# Identification and Formal Representation of Change Operations in LOINC Evolution

Anny Kartika Sari
Department of Computer Science and Electronics
Universitas Gadjah Mada
Yogyakarta, Indonesia

***Abstract*—LOINC (Logical Observation Identifiers Names and Codes) is one of the standardized health ontologies that is widely used by practitioners in the health sector. Like other ontologies in health field, LOINC evolves. This research focuses on representing formally the conceptual changes in LOINC. Four steps are taken to achieve this goal. First, the release of LOINC is studied to get an overview of the changes in LOINC. Secondly, the change operations that occur in LOINC are classified. Third, the changes are represented formally by considering the need to keep the history of changes in concepts. Finally, a few algorithms are developed to identify changes that occur between two releases of LOINC. The evaluation shows that the algorithms are able to identify change operations in LOINC with 100% of success rate. With a formal representation of changes that occur in LOINC, it is expected that adjustments to applications that use LOINC can be performed more straightforward. The history of reference to concepts in LOINC can also be traced back so that information about the changes on the reference can be obtained easily.**

***Keywords*—LOINC; ontology; evolution; change operation; formal representation; health**

## I. Introduction

At the moment, ontology is used in many areas. Ontology is the representation of knowledge in a certain domain of interest. Using ontology, knowledge can be represented formally to support the processes used in the applications of the domain.

Health is one of the areas that uses ontology intensively. Ontology in health is also referred to as terminology. In this field, there are several standardized ontologies. The development, usage and distribution of each ontology is standardized by a specific institution. The main content of each ontology is also different, specifically addressed the target of the ontology.

LOINC (Logical Observation Identifiers Names and Codes) is one of the standardized ontologies in health that is managed by Regenstrief Institute, Inc. LOINC has been used for many applications. Firstly, LOINC can be used for universal standard to identify medical laboratory observation to achieve the semantic uniformity in the observation. For instance, the terms used in health archetype, which is discussed in [1] to achieve semantic interoperability of Electronic Health Records, can refer to LOINC or other terminologies in health domain[1]. This way, if different terms are used in archetype built by other health providers, the system can directly look up the referred terminology concepts to find the meaning of the corresponding terms. Hence, semantic interoperability of the terms used by different providers can be achieved. In [2], LOINC is used as standardized terminologies. In the paper, the definition of the term cephalometric (standardized measurement from the angle and distance between specific landmarks of X-ray film that is used for orthodontic treatment planning and varied research applications) that originates from 10 different standards can be unified into single terminology taken from LOINC.

Another use of LOINC is in information retrieval applications. The data in this system is usually organized as a document [3]. In addition to ordinary documents, in a Web context, HTML pages are considered as documents too. Users in the information retrieval application need a specific document or several documents that match their needs. A collection of keywords is a tool that can represent documents desired by users. The user enters the keywords, then the system will provide output in the form of documents that correspond to the keywords. In this case, each collection of documents in the system will be annotated first by words or phrases that can describe the contents of the document. The concepts that exist in LOINC can be used both as keywords and annotations that will be embedded in each document.

The health sector is a field where knowledge develops rapidly. Research in the field of health will always provide updates on existing knowledge. Changes to knowledge result in changes in ontology. This change is referred to as ontology evolution. According to [4], the ontology evolution is a modification process on ontology to accommodate new information on the domain knowledge. Ontology must always be updated to represent current knowledge, otherwise ontology becomes outdated. Changes to the ontology will eventually cause changes to the applications that utilize the ontology.

As one of the ontologies in the health field, LOINC also experienced the same evolution. Every month, there is a new release of LOINC that contains changes to it. These changes will affect all applications that refer to LOINC concepts. For example, annotations on documents based on LOINC concept must also be adjusted so that the referenced concepts still exist in the latest version of LOINC, not the old concepts that might have been deleted because of changes to LOINC.

As LOINC continues to develop, there is a possibility that changes to a concept do not only happen once during its

[1] https://specifications.openehr.org/releases/RM/Release-1.0.4/support.html#_terminology_package

lifetime. If this happens, applications that use LOINC as a reference should keep a history of changes that occur in the corresponding concept. In this way, if a backward trace is needed about why a reference changes, the application can look at the history of the concepts it refers to. This is possible if the evolution of the concepts in LOINC is formally represented.

Unfortunately, there has been no research discussing the formalization of evolution in LOINC. There have been several studies that discuss the formalization of ontology, e.g. [5-7]. However, none has focused on formalization of evolution in LOINC. The absence of this formalization will lead to the difficulty of adjustments that must be made to the application when there is a change in a LOINC concept. So far, there are no guidelines for making these adjustments. Institutions that use LOINC must adjust individually. For institutions that have been using LOINC for a long time, this may not be a problem because they are already experienced in doing so, but for institutions that use LOINC only currently, this could be an unexpected problem. In addition, it will be difficult to backtrack the changes if the history of the changes is not maintained. This is supported by the fact that the latest version of LOINC only mentions changes that occur in that version compared to the previous version. Hence, the changes that occur in previous versions are not contained in the latest version.

This research focuses on the formal representation of changes in the evolution of LOINC. This representation will be useful to understand the changes in LOINC concepts. In addition, with this formal representation, the history of reference concepts in LOINC can be traced back. Thus, information about changes in references can be stored by the applications that use LOINC. Furthermore, the representation of LOINC changes is used to develop algorithms that can identify changes that occur between two versions of LOINC.

The rest of the paper is organized as follows. Chapter II discusses related work, followed by formal representation of LOINC in Chapter III. Classification of change operations is presented in Chapter IV, while Chapter V discusses the formal representation of change operations. Chapter VI addresses the method to maintain the history of the changes. Algorithms to identify change operations are described in Chapter VII. This is followed by Chapter VIII that includes Evaluation and Discussion. Chapter IX concludes this paper.

