Linköpings University | Department of Biomedical Engineering (IMT)
Master thesis, 30 hp | Biomedical Engineering
Spring 2016 | LiTH-IMT/MASTER-EX--16/039--SE

Improving the expression repository prototype for handling updates to SNOMED CT

Medhanie Weldemariam

Supervisor: Mikael Nyström
Examiner: Daniel Karlsson

Linköpings universitet
SE-581 83 Linköping
013-28 10 00, www.liu.se

Master of Science Thesis in Biomedical Engineering

Improving the Expression Repository Prototype for Handling Updates to SNOMED CT

Medhanie Weldemariam

LiTH-IMT/MASTER-EX--16/039--SE

Supervisors:
Mikael Nyström
IMT, Linköping University

Examiner:
Daniel Karlsson
IMT, Linköping University

Institutionen för medicinsk teknik (IMT)
Department of Biomedical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Copyright 2020 Medhanie Weldemariam

Upphovsrätt

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/.

Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.

© Författarens namn/ Medhanie Weldemariam

Abstract

Following to the integration of SNOMED CT in a local electronic health record, the benefits have been increasing with time. However, such benefits may be degraded when the right information is not accessed at the right time. The purpose of this thesis work is to develop an update management solution to the local expression repository that comes from SNOMED CT. The method used to implement the work involves understanding the existing work, developing an update management prototype and testing the implemented system. The update version system for the expression repository is basically the same as the SNOMED CT core components version mechanism. The update rules are constructed under the ambiguous and duplicate inactivation reasons. The results from the evaluation and testing shows that the implemented system is able to handle the update for possible expressions. The conclusion is that it is recommended to include not only the two major update reason but every possible update that could be released from SNOMED CT so that no information loss would generally happen.

Acknowledgments

First and foremost, I praise God, the almighty for providing me this opportunity and granting me the capability to proceed successfully. Next, I would like to thank all my family members especially my parents for their endless love and encouragement. Thank you very much to the Swedish Institute (SI) for offering me the chance to attend my master's degree program in Sweden. I would like to thank my professors Daniel Karlsson and Mikael Nyström for their continual assistance during this thesis project work. I also offer special thanks to all my friends for their encouragements during this project and my study time in Sweden.

Medhanie Weldemariam

November 2016,

Linköping, Sweden

Table of Contents

List of Figures

Lis of Tables

Abbreviations

RF2	Release Format 2
SNOMED CT	Systematized Nomenclature of Medicine Clinical Terminology
IHTSDO	International Health Terminology Standards Development Organization
UG	SNOMED CT® User
TIG	SNOMED CT® Technical Implementation Guide
EG	SNOMED CT® Editorial Guide
EHR	Electronic Health Records
CM	SNOMED CT Concept Model
FSN	Fully Specified Name (FSN)
SCG	SNOMED Compositional Grammar
WS	White Space
Refset	Reference Set

1 Introduction

This chapter is going to be an introduction to the thesis project, starting with a motivation in the form of a problem description in section 1.1. In section 1.2 the underlying purpose of the project is described, followed by explicit research questions in section 1.3. Finally, delimitations are considered in section 1.4. Additionally, a short overview of the structure of the thesis report is given in section 1.5.

1.1 Motivation

The development of digital technology such as electronic devices and web-enabled services are transforming most of our daily activities. The digital health infrastructure, such as Electronic Health Record (EHR) is one of the areas in which this technology is playing a great role. The term EHR- Electronic Health Record, refers to application software that allows healthcare professionals to keep track of health information about their patients electronically [7]. It stores patient information in a digital format including personal health records, treatments and allergies, laboratory test results, radiology images, vital signs, demographics such as age and weight, billing information and much more and that may be shared across different health cares through network connections [7]. A successful EHR, if implemented correctly can improve the quality of patient care, by increase productivity, and expand revenue. Moreover, it also benefits patient care by reducing medical errors, facilitating correct diagnoses, generating patient reminders for preventive and follow-up care and enhancing communication with patients [1].

Structured and well organized medical information is the main facilitator behind to those all benefits mentioned above. The lack of data standards has been a challenge to electronic connectivity in healthcare centers. An EHR must be designed in a way that it can achieve system interoperability and the benefits of a national health information infrastructure so that the communication among different health cares can be improved [7]. Thus, the data must be collected and maintained in a standardized format, using uniform definitions, in order to link data share health information among systems. To accomplish this, healthcare delivery organizations and providers will need to deploy a standard encoded data such as the SNOMED CT.

