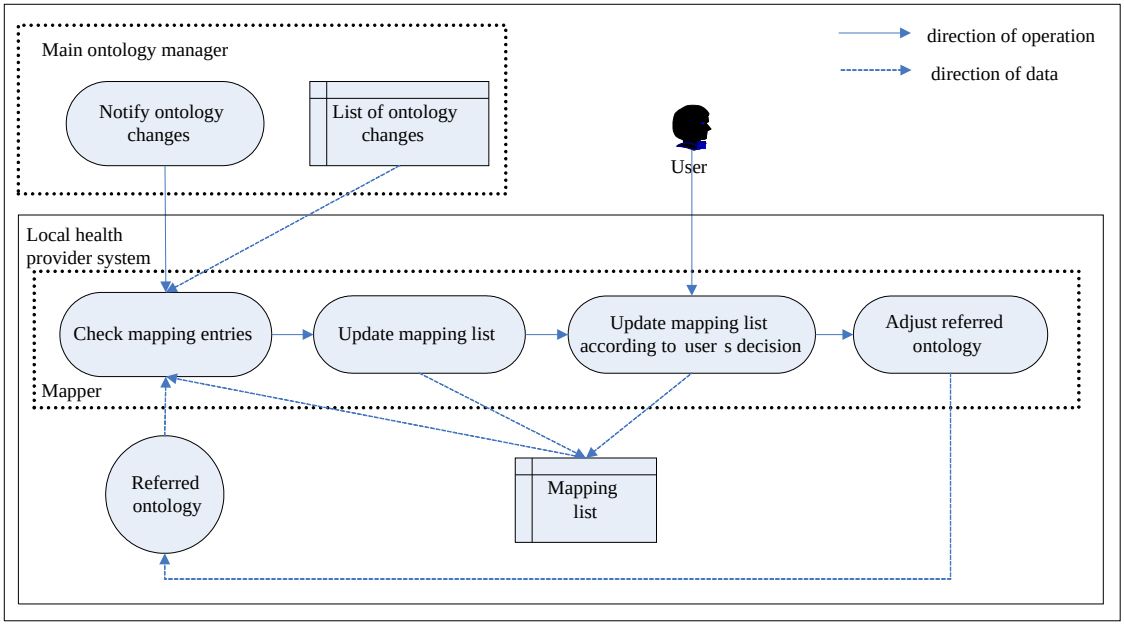

Fig. 4: The process of updating the mapping list when the ontology changes.

$UpdateMap(l_1, < a_1,\ t_1,\ c_x,\ v_1,\ dd/mm/yyyy,\ null,\ 'obsolete',\ 'change' >) \wedge\ AddMap(< a_1, t_1, c_2,\ v_{current},\ null,\ null,\ 'new',\ null >, L)$

3) Impact on the concept addition operation ($AddCon(c_2)$) When the concept_id field of a mapping entry refers to concept c_1 which happens to be the parent or child concept of the new added concept c_2, that mapping entry must be checked for the possibility of update. If t_1 is the value of the term_id in the mapping entry and $sim(t_1, c_2)$, i.e. the similarity value between t_1 and c_2, is higher than $sim(t_1, c_1)$, the mapping entry must be updated and a new mapping entry is proposed, otherwise nothing happens. The formal definition of the rule follows.

Rule 3: Propagation of $AddCon(c_2)$ operation
$\forall\ AddCon(c_2) \mid (\exists\ l_1 \in L,\ l_1 \equiv < a_1,\ t_1,\ c_x,\ v_1,\ dd/mm/yyyy,\ null,\ 'valid',\ null > \wedge\ (c_x = c_1\ \vee\ c_x = ce \mid ce\ is\ an\ extended\ concept\ \wedge A(ce) \equiv < c_1, v_{ce} >))$
$\wedge (c_1\ is\ the\ parent\ or\ child\ concept\ of\ c_2) :$
$sim(t_1, c_2) > sim(t_1, c_1) \rightarrow UpdateMap(l_1, < a_1, t_1, c_1, v_1, dd/mm/yyyy, null, 'obsolete', 'addition' >)$
$\wedge\ AddMap(< a_1, t_1, c_2,\ v_{current},\ null,\ null,\ 'new',\ null >, L)$

To evaluate the similarity values, MESH is used. If the name of c_2 is in the same descriptor record as the name of t_1 while it is not the case for the name of c_1, $sim(t_1, c_2) > sim(t_1, c_1)$. If it is the other way around, then $sim(t_1, c_1) > sim(t_1, c_2)$. The similarity value cannot be determined using MESH if one of these cases occurs: 1) the name of t_1 is not found in MESH descriptor records; 2) the names of c_1 and c_2 are in the same descriptor record as the name of t_1; 3) neither the name of c_1 nor c_2 is in the same descriptor record as the name of t_1. In these cases, Jaro-Winkler distance is used to find which concept name is more similar to the name of t_1.