## II. Related Work

According to [8], there are 6 phases in ontology evolution which include change capturing, change representation, semantics of change, change implementation, change propagation and change validation. The process is simplified in [9] by dividing it into 3 phases, namely change representation ontology, ontology change manipulation, and ontology change propagation. In terms of the ontology domain, the study in [10] discusses existing work in biomedical evolution in detail. However, this research focuses on one phase only, namely change representation ontology.

Ontology change representation is a phase on ontology evolution that discusses how to represent changes in ontology. Change representation should be done in a formal way so that changes can be manipulated and propagated to the application that uses the ontology. Several studies related to the representation of changes in ontology are summarized as follows.

In [11] a formal method called RLR (Represent, Legitimate and Reproduce) is presented. The method is proposed to analyze and support evolution management and changes in ontology in the biomedical field. The focus of this research is on the phase of representation. To represent changes, Description Logic (DL) is used. The method is based on a discrete state model and uses category theory for representation with diagrams. This method is applied to Fungal Web Ontology which is a formal ontology on the fungal domain genomics.

A representation scheme called CHO (Change History Ontology) is defined in [6]. There are two basic elements of CHO, namely the OntologyChange class and the ChangeSet class. The OntologyChange class contains a sub-class called Atomic-Change, which represents all classes, properties, individuals and constraints at the atomic level. The ChangeSet class is responsible for managing changes to ontologies and arranging them in time-indexed form.

The classification of change operations in health ontologies is discussed in [7]. In the research, change operations are classified into 2 types. The first type is basic operation which is only based on a list of changes that are already available in release ontology. The second type is semantic operation, which is an operation that has a complete definition that can be a combination of one or more basic operations. Both operations are represented formally with mathematical representations. In addition, algorithms are also prepared to identify semantic operations based on a list of basic operations.

Several studies on LOINC discuss the use of LOINC. In [11], a method is developed to evaluate the consistency and utilization of LOINC in different institutions, and to evaluate the level of interoperability that can be achieved by using LOINC as a standard code for data exchange. There are variations in the use of LOINC in data exchange, which shows that the interoperability between different institutions does not fully exist. To improve semantic interoperability, identification and correction of knowledge that is contradictory to LOINC is needed.

Research in [12] attempts to overcome difficulties in achieving semantic interoperability because of the use of different languages. In this study LOINC is translated into Italian. In addition, a tool is built that can find a unique list of LOINC Parts from a given set of LOINC terms.

Research on LOINC in [2], [13], [14] and [15] address the use of LOINC as a way to achieve semantic interoperability and its application in several case studies. In [2] the definition of the term cephalometric using LOINC terminology is brought to overcome differences in terms from 10 different standards. The case study of LOINC application for documents in hospital information systems is reviewed in [13], while a pilot project to standardize local laboratory data on Indian Health Service (IHS) medical facilities is conducted in [14] by mapping names of laboratory test into terms in LOINC. In [15], a process of "enhancing" local test names is developed by incorporating information required for LOINC mapping into the test names themselves.

None of the above studies specifically addresses the change operations in LOINC. In fact, the structure of LOINC can be considered as unique because it is not the same as the structure of general ontologies. The essence of ontology is concept, whereas in LOINC there are 6 fields which describe a particular term. Thus, we need a special representation for LOINC that can be used as a basis for the formal representations of concept changes in LOINC. This research focuses on the formal representation of LOINC and the representation of change operations that occur in LOINC concepts.

## III. Formal Representation of LOINC

There are 2 fundamental differences between LOINC and the formal definition of ontology in general, as follows:

*1)* In LOINC and other ontologies in health field, there is no definition of instance. This is different from the general definition of an ontology that include instance as a component of the ontology. For example, the definition of ontology in Davis et al. [16] and [17] includes instances as elements of ontology. In [18] it is suggested that in SNOMED CT, as one of the ontologies in health field, instance of a concept is one of the three possible entities: a stand-alone object without clinical context, artifact contained in the patient's electronic medical record, or the patient himself or clinical situation. These three elements are not found in LOINC because LOINC only contains concepts. The real object of the concept is not defined in LOINC.

*2)* LOINC represents concepts into 6 dimensions, which together provide an overview of a concept. Concept definition can be easily equated with concept definition in ontology. However, the six dimensions cannot be compared to object property or data property. Thus, a special representation is needed for the six dimensions of the LOINC concept.

Based on the above reasons, in this study, a formal definition for LOINC will be carried out specifically, which adopts the definition of ontology in the health field in [7] with adjustments to LOINC characteristics. The following is a formal definition of LOINC.

Definition 1. LOINC Ontology

$O_L \equiv \langle C, Co, P, T, Sy, Sc, M, R \rangle$ is LOINC ontology with:

- C: set of Concepts, referring to LOINC concepts.
- Co: set of Components
- P: set of Properties
- T: set of Times
- Sy: set of Systems (Specimens)
- Sc: set of Scales
- M: set Methods
- R: set of relationship that connects Concept with one of dimensions, that is, Component, Property, Time, System, Scale, or Method. To form a relationship r = (c, rel, d), the set of relationship type Rel is defined. In this case, Rel = {rco, rp, rt, rsy, rsc, rm} is a set of relationship types, containing various types of relationships between Concept and one of the dimensions of the LOINC concept. There are 6 members of Rel, namely rco (relation between concept and component), rp (relation between concept and property), rt (relation between concept and time), rsy (relation between concept and system), rsc (relation between concept and scale) ), and rm (relation between concept and method).

In ontology $O_L$, the following constraints must be met.