SNOMED CT which is abbreviated as Systematized Nomenclature of Medicine Clinical Terms- is a well-defined and computer-process-able resource for medical/clinical terminologies [1]. Sometimes it is considered as the most comprehensive terminology in the world [9, chapter 12]. It is owned, maintained and distributed by the International and non-profit organization called International Health Terminology Standards Development Organization (IHTSDO) which is located in Copenhagen, Denmark. And its primary purpose is to provide an accurate recording and sharing of clinical and related health information and achieve the semantic interoperability of health records. [3-section 3.1]. It contains hundreds of clinical concepts and attributes with unique codes that provides a standardized foundation for electronic health records (EHRs).

The SNOMED CT acting as the main source of core terminologies for the EHR, it allows collecting and recording clinical evidence in a consistent and common way of representations [1]. As a result of this, relevant clinical information can be shared with other healthcare centers having a common understanding and interpretation of the shared information. As a result of this, it reduces the specialty and geographical boundary effects that arise from the use of different terminologies or coding systems by different clinical departments and countries. The SNOMED CT also plays a great role in improving the communication among health care centers by providing an accurate access to relevant information, reducing the risk of duplications and errors. These can benefit the society to identify the key health issues, controlling the health of the society and keep updated to the major changes in clinical practices.

According to IHTSDO regulations [1, section 14], every member (licensed) countries have the right to access the terminologies, store them locally, manipulate and define their own local version of SNOMED CT such as expressions (refer section 2.4), medical thoughts defined based on the SNOMED CT terminologies. For example, the Swedish version of license agreement has the right to translate the terminologies into the Swedish language, as well as define more expressions which are relevant for its own use. A local expression repository is a data storage where all the SNOMED CT terminologies and expressions are locally stored.

The SNOMED CT terminology, when it is implemented in a software system, can represent clinically relevant information consistently, accurately, and effectively as an integral part of the EHR [1]. To get the maximum benefit from the SNOMED CT terminology, the system must keep its records updated and outdated information must be revised with time. Integrating the SNOMED CT ontology into the electronic health records alone would not result in getting the right and appropriate benefits of SNOMED CT. Except that the right and latest terminologies are accessed, the disadvantage could increase much more than the benefits which were supposed to be achieved from SNOMED CT implementation. An updated clinical terminology can aid in providing health professionals with a latest, easily accessible and complete information regarding medical information and similar facts.

1.2 Aim

The SNOMED CT International Edition is currently released and made available to its member countries twice a year on the 31st of January and the 31st of July [1 section 13]. The update release files could be sent to the member countries some more time in advance from the formal date, but by the end of the 31st January/July members must receive the release updates.

Once the SNOMED CT terminologies are mapped into the local repository, they must be kept updated by integrating the new releases of SNOMED CT from IHTSDO. Users should be aware that regular updates of SNOMED CT are made available and should be used in their systems to benefit from continuous improvements to coverage and quality. Thus, it is important to keep an eye for any update, understand the release schedule and the structure and content of the release files.

The aim of this thesis work is to extend the system which is already in use by designing and implementing a new feature prototype that can handle the update management of the expressions from the SNOMED CT terminologies in the local electronic health record database. The current system is able to store and query the SNOMED CT expressions in the Electronic health record. The expressions received from the SNOMED CT are stored in the local expression repository only once.

1.3 Research questions

The following research questions shall be answered to meet the goals and objectives of this thesis project:

1.3.1 Updates/changes to RF2-core components

Which SNOMED CT component changes are considered important for this work? This thesis project is going to make a study and identify the possible SNOMED CT component updates that could be received from IHTSDO released files. Next, to this, this thesis project is going to deal with identifying the most important component updates, out of all the component updates identified, that should be given a focus at the moment based on the current demand.

1.3.2 Update management rules

What possible update rules can be proposed to manage the release updates in the existing expression? Once the release updates from the SNOMED CT are known, it would be necessary to think about the representation of the SNOMED CT expressions in the local expression repository. These update rules will help to identify the effect of the released updates on the already existing expression.

1.4 Delimitation

As mentioned in section 1.2 and 1.3, this thesis project is going to be an extension of a system which is previously developed. This thesis project is interested in adding a feature that manages the updates from SNOMED CT on the existing expression repository. Thus, this work will construct a set of update rules for the major updates of SNOMED CT components based on the RF2 structure. The proposed update rules will be implemented and tested to evaluate by considering sample test cases on some of the update domains. Moreover, documentation is also one of the main targets of this work, thus a well-documented analysis of the problem, as well as discussion of possible solutions, will be handed in.

1.5 Report structure

This thesis documentation begins with an introduction in chapter 1. It covers the problem from a general perspective and presenting the underlying purpose of the thesis project. Moreover, explicit research questions and delimitations of the work are formulated. Chapter 2 provides a more detailed description of the assignment by providing a requirement specification, etc. In chapter 3, a theoretical basis for the problem area is presented by reading up on related works or the existing system and other information of relevance for the research questions. Chapter 4 covers the methods used in implementation and evaluation of the update management rules. Results and discussions are presented in chapter 5, followed by conclusions in chapter 6.