After the mapper updates the mapping list based on the above rules, the user receives a notification that the mapping list has been updated due to the ontology change. This is included in the 'update mapping list by the user' process described in Figure 4. The user examines all mapping entries with status value 'new'. He should change the 'new' status to 'valid' if he agrees to the new mapping, or 'invalid' if he does not accept it. Following the changes by the user, the mapper adjusts the mapping list using the following guidelines:

- If the 'new' status is changed to 'valid', the status of the corresponding mapping entry with the same application ID and term ID with 'obsolete' status is changed to 'invalid' by the mapper and the date-to field is set to the current date. The date-from value of the new entry is set to the current date. Hence, the new mapping entry is used by the applications to replace the old one.
- On the other hand, if the 'new' status is changed to 'invalid', the status of the other mapping entry with the same application ID and term ID with 'obsolete' status is changed back to 'valid' by the mapper. The status field of the new entry is set to 'not used'. The 'not used' status of a mapping entry indicates that the entry has not been used by the applications since the changes to the referred ontology. In other words, the 'valid' mapping entry with the same application ID and term ID does not refer to the current ontology.

The mapping list adjustment due to the decision by the user may have an impact on the referred ontology, especially to the extended concepts. This happens when the user decides

not to use the new proposed mapping entry to replace the obsolete one in the case of deletion and change operations. If the concept ce referred to by the obsolete mapping entry has not been included in the extended concepts part of the referred ontology, it is included in that part with $A(ce) \equiv < c_1, v >$ where c_1 is the concept proposed by the mapper to replace ce in the mapping list and v is the version of the referred ontology where ce originated. If concept ce has been included in the extended concepts part, only the value of c in the annotation needs to be changed to c_1.

V. Evaluation and Discussion

The proposed method is evaluated by presenting a case study in which a health archetype is used as an example of the application. Following the case study is discussion on the efficiency of the use of the mapper compared to the common method and the maintenance of the semantic currency contained in the applications due to the new mapping entries proposed by the mapper.

A. Application of the Method to the Archetype Term Binding Process

In this section, an application of the proposed method is presented for managing the mapping between the application terms and the ontology concepts when changes occur in the referred ontology. We used an archetype as the representation of an application which refers to the referred ontology. An archetype is a model of specific domain knowledge, in this case, clinical knowledge. Each archetype describes a complete clinical knowledge concept such as 'diagnosis' or 'test result' [26]. For the referred ontology, a sub-ontology, which is derived from SNOMED CT as the main ontology, is developed. The method proposed in [27] is used to build the sub-ontology.

For this work, an archetype is created as a representation of an application. The archetype was named *tooth_care_summary*. The archetype was created by considering the concepts changed in SNOMED CT so that the types of change operations in the sub-ontology can be shown in the archetype. Thus, the archetype itself is a modified version to fulfill the above requirement. The definition of the *tooth_care_summary* archetype is shown in Figure 5, which is previewed using the Archetype Editor -arc built by Ocean Informatics. It contains 40 terms, each of which is bound to a SNOMED CT concept. Based on the concepts required by the binding, a sub-ontology is built. The sub-ontology was extracted from the 20110131 version of the International SNOMED CT edition. The sub-ontology contains 557 concepts and 645 relationships of SNOMED CT. Based on the archetype terms and the sub-ontology concepts, the mapper created a mapping list.

Figure 6 shows part of the initial binding between the archetype terms and the 20110131 version of SNOMED CT concepts viewed by the Archetype Editor. The node column presents the term names, the code column refers to the bound SNOMED CT concepts, while the release column shows the version of SNOMED CT in which each of the concepts is initially included. Note that this is not the actual mapping list.