- $\forall r \in R$: r = (c, rel, d), with $c \in C$, $rel \in Rel$, and $d \in \{Co \cup P \cup T \cup Sy \cup Sc \cup M\}$ (1)
- $\forall r \in R$: rel = rco $\rightarrow$ d $\in$ Co (2)
- $\forall r \in R$: rel = rt $\rightarrow$ d $\in$ P (3)
- $\forall r \in R$: rel = rp $\rightarrow$ d $\in$ T (4)
- $\forall r \in R$: rel = rsy $\rightarrow$ d $\in$ Sy (5)
- $\forall r \in R$: rel = rsc $\rightarrow$ d $\in$ Sc (6)
- $\forall r \in R$: rel = rm $\rightarrow$ d $\in$ M (7)

In the definition above, LOINC ontology consists of 8 tuples, namely the set of concepts C, the set of Component Co, the set of Property P, the set of Time T, the set of System Sy, the set of Scale Sc, the set of Method M, and the set of relationship R. In constraint (1), the definition of a relationship r is a tuple (c, rel, d), where rel is the type of relationship between c and d. Constraints (2), (3), (4), (5), (6), (7) specify the type of d.

The ontology concepts in LOINC sometimes require more detailed information about a concept. In this study, the information referred to as attributes. The formal definition of attributes is as follows, adopted from [7] with some adjustments.

Definition 2. Set of concept attributes

$A_k \equiv \{a_{x1}(c) = v_1, a_{x2}(c) = v_2, ..., a_{xn}(c) = v_n\}$ is the set of attributes for concept c in ontology $O_L$ with $x_i$ is the atribut name and $v_i$ is the attribute value for c. Concept c is a concept that is included in set C.

LOINC has determined additional information that can be attributed to a concept. In this study, the information is represented as an attribute of the concept. The following are some of the concept attributes that are part of the definition of a concept in the LOINC ontology.

- name: the name of the concept.

- code: the code of the concept.
- class: the class of a concept, which can be selected from the choice of existing classes.
- classtype: the class type of the concept, with values from 1 to 4, where 1 = Laboratory class; 2 = Clinical class; 3 = Claims attachments; and 4 = Surveys.
- long_common_name: the long common name of the concept.
- short_name: the short name of the concept.
- status: indicates the status of the concept, whether ACTIVE or INACTIVE.
- version_first: the version of LOINC where the concept was first included in ontology.
- version_last_changed: the LOINC version where the concept is last changed.

## IV. Classification of Change Operations in LOINC

Change operations in LOINC can be divided into 3, namely additions, updates, and deletions. Each type of operation can be applied to different entities such as concepts, dimensions and relationships. The following is the detail of the change operations that are included in LOINC.

### A. *Addition Operation*

Addition operations can be performed on the set of concept, property, time, system, scale, method, and relationship.

- Addition to the concept set: Addition to the concept set is carried out if there is a new concept included in LOINC ontology.
- Addition to the component set: Addition to the component set is done if there is a new component included in LOINC ontology.
- Addition to the property set: Addition to property set is performed if there is a new type of property included in LOINC ontology, which might be a value from the property dimension of a concept.
- Addition to the time set: Addition to time set is carried out if there is a new type of interval time included in LOINC ontology, which might be a value from the time dimension of a concept.
- Addition to the system set: Addition to system set is carried out if there is a new system included in LOINC ontology, which might be a value from the system dimension of a concept.
- Addition to the scale set: Addition to scale set is carried out if there is a new scale type included in LOINC ontology, which might be a value from the scale dimension of a concept.
- Addition to the method set: Addition to method set is carried out if there is a new measurement method included in LOINC ontology, which might be a value from the method dimension of a concept.
- Addition to the relationship set: Addition to relationship set is carried out if there is a new relationship included in LOINC ontology, which connects a concept with one of the six dimensions. Addition to relationship set will definitely occur if the new concept included in ontology.

### B. *Update Operation*

Update changes can be made on the set of concepts and relationships.

- Update to a concept: Update to a concept is carried out if there is a change to the concept attribute, e.g. name, code, long common name, short name, status, and version last changed. Update to other attributes do not occur.
- Update to a relationship: This operation accommodates update to the value of concept dimension. Change can occur in the value of d of the relationship r = (c, rails, d). An example of this change is a change in the method of a concept, for instance from the original value Observed then updated to Reported.

### C. *Deletion Operation*

Deletion operations can be performed on the set of property, time, system, scale, and method. However, deletion is very rare. The deletion operation is not performed on concepts because in LOINC, a concept is never erased. If a concept is not used anymore, the status of the concept is set to INACTIVE. Deletion operation is not performed to relationship either, because a relationship connects a concept to its dimensions. If a concept has the status of INACTIVE, the relationship to its dimensions still exists.

Details of each deletion operation is as follows.

- Deletion of a component: Deletion of a component is done if there is a type of component that is not used anymore in LOINC ontology.
- Deletion of a property: Deletion of a property is done if there is a type of property that is not used anymore in LOINC ontology.
- Deletion of a time: Deletion of a time is carried out if there is a type of time interval excluded from LOINC ontology.
- Deletion of a system: Deletion of a system is carried out if there is a system type that is excluded from LOINC ontology.
- Deletion of a scale: Deletion of a scale is carried out if there is a type of scale that is no longer used in LOINC ontology.
- Deletion of a method: Deletion of the method is carried out if there is a measurement method that is not used in LOINC ontology.

## V. Formal Representation of Change Operations in LOINC

Update on a LOINC entities produces a change operation on the ontology. In this section, the formal definition of changed ontology and change operation in LOINC are presented before the discussion of each type of change operation. The definitions are adopted from [7] with some adjustments.

**Definition 3. Changed ontology**

Given ontology $O_L \equiv <C, Co, P, T, Sy, Sc, M, R>$.

$O_L'$ is the changed ontology to $O_L$ with $O_L' \equiv <C', Co', P', T', Sy', Sc', M', R'>$, C′ is the changed set of concepts, Co′ is the changed set of components, P' is the changed set of properties, T' is the changed set of time, Sy' is the changed set of systems, Sc' is the changed set of scales, M' is the changed set of methods, and R' is the changed set of relationships.