2

Theory

This chapter presents the theoretical basis for this thesis project. It starts with a brief introduction to SNOMED CT by presenting the logical and conceptual model of SNOMED CT. The SNOMED CT expressions, Release file Format 2 (RF2), its structure and specification are described. The method presented in chapter three greatly depends on the structure of the SNOMED CT as well as its update release structure, RF2; therefore, it is important that it is presented in this theory chapter. The main goal of this chapter is to introduce the reader to the basics of the idea that could greatly be used in the coming chapters. Furthermore, this chapter might also present the current functioning work in a form of related work.

2.1 Introduction to the SNOMED CT

SNOMED CT is a multilingual and all-inclusive clinical terminology which covers a wide range of systematically structured computer readable medical resources which provides codes, terms, synonyms and definitions used in clinical documentation and reporting [8]. It enables a wider sharing and reuse of structured clinical information by reducing the specialty boundary effects that arise from the use of different terminologies or coding systems by different clinicians or departments. It also allows consistent processing and presenting of clinical data in electronic health records in ways that serve different purposes [1]. Moreover, its global scope characteristics also enable to reduce geographical borderline effects arising from the use of different terminologies or coding systems in different organizations and countries [8].

2.2 Logical Model of SNOMED CT

The contents of SNOMED CT are defined under their three main core components. The basic structure of these components and their relationship to each other is defined using the logical model of SNOMED CT [1]. In the following two sub section of this chapter, the SNOMED CT Core components as it can be seen in figure 1, and Reference sets will be discussed.

2.2.1 Core Components of SNOMED CT

In its current form, the SNOMED CT terminologies are represented using three building block components; these are; Concepts, Descriptions and Relationships [1, section 5]. Their relationship among each other can be seen in figure1 taken from SNOMED CT Starter Guide documentation [1].

2.2.1.1 Concepts:

The SNOMED CT concepts are the building blocks for meaning [8]. They are used to represent clinical thoughts and knowledge to enable unambiguous recording of data. These concepts cover various aspects of clinical practice and each concept is identified using its own unique identifier [1].

For example: 22298006 |myocardial infarction|

The number 22298006 represents the identifier of the concept and the human readable text "myocardial infarction" also represents the preferred name for the concept.

2.2.1.2 Descriptions:

The SNOMED CT descriptions are human-readable terms (strings of readable characters) used to express the meanings of the concepts [1]. They appear linked to the concepts to give an appropriate understanding. The SNOMED CT content also contains several associated terms or synonyms associating to clinical concepts, each with their own unique identifier [8].

Every concept is represented using either FSN - Fully Specified Name or Synonyms types of description [8, section 4.3]. The FSN are, normally not visible and are used to uniquely and unambiguously describe a concept [1, section 5]. They can play a great role in differentiating concepts that are referred using the same phrase. Synonyms can be seen as the type of concept's description which can be used to select a concept and are not necessarily unique [1, section 5]. Thus, a concept can possibly have multiple synonyms; however, their interpretation can be decided using the concept identifier. Here, we can see that synonyms provide an advantage, in that users can choose any available term of their preference to refer to a specific clinical meaning.

2.2.1.3 Relationships:

The SNOMED CT relationships (concept attributes) link the concepts to other concepts that have related meaning in some way [1, section 5]. These links, or semantic relationships (sometimes can be referred as attributes), between clinical concepts, are concepts for themselves and have a relationship identifier [1, section 5]. They provide a formal definition of the concepts and can be used for various purposes such as tracking of retired concepts, representing "facts" that may vary, and support post-coordination (section 2.2.2) by suggesting valid qualifiers.