After the 20110131 version, SNOMED CT has been changed and the newer version is the 20110731 SNOMED CT. To accommodate the changes, the sub-ontology has been changed as well. Some change operations were performed. The mapper checked the mapping entries to see if the change operations were affected by the change operations. Apparently, there were seven operations which were related to the mapping entries. The effect of each change operation to the related mapping entry is summarised in Table II. In Figure 6, the highlighted rows contain the concepts which are affected by the change operations.

The highlighted rows in Figure 7 are the binding between archetype terms and the SNOMED CT concepts which are affected by the sub-ontology changes. It can be seen that the rows contain different concepts ids and release versions of SNOMED CT from the concept ids and release versions contained in Figure 6.

B. Discussion

There are several issues related to the proposed method which will be discussed in this section. These issues are elaborated as follows.

The use of sub-ontologies to increase efficiency in change propagation process

In the proposed method, applications refer to sub-ontologies instead of the base ontology. The number of concepts in a sub-ontology is smaller than the number of concepts in a base ontology. When the base ontology changes, only the relevant changes are propagated to each of the sub-ontologies. In this way, the number of changes in each of the sub-ontologies is also smaller than the number of changes in the base ontology.

After updating its components according to the changes propagated by the base ontology, a sub-ontology then propagates these changes to the applications referring to it. This process is performed by the mapper by updating its mapping list according to the rules of sub-ontology to application change propagation. Since the number of change operations in the sub-ontology is smaller than the number of change operations in the base ontology, the mapper does not need much time to examine all the change operations which occurred in the sub-ontology. The time needed to examine the change operations will be longer if the sub-ontology does not exist, which implies that the mapper must look up all the change operations in the base ontology. In the case of SNOMED CT used in the evaluation of the method, the number of changes which occurred to Version 20110131, which is included in Version 20110731, is 8,697 operations. It will take time for the mapper to check whether each of the operations falls into one of the rules for propagating the changes to applications.

Benefits of the use of the mapper

There are several advantages of the use of the mapper in the change propagation process to applications as follows.

- The mapper maintains the history of the application term references to ontology concepts.
  The mapper never deletes its entries. Invalid (not used) entries are only marked by the 'invalid' value of the

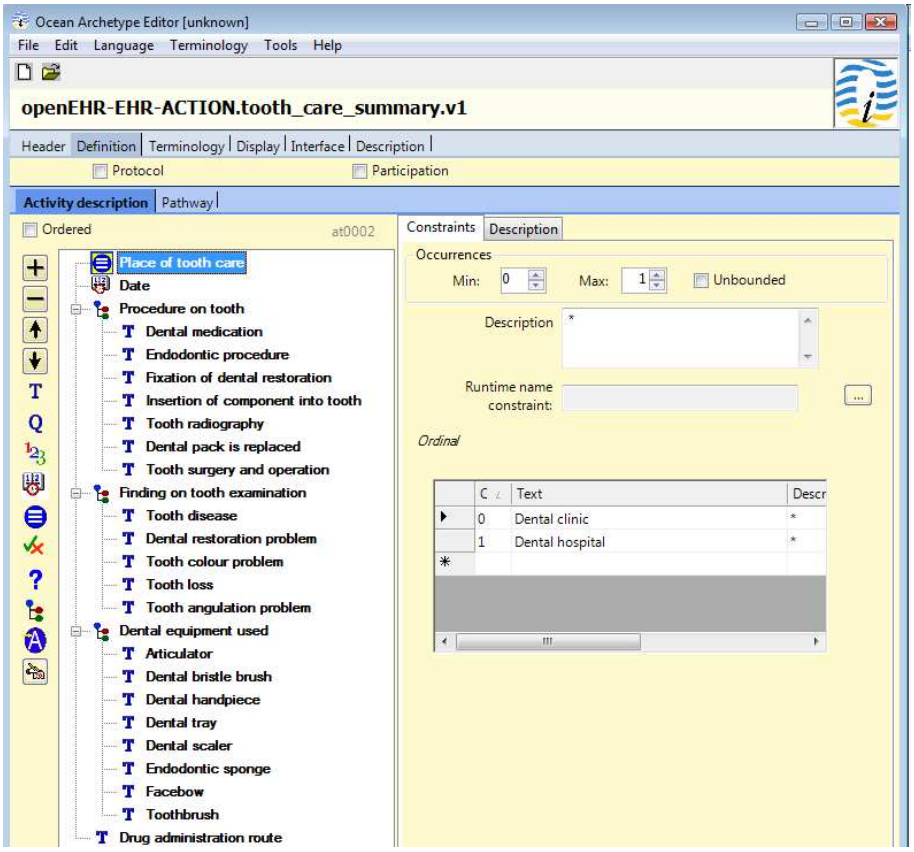

Fig. 5: The definition of *tooth_care_summary* archetype.