**Definition 4. Ontology change operation**

Given ontology $O_L \equiv <C, Co, P, T, Sy, Sc, M, R>$.

$O_p$ is a change operation in ontology $O_L$ such that if $O_p$ is executed then $((C' \leftarrow C) \vee ((Co' \leftarrow Co) \vee (P' \leftarrow P) \vee (T' \leftarrow T) \vee (Sy' \leftarrow Sy) \vee (Sc' \leftarrow Sc) \vee (M' \leftarrow M) \vee (R' \leftarrow R))$.

From the two definitions above, it can be concluded that ontology changes are caused by the existence of at least one of ontology change operations. Ontology change operations can be applied to each ontology entities. However, the type of operation for each entity is different. The type and representation of change operations in LOINC ontology is described as follows.

### *D. Operations on Concepts*

In LOINC, concepts can be added or changed. Based on this, 8 types of operations to ontology concept are defined as follows. Note that each operation will result in the change of concept c, which means that the set of concepts C will also change.

*1) Concept addition (AddConcept)*: The concept addition operation is an operation carried out to incorporate a new concept in LOINC ontology. In other words, the new concept is added to the set of concepts C. The formal definition of concept addition operation is as follows.

**Definition 5. AddConcept operation**

$AddConcept(c_{new}, O_L) \Leftrightarrow O_L \mid C \leftarrow C \cup \{c_{new}\}$

*2) Concept renaming (UpdConceptName)*: The concept renaming operation is an operation performed to change the name of the concept to LOINC ontology. In this way, the value of the name attributes changes. The formal definition of the concept renaming operation is as follows.

**Definition 6. UpdConceptName operation**

$UpdConceptName(c, name_{new}, O_L) \Leftrightarrow C \mid name(c) \leftarrow name_{new}$

*3) Update concept's code (UpdConceptCode)*: The operation of changing the concept code is an operation carried out to change the value of concept code in LOINC ontology. The value of the code attribute will change. The formal definition of concept code change operation is as follows.

**Definition 7. UpdConceptCode operation**

$UpdConceptCode (c, code_{new}, O_L) \Leftrightarrow C \mid code (c) \leftarrow code_{new}$

*4) Update concept's long common name (UpdConceptLcn)*: This operation changes the value of a concept's long common name in LOINC ontology. The formal definition of the operation is as follows.

**Definition 8. UpdConceptLcn operation**

$UpdConceptLcn (c, lcn_{new}, O_L) \Leftrightarrow C \mid lcn(c) \leftarrow lcn_{new}$

*5) Update concept's short name (UpdConceptSn)*: This operation changes a concept's short name in LOINC ontology. Hence, the value of the short name attribute will be changed. The formal definition of the operation is as follows.

**Definition 9. UpdConceptSn Operation**

$UpdConceptSn (c, lsn_{new}, O_L) \Leftrightarrow C \mid sn(c) \leftarrow sn_{new}$

*6) Update concept's status (UpdConceptStatus)*: This operation will update the status of a concept in LOINC ontology. This means that the value of status attribute changes. The formal definitions of the operation is as follows.

**Definition 10. UpdConceptStatus Operation**

$UpdConceptStatus (c, status_{new}, O_L) \Leftrightarrow C \mid status(c) \leftarrow status_{new}$

*7) Update concept's version last changed (UpdConceptVersion)*: This operation will update the version in which the concept is last changed. In other words, the attribute value of version last changed will change. The formal definition of the operation is as follows.

**Definition 11. UpdConceptVersion Operation**

$UpdConceptLcn (c, version_{new}, O_L) \Leftrightarrow C \mid version(c) \leftarrow version_{new}$

8. *Update concept's class (UpdConceptClass)*: This operation is to update the class of a concept in LOINC ontology, which means that the class attribute values is changed. The formal definition of the operation is as follows.

**Definition 12. UpdConceptClass operation**

$UpdConceptClass (c, class_{new}, O_L) \Leftrightarrow C \mid class(c) \leftarrow class_{new}$

### *E. Operations on the Dimensions of LOINC*

In LOINC, the value of each of the 6 dimensions, i.e. component, property, time, system, scale and method, can be added or removed. However, change operations to the dimensions are very rare. Nevertheless, the operations need to be defined formally. Since the change operations applied to the dimensions are very similar, only operations to component dimension are presented here. The formal definition of operations to other dimensions is basically the same as the operations to component.

*1) Component addition (AddComponent)*: This operation will add a new component to LOINC ontology. The new component is included in the set of components Co. The formal definition of the operation is as follows.

**Definition 13. AddComponent operation**

$AddComponent (co_{new}, O_L) \Leftrightarrow O_L \mid Co \leftarrow Co \cup \{co_{new}\}$

*2) Component deletion (DelComponent)*: This operation is to remove a component from LOINC ontology. This means that the component is removed from the set of components Co. The formal definition of the operation is as follows.

**Definition 14. DelComponent operation**

DelComponent ($co_{del}$, $O_L$) $\Leftrightarrow O_L \mid Co \leftarrow Co - \{co_{del}\}$

*F. Operations on Relationships*

In LOINC, relationships on ontologies can be added to ontology or changed. Based on this, 2 types of change operations for relationship are defined as follows.

*1) Add relationship (AddRelationship)*: Addition operation is an operation performed to include a new relationship to LOINC ontology. This means that the new relationship is added to the relationship set R. The formal definition of this operation is as follows.

**Definition 15. AddRelationship Operations**

AddRelationship ($r_{new}$, $O_L$) $\Leftrightarrow O_L \mid R \leftarrow R \cup \{r_{new}\}$

*2) Update relationship (UpdRelationship)*: This operation is carried out to update the relationship in LOINC ontology. In this case, the value of d in (c, rel, d) is changed so that it refers to another value. The formal definition of this operation is as follows.