The relationship types-

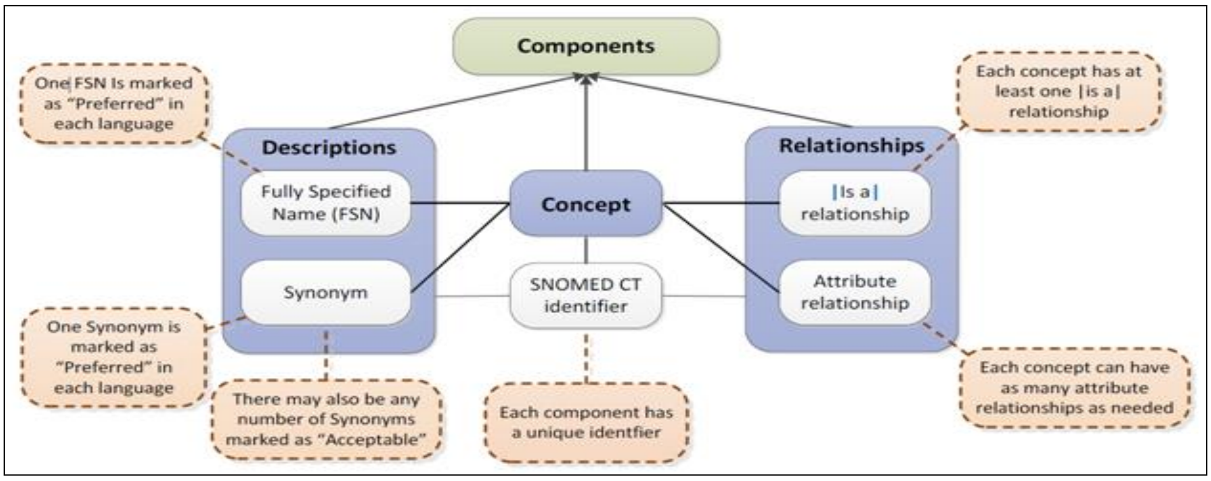
Components
One FSN Is marked as "Preferred" in each language
Each concept has at least one |is a| relationship
Descriptions
Relationships
Fully Specified Name (FSN)
Concept
|Is a| relationship
Synonym
SNOMED CT identifier
Attribute relationship
One Synonym is marked as "Preferred" in each language
There may also be any number of Synonyms marked as "Acceptable"
Each component has a unique identfier
Each concept can have as many attribute relationships as needed

Figure 1. SNOMED CT Logical Model [1]

-can be of different types, such as IS_A (relationship identifier: 116680003) and FINDING_SITE (relationship identifier: 363698007) [1].

Hierarchal (subtype) Relationship

When two concepts are directly linked by a single |is a| relationship, as it can be seen in figure 2, the source concept (the one at the bottom) is said to be a "subtype child" of the destination concept (the one at the top) [8]. The destination concept is referred to as a "super-type parent" [1, section 5]. Any concept that is the source of a sequence of one or more |is a| relationship leading to a specified destination concept, is a "subtype descendant" [1, section 5] of that concept. Similarly, any concept that is the destination of a sequence of one or more |is a| relationship leading to a specified source concept, is a "super-type ancestor" [1, section 5] of that concept. It is also said that the source concept of a |is a| relationship "is subsumed by" the target concept, and that the target concept of a |is a| relationship "subsumes" the source concept.

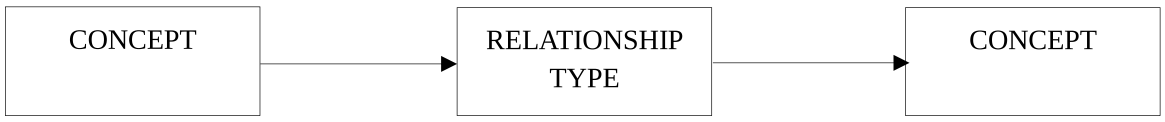

Figure 2. SNOMED CT Concept Relationships

Concept unique identifiers do not carry a constantly defined meaning; the meaning of the concept is defined by its relationships which are related to other concepts in the terminology. Every concept can have multiple human-readable descriptions including the unique and mandatory Fully Specified Name. For example:

CONCEPT

Concept ID – 22298006

Fully Specified Name **-** Myocardial infarction (disorder)

DESCRIPTIONS

Preferred Term **-** Myocardial infarction (disorder), Myocardial infarction

Synonyms **-** Myocardial infarction, MI - Myocardial infarction, Infarction of heart,

Cardiac infarction, Heart attack, Myocardial infarct

RELATIONSHIPS

Myocardial infarction **IS A**

Myocardial disease (disorder)

Necrosis of anatomical site (disorder)

Ischemic heart disease (disorder)

The defining **ATTRIBUTES** are

Associated morphology **-** Infarct (morphologic abnormality)

Finding site **-** Myocardium structure (body structure)

2.2.2 Reference sets (Refsets)

In addition to the core components, SNOMED CT also supports Reference sets. Reference sets are a standard and flexible set of SNOMED CT components, which are used to customize or modify any additional non-defining requirement of the components [8]. The SNOMED CT is a huge terminology, actually the most comprehensive clinical terminology in the world, which consists of more than 300, 000 concepts. Since this all concepts are needed in practice it makes sense to extract a smaller a collection of concepts from SNOMED CT.

The reference set can basically act like an index, pointing to a chapter or concept that is relevant to a subject and each reference set has a unique numeric concept identifier. The reference set serves for different purposes, these include the representation of subsets, language preferences for use of particular terms and to manage extensions, data structures foundation for SNOMED CT and SNOMED CT release files, and support their implementation [1].

Thus, a reference set is a purpose built a subset of the terminology that contains only the terms that are needed in a specific context, for example, all terms needed for medical specialty or in an emergency department or for pick lists in clinical-facing applications such as problem lists [3, section 7.9].