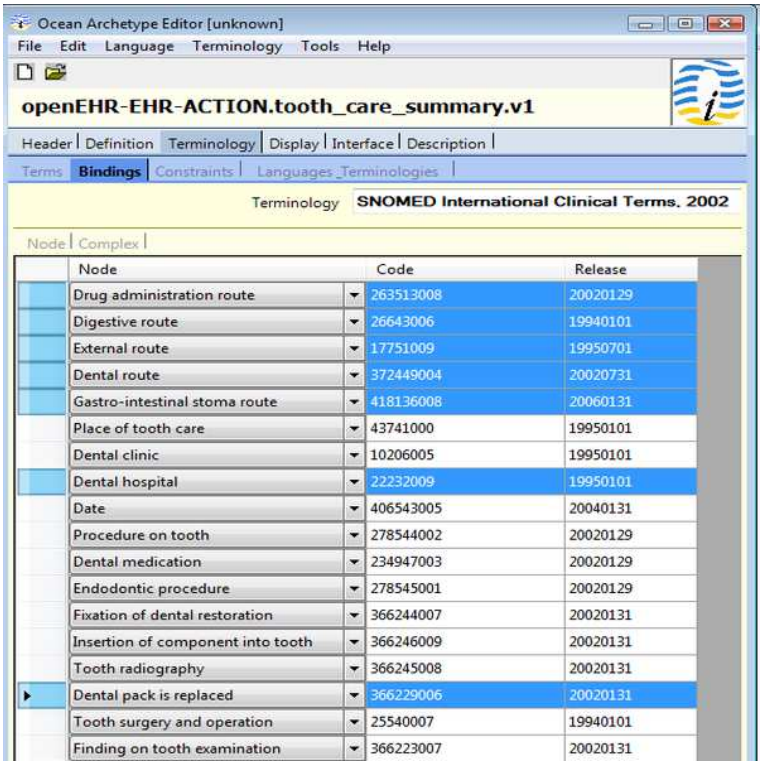

Fig. 6: The binding between the terms in the archetype and the SNOMED CT concepts before the sub-ontology changes.

TABLE II: Change operations found in the sub-ontology which are related to the mapping entries

Change operation in sub-ontology	Related mapping entry and the change					
	Term id	Term name	Concept id of the existing entry	Decision by the mapper	Reason	Concept id of the new entry
MergeCon (263513008, 410675002)	at0037	Drug admin-istration route	263513008	Status is changed to obsolete, new entry is proposed	Concept is merged	410675002
InsertCon (447964005)	at0038	Digestive route	26643006	Status is changed to obsolete, new entry is proposed	The term name is more similar to the name of the new concept	447964005
DelCon (17751009)	at0039	External route	17751009	Status is changed to obsolete, new entry is proposed	The deleted concept has only 1 parent concept (284009009), but no child concept	284009009
MoveCon (372449004)	at0040	Dental route	372449004	No need to update	The moved concept is still included in the sub-ontology	-
InsertCon (372454008)	at0041	Gastro-intestinal stoma route	418136008	No need to update	The term name is not more similar to the name of the inserted concept	-
AddLeaf (448399001)	at0004	Dental hospital	22232009	Status is changed to obsolete, new entry is proposed	The term name is more similar to the name of the new leaf concept	448399001
AddLeaf (447896001)	at0012	Dental pack is replaced	234718000	Status is changed to obsolete, new entry is proposed	The term name is more similar to the name of the new leaf concept	447896001