**Definition 16. UpdRelationship operation**

UpdRelationship (r, $d_{new}$, $O_L$) $\Leftrightarrow$ C | (c, rel, d) $\leftarrow$ (c, rel, $d_{new}$)

## VI. Representation of Versioning for LOINC

Ontology versioning is the ability to manage changes in ontology by making or managing different versions of the ontology taken at different times [19]. In this study, versioning of LOINC needs to be done because LOINC releases are always different from time to time.

To represent the versioning of LOINC, a log file $O_v$ is built. This log file contains a collection of records, each of which stores an operation that occurs in LOINC. In this research, a record is represented as an XML element. Each record contains information about a change operation. The attributes in each record are as follows:

- <id>: indicates the id number of the operation.
- <def>: contains a formal definition of the operation, which is one of the operation definitions presented in Section V. This attribute does not only store the name of the operation, but the arguments of the operations are also kept. Hence, the information about the operation, including the concept that is manipulated and details of the changes, is recorded well.
- <version>: indicates the current version of LOINC in which the change operation occurs.
- <id_prev>: indicates the id number of the previous change operation that was applied to the same entity, either a concept, a dimension or a relationship.

The formal definition of adding a change operation to the log file is as follows.

**Definition 17. Append operation (Append)**

Append($O_v$, <id, def, version, id_prev>)

As previously mentioned, the information contained in the log file is the id of the operation, the formal definition of the operation, the current version of LOINC in which the change occurs, and the id of the previous operation performed on the same entity. The <id_prev> attribute can be used to trace the history of changes to a particular entity, including a concept. Thus, if the referenced concept in an application is different from time to time, the record in the log file can be used to identify the reason of the differences.

## VII. Algorithms to Identify Change Operations

Each LOINC release includes several files related to the changes that occur in LOINC. Identification of change operation can be conducted by checking the entries in each file. Identification is needed so that change operations can be found easily. This section gives detailed description of the files that can be used to identify operations and the algorithms to identify the operations.

The main file related to changes in LOINC is the LOINC_updates file. This file lists the changes that occur in that particular LOINC version. Each record in the list consists of the following columns:

- RecType, is the column that contains the type of change that occurs in a LOINC concept. In this column, there are only 3 types of values, namely BEFORE, CHANGED and ADD. The record that contains BEFORE is paired with the record that contains CHANGED, which means that the two records represent the concept before and after it is changed. The record that contains ADD shows that a concept is added to LOINC ontology.
- LOINC_NUM, is a column that shows the code of the concept that is changed or added.
- COMPONENT, is a column that shows the component value of the concept.
- PROPERTY, is a column that shows the property value of the concept.
- TIME_ASPCT, is a column that shows the time value of the concept.
- SYSTEM, is a column that shows the system value of the concept.
- SCALE_TYP, is a column that shows the scale value of the concept.
- METHOD_TYP, is a column that shows the method value of the concept.
- CLASS, is a column that shows the class value of the concept.

In the list, a change is indicated by the BEFORE and CHANGED record pairs for a particular concept. Changing to dimensions is indicated by the differences between the values of the corresponding dimension column for the pair of records. For example, one of the concepts included in file LOINC_2.52_2.54_Updates.csv is the concept with LOINC_NUM 10232-7. The value of the SYSTEM dimension in the BEFORE record is Aortic root, while the SYSTEM dimension value in the AFTER record is Aorta.root. Based on this information, a change operation has taken place. In this case, the change operation that occurs is UpdRelationship. Originally concept 10232-7 relates to system Aortic root with relationship type rsy, while in the new release, the concept relates to system Aorta.root. Thus, the formal definition of the operation is UpdRelationship((10232-7, rsy, Aortic root), Aorta.root, $O_L$).

RecType with ADD value indicates the addition of a new concept into LOINC. Each dimension value of the new concept is set accordingly, as shown in the entry of the corresponding dimension column. Hence, when a new concept is added, there are 6 AddRelationship operations that must be defined to represent the relationships between the new concept and each dimension. Furthermore, the UpdConceptClass operation is also defined to set the Class attribute of the new concept. For example, the value of each column in one of the records in file LOINC_2.52_2.54_Updates.csv is as follows: ADD; 60738-2; Intraluminal; Pres; Pt; Esophagus; Qn; -; GI. Based on the record, a new concept, i.e. concept 60738-2, is added to the ontology, with Intraluminal as component dimension, Pres as property dimension, Pt as time dimension, Esophagus as system dimension, Qn as scale dimension, no value for method dimension, and GI as class attribute value. Using the definition of change operations in Section V, there are 7 operations exist as follows.

- AddConcept (60738-2, $O_L$), which is an operation to add a new concept with code 60738-2 into ontology.
- AddRelationship ((60738-2, rco, Intraluminal), $O_L$), which is an operation to add a relationship that connects concept 60738-2 with the component dimension of the value Intraluminal. This means that the component of the concept is Intraluminal.
- AddRelationship ((60738-2, rp, Pres), $O_L$)), which is an operation to add a relationship that connects concept 60738-2 with the property dimension of the value Pres. This means that the property of the concept is Pres.
- AddRelationship ((60738-2, rt, Pt), $O_L$)), which is an operation to add a relationship that connects the concept 60738-2 with the time dimension of the value Pt. This means that the time of the concept is Pt.
- AddRelationship ((60738-2, rsy, Esophagus), $O_L$)), which is an operation to add a relationship that connects the concept of 60738-2 with the system dimensions of the value Esophagus. This means that the system of the concept is Esophagus.
- AddRelationship ((60738-2, rsc, Qn), $O_L$)), which is an operation to add a relationship that connects the concept 60738-2 with scale dimensions of the value Qn. This means that the scale of the concept is Qn.
- UpdConceptClass (60738-2, GI, $O_L$)), which is an operation to set the value of the class attribute value of the concept to GI.