2.2.3 The SNOW OWL and SNOMED CT

The SNOMED CT Ontologies and terminologies are continuously authored using Description Logics [10] such as the OWL [11], which support formal descriptions and definitions. Description logics are a family of knowledge representation languages that can be used to represent the an application domain knowledge in a structured and formally well-understood way [10]. The name DL-description logics, is drived from the fact that, on the one hand, the important notions of the domain are described by concept descriptions, i.e., expressions that are built from single concepts and single roles using the concept and role constructors provided by the particular description logic (DL) [10].

OWL (The Web Ontology Language) Ontologies is the World Wide Web Consortium (W3C) recommended ontology language for the Semantic Web, are used to capture knowledge about some domain of interest [11]. Different ontology languages provide different facilities. OWL which is basically from the World Wide Web Consortium (W3C), is the most recent development in standard ontology languages. SNOW OWL is a SNOMED CT Browser which is used as an editor of the SNOMED CT concepts. It has two basic functions, which are displaying detailed information on a concept and also serves as an editor so it allows making changes to a concept, for example, adding another clinical phrase to describe the concept or retiring the concept. Moreover, it also plays a great role in creating and maintaining different kinds of reference sets which are named from the SNOMED CT RF2. Release Format 2 specification (chapter 3, section 3.2) can also be used to compare and export the reference sets. It describes the concepts in the domain and also the relationships that hold between those concepts.

2.3 Concept Model of SNOMED CT

The SNOMED CT concept model is used to define the rules in which how the SNOMED CT concepts can be defined or represented. It describes the logical definitions that are used to represent the permitted domains and ranges of a concept [1]. The SNOMED CT terminologies are organized in a 19 high-level hierarchical concepts/topics ranging from |Clinical finding| to |SNOMED CT Model Component| listed from general to the most detailed [3, chapter 6]. All of this high-level concepts are occupied by the root concept of SNOMED CT.

The SNOMED CT hierarchy is defined using the Is-a relationship type by relating a concept to a more general concept. Almost all active SNOMED CT concepts are the source of at least one |is a| relationship except the root concept, |SNOMED CT Concept|, which is the most general concept [1, section 6]. Similar to the discussions in section 2.2.1, the |is a| relationship states that the root concept is the super-type of the rest of concepts and also all the other concepts are subtype to the source/root concept. Thus, the hierarchy can increase in length by adding more |is a| relationship to the top concept, as a result, the concepts turn from being general to more specific.

SNOMED CT attribute (or relationship type) is used to represent a characteristic of the meaning of a concept by associating the source concept with the value of defining characteristics and it has a name which is represented by a *concept [6, section 5.1]*. SNOMED CT currently uses more than seventy-eight (78) defining attributes when defining the meaning of concepts. Each SNOMED CT relationship types (attributes) can be applied into a set of concepts in the hierarchy using the Domain and Range. The Domains are sets of SNOMED CT concept hierarchies to which a specific attribute can be applied [1]. The ranges are the set of SNOMED CT concepts that are allowed as the value of a specified attribute [1].

The SNOMED CT Editorial Guide chapter 5 or SNOMED CT Technical Implementation Guide, chapter 6 can be referred to the list of SNOMED CT's allowed Domain and Ranges.

The following example will clearly show the relationship between domain, attribute, and range, firgure 3 and 4 also shows the general structure and more specific example of the representation of domain and range relationship.

- |Clinical finding| is one of the top-level concepts, and it represents the result of a clinical observation, assessment or judgment which includes normal and abnormal clinical states (e.g. |asthma|, |headache|, |normal breath sounds|). Its hierarchy can include concept used to represent diagnoses.
- |Procedure| is also another top level concept which represents activities performed in the provision of health care which includes diagnostic procedures and medical administrative procedures (e.g. |appendectomy|, |physiotherapy|, |subcutaneous injection|).

The Domain of the attribute |associated morphology| is the |clinical finding| hierarchy. Therefore, a |procedure| cannot have a |associated morphology|. However, a |procedure| can have a |procedure morphology|.

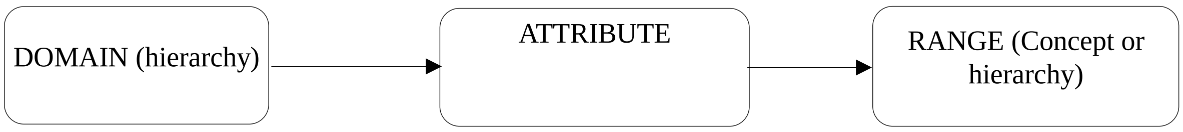

Figure 3. The logical definition of domain and range

The range for the attribute |associated morphology| is the concept |morphologically-