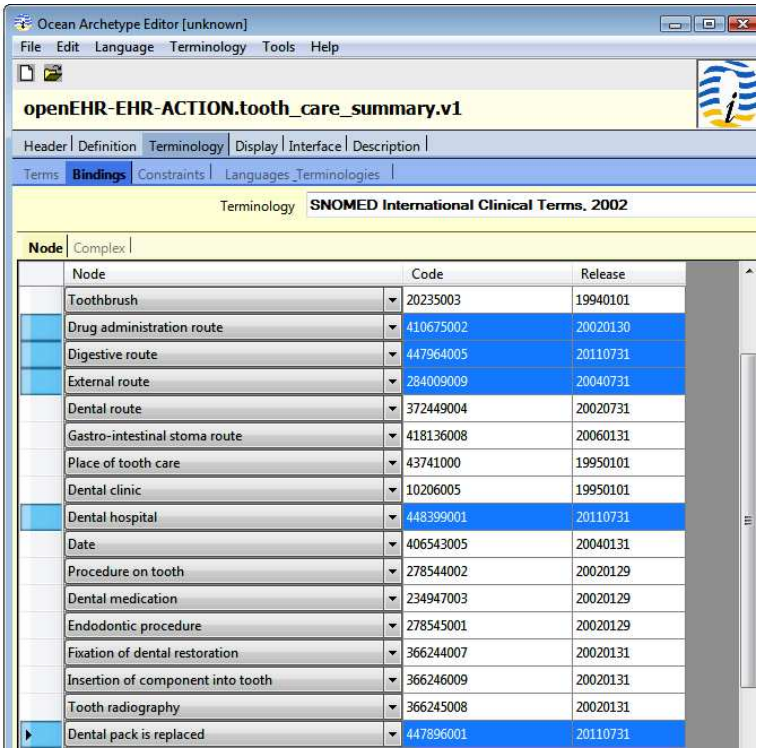

Fig. 7: The binding between the terms in the archetype and the SNOMED CT concepts after the sub-ontology changes.

status field. In this way, the history of references of application terms can be preserved. All references to a particular application term have the same values of *application_id* and *term_id* fields. Each invalid reference has a value for its *reason* field which records the reason why the reference is not used. Furthermore, the user can also check if there are application terms which are out-of-date and may need to be changed. The mapping entries with the most current value of the *version* field and 'not used' value of *reason* show that the corresponding application terms still refer to older ontology concepts.

- The mapper enables the semi-automatic update process of the mapping list.
  A fully manual update is error prone and obviously takes a longer time. In the sub-ontology used in the evaluation, there are 40 terms to be mapped to the sub-ontology concepts. In the case of a manual process, when the sub-ontology changes, the user must check each term to see whether it refers to a deleted sub-ontology concept or not. Since there are 557 concepts in the sub-ontology, the manual checking process is hardly feasible. The mapper can do the checking faster because it has the mapping list which includes the concept ids. Hence, the mapper only needs to find the concept ids which are included in the list of sub-ontology changes given by the central sub-ontology manager.

- The mapper facilitates the management of the mapping between the application terms and the sub-ontology concepts such that the sub-ontology changes can be handled without the need to modify the applications.
  This can be done because the mapping is managed outside the applications. The separation between the mapping mechanism and the applications is important because, unlike an ontology which can be changed automatically, the decision to change an application cannot be made instantly because it may affect the existing data and raise some technical issues related to the application development.

- The mapper can propose a better mapping of the application terms
  In the example, there are three terms which are mapped to the new concepts which are semantically more similar to the terms. In the previous mapping, the terms are mapped to the less similar concepts because they are the best choice in the previous sub-ontology. The new concepts added to the sub-ontology apparently have better similarity to the terms. Again, a manual examination takes time because there are 40 terms to be checked. The mapper can quickly propose the new mapping based on the rule of mapping list changes due to an addition of a leaf concept or an insertion of a concept. This will improve the quality of the mapping between the archetype terms and the ontology concepts. Otherwise, the application terms will maintain their references to less similar concepts, while there are actually concepts which have better similarity to them. At this moment, a mapping is improved by changing the referred concept to its new child or parent concept only in the changed sub-ontology. In the future, improvements might be made by changing the referred concept to any new concept in the sub-ontology or even any new concept in the base ontology. However, a discussion on this issue is beyond this thesis.

Validity of the mapping entries with regard to the current sub-ontology

A concept deleted from the sub-ontology indicates that the concept is not available in the current sub-ontology. In some ontologies such as SNOMED CT, a concept merged to another one is also considered a deleted concept. A reference to a non-existent concept is obviously not valid, and hence, should not exist in the mapping list. In the sub-ontology, the concept to be deleted is concept 17751009, while the concept to be merged is concept 263513008. In Figure 6, those two concepts are listed in the binding list between the archetype terms and the sub-ontology concepts. However, in Figure 7, these two concepts are not listed in the binding list. This is correct since the two concepts do not exist in the current sub-ontology. The terms previously bound to the two concepts are now binding to other concepts which exist in the sub-ontology. This shows that the proposed method is able to keep the application terms referring to the valid concepts in the current sub-ontology.