In this section, 3 algorithms are presented, i.e. the algorithm to identify the type of change operation, the algorithm to define relationship operations, and the algorithm to define concept addition operations. The algorithms are presented in Fig. 1, Fig. 2, and Fig. 3, respectively. These algorithms are based on the fact that there are only 7 dimensions or attributes associated with the concept in the LOINC_Updates file. The related operations are UpdRelationship (operation to change relationship), AddConcept (operation to add new concept to LOINC), and UpdConceptClass (operation to update the value of class attribute).

The first algorithm shown in Fig. 1 shows the steps to identify the type of change operation that occurs based on the entry (record) in the LOINC_Updates file. If the value of the Rectype is BEFORE, it means there is a change in the value of dimensions or concept attributes, hence, the Update procedure is called with arguments: LOINC_NUM (concept code), BEFORE (the first of the pair record with the Rectype value of BEFORE), and CHANGED (the second of the pair record with the Rectype value of CHANGED, i.e. the record next after the BEFORE record). If the value of the RecType is ADD, which means that there is an addition of a concept, the Add procedure is called with arguments: LOINC_NUM (concept code) and ADD (the corresponding record with the Rectype value of ADD). The result of this algorithm is the identification of the types of operations that occur in a concept.

Fig. 2 shows the algorithm that is used to define operations related to relationship change. This algorithm is called if there is a pair of records with the RecType values of BEFORE and CHANGED found in the LOINC_Updates file. There are 7 possible operations that can be identified and defined, which consist of 6 UpdRelationship operations, each of which is for different concept dimension, and 1 UpdClass operation for the class attribute on the concept. For each operation, an update of the corresponding value is carried out. After that, an entry was added to $O_v$ log that recorded the change operation in the corresponding concept, accompanied by information about the formal definition of the operation, the operation id, the current version of LOINC that contains the change, and the version of LOINC in which the same concept was changed. The result of this algorithm is that all the corresponding operations have been defined, and the change operation records are listed in the $O_v$ log file.

```
ALGORITHM 1: CHANGE OPERATION IDENTIFICATION
IdentifyChange()
{
    if exist(LOINC_NUM) then
        if RecType = BEFORE then
            Update(LOINC_NUM, BEFORE, CHANGED)
        else if RecType = ADD then
            Add(LOINC_NUM, ADD)
        endif
    endif
}
```

Fig. 1. Algorithm to Identify the Type of Operation.

```
ALGORITHM 2: UPDATE RELATIONSHIP OPERATION
Update(LOINC_NUM, BEFORE, CHANGED)
{
if COMPONENT(BEFORE) <> COMPONENT(CHANGED) then
    (LOINC_NUM, rco, COMPONENT(CHANGED)) ← (LOINC_NUM, rco, COMPONENT(BEFORE))
    Append(Ov, <id, UpdRelationship (LOINC_NUM, (LOINC_NUM, rco, COMPONENT(CHANGED)), $O_L$), current version, id_prev>)
endif
if PROPERTY(BEFORE) <> PROPERTY(CHANGED) then
    (LOINC_NUM, rp, PROPERTY(CHANGED)) ← (LOINC_NUM, rp, PROPERTY(BEFORE))
    Append(Ov, <id, UpdRelationship (LOINC_NUM, (LOINC_NUM, rp, PROPERTY(CHANGED)), $O_L$), current version, id_prev>)
endif
if TIME(BEFORE) <> TIME(CHANGED) then
    (LOINC_NUM, rt, TIME(CHANGED)) ← (LOINC_NUM, rt, TIME(BEFORE))
    Append(Ov, <id, UpdRelationship (LOINC_NUM, (LOINC_NUM, rt, TIME(CHANGED)), $O_L$), current version, id_prev>)
endif
if SYSTEM(BEFORE) <> SYSTEM (CHANGED) then
    (LOINC_NUM, rsy, SYSTEM (CHANGED)) ← (LOINC_NUM, rsy, SYSTEM(BEFORE))
    Append(Ov, <id, UpdRelationship (LOINC_NUM, (LOINC_NUM, rsy, SYSTEM (CHANGED)), $O_L$), current version, id_prev>)
endif
if SCALE(BEFORE) <> SCALE (CHANGED) then
    (LOINC_NUM, rsc, SCALE (CHANGED)) ← (LOINC_NUM, rsc, SCALE (BEFORE))
    Append(Ov, <id, UpdRelationship (LOINC_NUM, (LOINC_NUM, rsc, SCALE (CHANGED)), $O_L$), current version, id_prev>)
endif
if METHOD(BEFORE) <> METHOD(CHANGED) then
    (LOINC_NUM, rm, METHOD (CHANGED)) ← (LOINC_NUM, rm, METHOD (BEFORE))
    Append(Ov, <id, UpdRelationship (LOINC_NUM, (LOINC_NUM, rm, METHOD (CHANGED)), $O_L$), current version, id_prev>)
endif
if CLASS(BEFORE) <> CLASS(CHANGED) then
    class(LOINC_NUM) ← CLASS(CHANGED)
    Append(Ov, <id, UpdConceptClass(LOINC_NUM, CLASS(CHANGED), $O_L$), current version, id_prev>)
endif
}
```

Fig. 2. Algorithm to Define Operations on Relationships and Class Attribute.