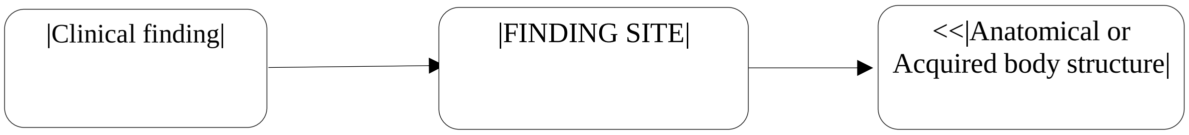

Figure 4. Example of domain and range

-abnormal structure| and its subtype decedents. The range for the attribute |finding site| is | anatomical or acquired body structure| and its subtype descendant in the |body structure| hierarchy. The diagram example below is taken from SNOMED CT starter guide chapter 6 and describes this idea [1].

The SNOMED CT conceptual model is the basis for defining and handling the clinical terms and definitions stored in clinical records and also facilitates the right use of clinical terms for further analytical decision.

2.4 SNOMED CT Expressions

Clinical information recorded in EHRs using SNOMED CT is commonly represented using SNOMED CT concepts [5, section 2]. These concepts may either be defined in the SNOMED CT international release or in an appropriate SNOMED CT extension. There are, however, times when a clinician needs to record and share a clinical meaning, which has not been defined in any release of SNOMED CT. In these situations, SNOMED CT expressions can be used to represent a new clinical meaning using the definition rule compositional grammar syntax [5, section 5]. SNOMED CT allows defining additional clinical phrases when the SNOMED CT concepts are found to be not enough expressive to define a specific clinical meaning at a required level. Thus, more medical terms and definitions can be recorded, just as a single but combination of ideas that doesn't require a single concept to be included [2].

SNOMED CT expressions are also components supported by SNOMED CT. They are an automatically processable sequence of one or more concept identifiers, which are used to represent a meaningful clinical idea in a logical manner. Expressions are represented using the SNOMED CT compositional grammar, which is a lightweight syntax for representing SNOMED CT expressions [5, section 2]. Clinical expressions using SNOMED CT concepts can be of two types: precoordinated expressions, which use a single SNOMED CT concept identifier; and postcoordinated expressions, which contain more than one SNOMED CT identifier.

2.4.1 Precoordinated expressions

In situations where clinical expressions represent two concepts or a concept and modifiers occur commonly in clinical settings, it is important to create one code that represents both concepts. Precoordinated expressions are expressions that represent clinical meaning using individual concept identifiers which are already predefined in SNOMED CT [3, section 8.2.1.

For example, a new SNOMED CT code can be generated to represents a “severe headache” which can be used to express a situation when a patient experiences a serious headache. This newly generated concept is identified using a single concept identifier and is the result of precoordination, and it would allow users to document a single concept than identifying two different concepts, i.e. a headache and severe. This would also result in simplifying the process of data storing and information retrieval (IR) since a single code representation is simple to store and retrieve than a combination of two codes.

Another example can be, the concept 174041007 |laparoscopic emergency appendectomy| [3, section 8.2.1.1]. This would help users from having to identify three distinct codes, which are ‘removal of appendix’, ‘using a laparoscope’ and ‘emergency procedure’ during clinical documentation. This also facilitates the process of both information storing and retrieval, as a single code much easier to store and retrieve than a combination of two codes. Besides the unique concept identifier and descriptions, each concept also has a formal logic definition characterized using a set of defining relationships to other concepts.

2.4.2 Post-coordinated expressions

In addition to precoordinated concepts, SNOMED CT also allows creating postcoordinated expressions which are used to represent clinical meaning by combining two or more concepts and attribute relations [3, section 8.2.1.2.

For example there exist a SNOMED CT concept for the symptom “headache” which is identified by the identifier code 25064002 and also there exist another SNOMED CT modifier concept called “severe” which is used to or helps to express the level or degree of the pain that a person can experience that is identified by the identifier code 24484000. Having these two separate SNOMED CT concepts, one can combine them to create a new concept/expression to give an additional definition for the situations when a patient has a severe headache. Similarly, a normal headache only and or a mild headache can also be represented in a similar mechanism. Thus, we can say that the above two concepts are now postcoordinated expression since they produce an expression which represents a single clinical meaning.

Postcoordination is not just a list of concept identifiers, it follows a set of rules that mimic the way attributes and values are used to define SNOMED CT concepts. And it extends the terminology content in details without adding more precoordinated concepts [1]. The need for postcoordination can arise when there are no precoordinated concepts available for certain clinical terms.

2.4.3 Representing SNOMED CT expressions

SNOMED CT expressions are described using a syntax called SNOMED CT Compositional Grammar and has the following general format. The syntax of SNOMED CT Compositional Grammar is specified in Augmented Backus-Naur Form (ABNF) [5, section 5.2]. A concept definition minimally consists of the concept identifier and an optional concept description (human-readable). Expressions with multiple focus concepts are expressed by joining two or more concepts with a plus sign and refinements to the concepts can also be appended after a colon following the concept as it can be seen in figure7 below.