Possibility of application change due to sub-ontology change

The changes to the sub-ontology suggest that the knowledge has changed as well. For an application which has a very high requirement for knowledge update, the changes in the sub-ontology can be interpreted as an indication that the application needs to be updated. For instance, if a concept referred to by an application term is deleted from the sub-ontology, it may be the case that the term should be deleted from the application due to its obsoleteness. The mapper can give notification to the applications with regard to the changes, and it is the decision of the applications to update the terms included in them. If the terms are updated, the sub-ontology must be updated too because the selected concepts which are referred to by the application terms might be changed. This leads to another process of sub-ontology changes.

VI. Conclusion

In this paper, a change propagation mechanism from an ontology to the depending application has been proposed. A mechanism which is able to manage the continuous access of the applications to the ontology they refer to is proposed. Using the mechanism, the applications keep referring to the up-to-date ontology even when the ontology changes.The heart of the mechanism is the component called the mapper, which includes a mapping list. The mapping list contains mapping entries, each of which represent the mapping between an application term and an ontology concept. The task of the mapper is to manage the mapping list in the event of ontology changes so that the reference to a non-existing concept is avoided and the quality of the mapping entries can be improved by proposing new mapping entries which contain more relevant pairs of application terms and ontology concepts. Some rules,

which are based on the ontology change operations, have been created for the mapper to update the mapping list.

The proposed method has been applied to an application together with its referred ontology. An archetype is used to represent an application, while an ontology has been developed based on the SNOMED CT ontology and the archetype terms. It is shown that the mapper offers more efficient change propagation than the commonly used method in terms of the number of change operations which should be propagated to the applications. The use of the mapper also enables semi-automatic updating to the mapping list which is obviously faster than manual inspection. Moreover, the rules used by the mapper can maintain the semantic currency of the mapping because the mapper can propose a new mapping entry in which the application term is semantically more similar to the new concept than the previous one.

For future work, the application of the method in other domains, such as bioinformatics and Internet of Things, can be observed. Similar to the application of the method in health domain, the application to other domains requires the availability of a standardised ontology, the distributed environment nature, and the reference of application terms to ontology concepts. Technical aspects of the application of the approach may also be interesting for future work. The performance, reliability and scalability of the deployment of the approach in distributed environment needs to be examined.

REFERENCES

[1] D. E. Oliver and Y. Shahar, "Change management of shared and local health-care terminologies," Methods of Information Medicine, vol. 39, pp. 278–290, 2000.

[2] D. E. Oliver, Y. Shahar, E. H. Shortliffe, and M. A. Musen, "Representation of change in controlled medical terminologies," Artificial Intelligence in Medicine, vol. 15, pp. 53–76, 1999.

[3] N. Stojanovic, L. Stojanovic, and S. Handschuh, "Evolution in the ontology-based knowledge management systems," in Proceedings of the 10th European Conference on Information Systems, Gdansk, Poland, 2002, pp. 840–850.

[4] M. Klein, D. Fensel, A. Kiryakov, and D. Ognyanov, "Ontology versioning and change detection on the web," in Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), LNAI 2473, A. Gomez-Perez and V. Benjamins, Eds. Springer-Verlag, Berlin, Heidelberg, 2002, pp. 197–212.

[5] L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic, "User-driven ontology evolution management," in Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, ser. EKAW '02. London, UK: Springer-Verlag, 2002, pp. 285–300.

[6] A. Maedche, B. Motik, L. Stojanovic, R. Studer, and R. Volz, "Managing multiple ontologies and ontology evolution in ontologging," in Intelligent Information Processing. Kluwer, 2002, pp. 51–63.

[7] N. F. Noy, A. Chugh, W. Liu, and M. A. Musen, "A framework for ontology evolution in collaborative environments," in Proceedings of the 5th International Semantic Web Conference, ser. LNCS Volume 4273. Springer, 2006, pp. 544–558.

[8] F. Zablith, "Dynamic ontology evolution," in International Semantic Web Conference (ISWC) Doctoral Consortium, Karlsruhe, Germany, 2008.

[9] F. Zablith, M. Sabou, M. d'Aquin, and E. Motta, "Ontology evolution with evolva," in Proceedings of the 6th European Semantic Web Conference (ESWC) LNCS 5554. Springer-Verlag, Berlin, Heidelberg, 2009, pp. 908–912.