Fig. 3 shows the algorithm for adding a new concept to the ontology. This algorithm is called if there is a record with the RecType value of ADD found in the LOINC_Updates file. Eight operations are identified when a concept is added, i.e. 1 AddConcept operation, 6 AddRelationship operations, each of which is for one concept dimension, and 1 UpdClass operation to set the class attribute of the concept. For AddConcept operations, a new concept is added to set of concepts C, then an entry is added to the $O_v$ log to record the operation. The AddConcept operation is always followed by 6 AddRelationship operations to define the value for each of the concept dimension. Thus, there are 6 AddRelationship operations, each of which has different relationship type, adjusted with the dimension name. In addition, there is an UpdClass operation to set the value of the class attribute of the concept. For each of these operations, an entry in the log $O_v$ is added to record the operation, accompanied by the information about the formal definition of the operation, the operation id, the current version of LOINC that contains the change. The value of id_prev is set to null since the concept is new in the ontology, hence, there is no previous change applied to the concept. The result of this algorithm is that the AddConcept operation and the operations that accompany it are all defined, while the records corresponding to the change operations are also written to $O_v$ log file.

```
ALGORITHM 3: CONCEPT ADDITION OPERATION
Add(LOINC_NUM, ADD)
{
    Add(LOINC_NUM, C)
    Append(Ov, <id, AddConcept(LOINC_NUM, OL), current version, id_prev>
    Add((LOINC_NUM, rco, COMPONENT(ADD)), R)
    Append(Ov, <id, AddRelationship((LOINC_NUM, rco, COMPONENT(ADD)), OL), current version, id_prev>)
    Add((LOINC_NUM, rp, PROPERTY(ADD)), R)
    Append(Ov, <id, AddRelationship((LOINC_NUM, rp, PROPERTY(ADD)), OL), current version, id_prev>)
    Add((LOINC_NUM, rsy, SYSTEM(ADD)), R)
    Append(Ov, <id, AddRelationship((LOINC_NUM, rsy, SYSTEM(ADD)), OL), current version, id_prev>)
    Add((LOINC_NUM, rsc, SCALE(ADD)), R)
    Append(Ov, <id, AddRelationship((LOINC_NUM, rsc, SCALE(ADD)), OL), current version, id_prev>)
    Add((LOINC_NUM, rt, TIME(ADD)), R)
    Append(Ov, <id, AddRelationship((LOINC_NUM, rt, TIME(ADD)), OL), current version, id_prev>)
    Add((LOINC_NUM, rm, METHOD(ADD)), R)
    Append(Ov, <id, AddRelationship((LOINC_NUM, rm, METHOD(ADD)), OL), current version, id_prev>)
    class(LOINC_NUM) ← CLASS(ADD)
    Append(Ov, <id, UpdConceptClass(LOINC_NUM, CLASS(ADD), OL), current version, id_prev>)
}
```

Fig. 3. Algorithm to Define Concept Addition Operations.

## VIII. Evaluation and Discussion

The three algorithms described in Section VII have been implemented in C++. To evaluate the methods, including the formal definition of the change operations, an evaluation has been carried out by applying them to LOINC Release of June 2017. For this reason, LOINC_2.52_2.54_Updates.csv is used which contains changes that occur in that release. The following is the detailed description of the data contained in the file, along with the calculation of the number of operations that should be identified.

- Number of records: 9001
- Number of record pairs with RecType of BEFORE and CHANGED: 3154

Among the 3154 record pairs, each produces one or more UpdRelationship operations or UpdConceptClass operation. The total number of operations is 3220, with details as follows, which is also the target number of change operation identification:

*a)* 579 UpdRelationship operations with relationship type of rco.

*b)* 176 UpdRelationship operations with relationship type of rp.

*c)* 2067 UpdRelationship operations with relationship type of rt.

*d)* 134 UpdRelationship operations with relationship type of rsy.

*e)* 4 UpdRelationship operations with relationship type of rsc.

*f)* 251 UpdRelationship operations with relationship type of rm.

*g)* 9 UpdConceptClass operations.

- Number of records with RecType of ADD: 2693

Based on Section VII, the target number of operations that must be identified is as follows:

*a)* 2693 AddConcept operation.

*b)* 2693 AddRelationship operations with relationship type of rco.

*c)* 2693 AddRelationship operations with relationship type of rp.

*d)* 2693 AddRelationship operations with relationship type of rsy.

*e)* 2693 AddRelationship operations with relationship type of rsc.

*f)* 2693 AddRelationship operation with relationship type of rt.

*g)* 2693 AddRelationship operations with relationship type of rm.

*h)* 2693 UpdConceptClass operations.

- Total number of update/addition operations: 22071.

Table 1 lists the operation identification results for each of the operation types. From the table, it can be seen that the algorithms can identify operations with a success rate of 100%. Hence, it can be concluded that the algorithms have been compiled correctly and can be used to identify changes to LOINC using the files available in each LOINC release.

Other than the identification of change operations, the algorithms also produce a log file that can show a history of changes to a particular concept. This file can be used to track changes that occur during the life of a concept. Since in this research there is no process of identifying change operations in previous releases (due to data limitations), evaluation to the log files related to the history of a concept cannot be performed. However, log files can still be used in the future because the change operations listed in the log file will accumulate. Hence, the changes in the binding/reference of a term in an application to a LOINC concept can be traced back to the sequence of changes starting from LOINC Release in June 2017.