Focus Concept: attribute name = attribute value

Focus Concept:
Attribute Name
Attribute Value

Figure 5. The main parts of an expression

The attribute name and an attribute value are SNOMED CT concept identifiers, used by SNOMED CT to represent relationships and post-coordinated expressions in a generic way. The attribute name identifies the type of information and the attribute value provides a value.

Simple expressions basically use the following syntax to define expressions [5, section 5] which can also be rered to figure 5.

expression = ws [definitionStatus ws] subExpression ws.

As discussed in section 2.2.1, post-coordinated expressions made up of one concept identifier, and this syntax at its simplest form is also showing the representation of simple/ post-coordinated expressions.

For example 22298006 | Myocardial infarction |

Even if it is not a must, but it is always recommended to include the term for the concept Identifier as it increases the readability of the represented expression.

Post-coordinated expressions, as discussed in section 2.2.2, they are made from more than one concept identifiers, and in such cases, the above simple expression could extend itself to include multiple focus concepts. This can be represented by including multiple concepts under its concept reference, as many as needed concatenating using the sign “+” in between every concept [5, section 5].

For example, the expression 421720008 | spray dose form | + 7946007 | drug suspension | is a combination of two different SNOMED CT concepts, and thus it is combined using “+” sign.

While representing an expression, it might be necessary to qualify a concept so that refinements could be added. These refinements are added by including the qualifying expression next to the colon sign “:” and the qualifying concept before the colon, thus it will refer to the syntax we discussed above "attribute = value" [4, section 6.3]. For example:

```
125605004:

              363698007=29627003
```

Here, if more than one attributes are needed to appear they can be separated using a “,” comma.

The *refinement* part of the syntax can be consists of a combination of both grouped and un-grouped attributes which helps to refine the give expression. As the definition of an expression becomes larger and increases its qualifications, it could be very difficult to read the expressions and might result in ambiguity. For example, expressions might contain many attributes that qualify the expression [4, section 6.3.2] and also there might be cases during which expressions are nested by one another [4, section 6.3.3]. In order to read and understand such nested expressions clearly, it is important that they are grouped using { } and also separated using brackets ( ).

For example, expressions can be refined using nested parentheses

```
53057004 |hand pain|:

              363698007 |finding site| = (76505004 |thumb structure| :

                                            272741003 | laterality | =7771000|left|)
```

Expressions can also be refined by grouping in braces

```
71388002 | procedure |:
```

```
{260686004 | method | = 129304002 | excision - action |, 363704007 | procedure
site | = 66754008 | appendix structure |}
```

An expression that represents a clinical meaning can be semantically equivalent to or a subtype of another expression [5, section 6.8]. In such cases the expression can use the definition status equivalentTo which is represent by three consecutive equals sign, "===" and subtypeOf, which is represented by three consecutive less than sign, "<<<"

Some attributes when they are added to the concept model of SNOMED CT, their values might be required to be specified and be concrete. These values could take the data types such as decimal, integer, and strings. Thus, both the numeric values can be specified by preceding a hash # sign. Similarly, the string values of attributes can also be specified using a double quote " " as their prefix [5, section 6.7].

The following picture summarizes the main components of an expression is taken from SNOMED CT Expression constraint language specification, section 4.1 can be referred for further details.