[10] T. Kirsten, A. Gross, M. Hartung, and R. Erhard, "Gomma: a component-based infrastructure for managing and analyzing life science ontologies and their evolution," Journal of Biomedical Semantics, vol. 2, 2011.

[11] M. Hartung, A. Grob, and E. Rahm, "Conto-diff: generation of complex evolution mappings for life science ontologies," Journal of Biomedical Informatics, vol. 46 (2013), pp. 15–32, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.jbi.2012.04.009

[12] A. M. Khattak, K. Latif, S. Khan, and N. Ahmed, "Managing change history in web ontologies," in Proceedings of the Fourth International Conference on Semantics, Knowledge and Grid, China, 2008.

[13] A. M. Khattak, Z. Pervez, S. Lee, and Y.-K. Lee, "After effects of ontology evolution," in Proceedings of the 5th International Conference on Future Information Technology (FutureTech), 2010, pp. 1 – 6.

[14] A. M. Khattak, K. Latif, and S. Lee, "Change management in evolving web ontologies," Tsinghua Science and Technology, vol. 37 (2013), pp. 1–16, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.knosys.2012.05.005

[15] J. Mapoles, C. Smith, J. Cook, and B. Levy, "Strategies for updating terminology mappings and subsets using snomed ct," in Proceedings of the 3rd international conference on Knowledge Representation in Medicine, R. Cornet and K. Spackman, Eds., 2008.

[16] A. Shaban-Nejad and V. Haarslev, "Bio-medical ontologies maintenance and change management," in Biomedical Data and Applications, ser. Studies in Computational Intelligence Volume 224, A. Sidhu and T. Dillon, Eds. Springer, 2009, pp. 143–168.

[17] R. Palma, O. Corcho, A. Gmez-Prez, and P. Haase, "A holistic approach to collaborative ontology development based on change management," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 9, pp. 299–314, September 2011.

[18] A. K. Sari and W. Rahayu, "A methodology for change propagation in health ontology," in Proceedings of the 15th Pacific Asia Conference on Information Systems, Brisbane, Australia, 2011.

[19] A. K. Sari, W. Rahayu, and M. Bhatt, "An approach for sub-ontology evolution in a distributed health care enterprise," Information Systems, vol. 38, pp. 727–744, July 2013.

[20] L. Stojanovic, A. Abecker, N. Stojanovic, and R. Studer, "On managing changes in the ontology-based e-government," in Proceedings of the 3rd International Conference on Ontologies, Databases and Application of Semantics (ODBASE 2004), ser. LNCS Volume 3291. Springer Verlag, 2004, pp. 1080–1097.

[21] Z. Huang and H. Stuckenschmidt, "Reasoning with multi-version ontologies: A temporal logic approach," in Proceedings of the 4th International Semantic Web Conference (ISWC), 2005, pp. 398–412.

[22] Y. Liang, H. Alani, D. Dupplaw, and N. Shadbolt, "An approach to cope with ontology changes for ontology-based applications," in Second Advanced Knowledge Technologies DTA Symposium, Aberdeen, Scotland, 2006.

[23] Y. Liang, H. Alani, and N. Shadbolt, "Changing ontology breaks the queries," in Doctoral Symposium of The 5th International Semantic Web Conference, ser. LNCS Volume 4273. Athens, GA, U.S.A: Springer-Verlag, 2006, pp. 982–985.

[24] A. C. Yu and J. J. Cimino, "A comparison of two methods for retrieving icd-9-cm data: The effect of using an ontology-based method for handling terminology changes," Journal of Biomedical Informatics, vol. 44, pp. 289–298, 2011.

[25] H. Gu, J. Geller, L.-m. Liu, and M. Halper, "Using a similarity measurement to partition a vocabulary of medical concepts," in Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA '99). London, UK: Springer-Verlag, 1999, pp. 712–723.

[26] H. Leslie and S. Heard, "Archetypes 101," in HIC 2006, J. Westbrook and J. Callen, Eds. Sydney: Health Informatics Society of Australia Ltd (HISA), 2006.

[27] A. K. Sari, W. Rahayu, and M. Bhatt, "Archetype sub-ontology: Improving constraint-based clinical knowledge model in electronic health records," Knowledge Based Systems, vol. 26, pp. 75–85, February 2012.