TABLE I. RESULT OF OPERATION IDENTIFICATION USING THE PROPOSED ALGORITHMS

| **Type of operation** | **The number of operations** | **The number of successful identifications** | **Percentage of successful identification** |
|---|---|---|---|
| UpdRelationship of relationship type rco | 579 | 579 | 100% |
| UpdRelationship of relationship type rp | 176 | 176 | 100% |
| UpdRelationship of relationship type rt | 2067 | 2067 | 100% |
| UpdRelationship of relationship type rsy | 134 | 134 | 100% |
| UpdRelationship of relationship type rsc | 4 | 4 | 100% |
| UpdRelationship of relationship type rm | 251 | 251 | 100% |
| AddClass | 2693 | 2693 | 100% |
| AddRelationship of relationship type rco | 2693 | 2693 | 100% |
| AddRelationship of relationship type rp | 2693 | 2693 | 100% |
| AddRelationship of relationship type rt | 2693 | 2693 | 100% |
| AddRelationship of relationship type rsy | 2693 | 2693 | 100% |
| AddRelationship of relationship type rsc | 2693 | 2693 | 100% |
| AddRelationship of relationship type rm | 2693 | 2693 | 100% |
| UpdConceptClass | 2702 | 2702 | 100% |
| Total operations | | 22071 | |

## IX. CONCLUSION

In this paper, a formal representation of change operations in the evolution of LOINC has been presented. Operations can be classified into 3, namely addition, change/update, and deletion, and each operation type has different target of entities. The classification of change operations is based on the changes that occur in the release of LOINC. In addition, formal representation of change operations is presented. Algorithms to identify each change operation have been implemented using the files related to changes that are included in the release of LOINC.

The evaluation result shows that the algorithm can be used to identify change operations that occur in the LOINC release of June 2017 with 100% success rate. Log files produced from the identification of operation changes has been generated to keep a history of changes that occur in a particular concept. By utilizing this log file, the history of reference to LOINC concepts can also be traced back so that information about reference changes can be obtained easily.

For future work, an ontology can be defined to maintain the change operations that occur in LOINC. Moreover, algorithms for identifying change operations can be completed with operations other than AddConcept, AddRelationship, UpdRelationship, and UpdConceptClass. These algorithms will require the data contained in 2 LOINC versions that are released in sequence. In addition, the domain of the ontology can also be extended to other ontologies related to biomedical field, such as Gene Ontology.

## REFERENCES

[1] S. Garde, R. Chen, H. Leslie, T. Beale, I. McNicoll, and S. Heard, "Archetype-Based Knowledge Management for Semantic Interoperability of Electronic Health Records", Proceedings of MIE 2009: The XXIInd International Congress of the European Federation for Medical Informatics, Sarajevo, Bosnia and Herzegovina, Agust 30 - September 2, 2009.

[2] P. J. Kroth, S. Daneshvari, E. F. Harris, D. J. Vreeman, and H. J. H. Edgar, "Using LOINC to link ten terminology standards to one unified standard in a specialized domain", J. Biomed Inform., vol. 45(4), pp. 674-682, August 2012.

[3] A. Silberschatz, H.F. Korth, and S. Sudarshan, Database System Concepts, 6th ed., McGraw Hill, 2011.

[4] A.M. Khattak, Z. Pervez, S. Lee, and Y. K. Lee, "After Effects of Ontology Evolution", 5th International Conference on Future Information Technology (FutureTech), 2010.

[5] G. Konstantinidis, F. Giorgos, A. Grigoris, and V. Christophides, "A Formal Approach for RDF/S Ontology Evolution", ECAI 2008, IOS Press, pp. 70-74, 2008.

[6] A.M. Khattak, K. Latif, and S. Lee, "Change management in evolving web ontologies", Knowledge-Based Systems, vol. 37, pp. 1–18, 2013.

[7] A.K. Sari, W. Rahayu, and M. Bhatt, "An approach for sub-ontology evolution in a distributed health care enterprise", Information Systems, vol. 38(5), pp. 727-744, 2013.

[8] A. Maedche, B. Motik, and L. Stojanovic, "Managing multiple and distributed ontologies on the semantic web", The VLDB Journal, vol. 12, pp. 286–302, 2003.

[9] 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, 2011.

[10] A. Shaban-Nejad and V. Haarslev, "Bio-medical ontologies maintenance and change management", In: Sidhu A.S., Dillon T.S. (eds) Biomedical Data and Applications, Studies in Computational Intelligence, vol 224. Springer, 2009.

[11] M. C. Lin, D.J. Vreeman, C. J. McDonald, and S. M. Huff, "Auditing consistency and usefulness of LOINC use among three large institutions –Using version spaces for grouping LOINC codes", Journal of Biomedical Informatics, vol. 45 (4), pp. 658-666, 2012.

[12] D.J. Vreeman, M.T. Chiaravalloti, J. Hook, and C. J. McDonald, "Enabling international adoption of LOINC through translation", Journal of Biomedical Informatics, vol. 45 (4) , pp. 667-673, 2012.

[13] M. Dugas, S. Thun, T. Frankewitsch, and K. U. Heitmann, "LOINC® Codes for Hospital Information Systems Documents: A Case Study", Journal of the American Medical Informatics Association, vol. 16 (3), pp. 400-403, 2012.

[14] A. N. Khan, S. P. Griffith, C. Moore, D. Russell, A. C. Rosario Jr., and J. Bertolli, "Standardizing Laboratory Data by Mapping to LOINC", Journal of the American Medical Informatics Association, vol. 13 (3), pp. 353-355, 2006.

[15] H. Kim, R. El-Kareh, A. Goel, F. N. U., Vineet, and W. W. Chapman, "An approach to improve LOINC mapping through augmentation of local test names", Journal of Biomedical Informatics, vol. 45 (4), pp. 651-657, 2012.

[16] J. Davis, R. Studer, and P. Warren, Semantic Web Technologies Trends and Research in Ontology-based Systems, John Wiley & Sons, Ltd, West Sussex, 2006.

[17] M. Ehrig, Ontology Alignment: Bridging the Semantic Gap, Semantic Web and Beyond Computing for Human Experience, Springer-Verlag US, 2007.

[18] S. Schulz and R. Cornet, "SNOMED CT's ontological commitment", in: B. Smith (Ed.), ICBO: International Conference on Biomedical Ontology, LNCS, National Center for Ontological Research, Buffalo, New York, pp. 55–58, 2009.

[19] M. Klein and N. Noy, "A component-based framework for ontology evolution", In Workshop on Ontologies and Distributed Systems at IJCAI-03, Acapulco, Mexico, 2003.