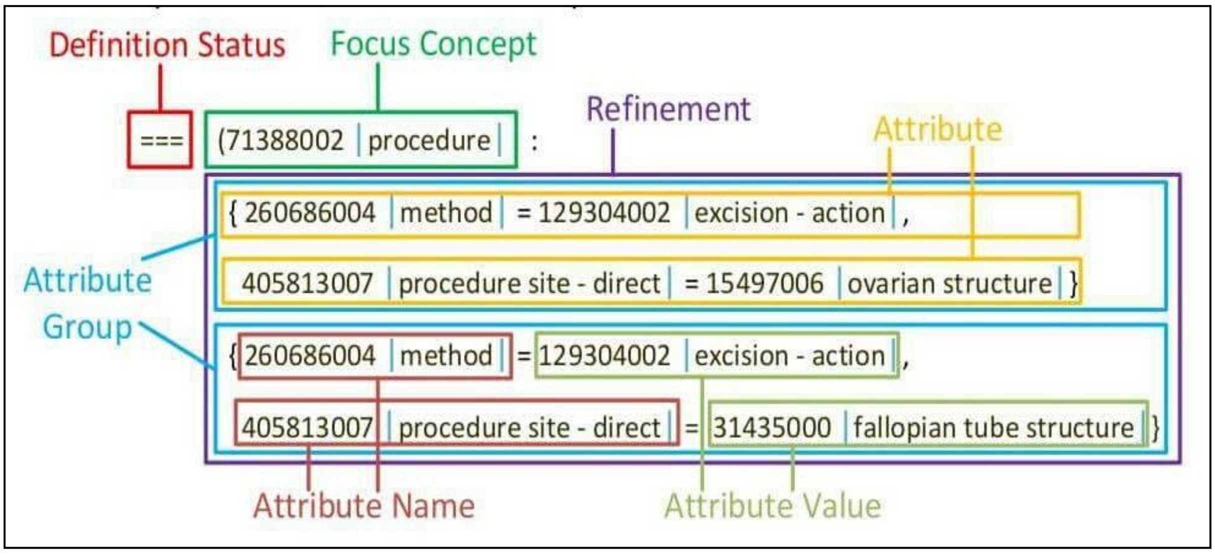
Definition Status
Focus Concept
Refinement
Attribute
===
(71388002 |procedure| :
{ 260686004 |method| = 129304002 |excision - action| ,
405813007 |procedure site - direct| = 15497006 |ovarian structure| }
Attribute Group
{260686004 |method| = 129304002 |excision - action| ,
405813007 |procedure site - direct| = 31435000 |fallopian tube structure| }
Attribute Name
Attribute Value

Figure 6. The main sections of SNOMED CT Expression [5]

2.5 SNOMED CT RF2 and File Formats

The SNOMED CT vocabulary is a dynamic clinical terminology which means that it is not a permanently stored data, rather, with time, as it has been described earlier every 6 months on the 31st of January and the 31st of July, changes could happen to the terminologies. Such changes might include, new concepts are added to the existing terminology, or the currently existing concepts can be made inactive or else their modelling structure is changed to reflect current medical practice. This shows that the terminology should have mechanisms to keep aligned with current health care requirements. Thus, users need to address how to deal with a changing terminology in their healthcare products and that some of the codes may become inactive and thus no longer recommended for use. Moreover, it could also be possible to retrieve exactly what has originally recorded in the medical record and the version of the code sets that were available at the time.

IHTSDO regularly updates SNOMED CT and the policy by the IHTSDO implies releasing the updates twice per year that includes the following components [1, section 13]:

- SNOMED CT Core terminology components i.e. concepts, descriptions, and relationships
- Reference sets, which represent subsets, cross maps to classifications and coding schemes.
- Documentation

SNOMED CT is currently available in a well-defined release format called RF2 (release format 2). The RF2 is the modern format which was developed by the International Health Terminology Standards Development Organisation (IHTSDO) to benefit the creation of reference sets [3, section 9.2.1].

RF2 is a well-defined distribution mechanism provided by SNOMED CT and suppliers need to transform the content to import into their data model so that it can be used by healthcare professionals via their electronic patient records.

The main purpose of the release format is for its use in the distribution of SNOMED CT, to provide a format that is flexible, unambiguous and useful. The RF2 is not intended to control which database format SNOMED CT is stored in when it is used in documentation systems [3, section 9.2.1.1.1]. However, it is aimed at strengthening SNOMED CT by providing a format that is simple and stable, while enabling innovation through transformation to cater for variable requirements.

RF2 consists of several tab-delimited text files, and represent the character content using the Unicode UTF-8 specification. Each text file contains a table with rows and tab-separated columns [3, section 9.2.1.3.3]. It supports extensions to the International Release using name spaces allocated to licensees to represent the provenance of added components and to ensure the uniqueness of identifiers [1, section 13]. The three most important components of the release files are the concepts, descriptions and relationships [3, section 5.5] and their file formats is described in the following sub sections.

2.5.1 Concept file

The Concept file stores data related to SNOMED CT clinical concepts. It is defined and holds a meaning using its equivalent attribute Fully Specified Name, which is stored in the description file [3, section 5.5.3.1]. The concepts table contains one concept per row. For each concept, a unique ID number is stored, together with information regarding whether the concept is fully defined or primitive.

Key Fields	
id	Uniquely identifies the concept.
Data fields	
effectiveTime	Express the date at which the concept come into effect.
active	Express the status of the concept, I.e. weather it is active or inactive after it has come into effect
moduleId	Specifies the concept version's module.
definitionStatusId	Indicates whether a Concept is Primitive or Fully defined.

Table 1. The concept file

2.5.2 Description file

The Description file stores data that describe SNOMED CT concepts. It consists of one concept description per row. For each description, a unique ID number is stored, together with information regarding which concept the description refers to and the textual term from which the description is composed. But, the timestamp field which contains the latest effectiveTime is used to identify the current record [3, section 5.5.3.2].

Key Fields

id	Uniquely identifies the description.
Data fields	
effectiveTime	Express the date at which the description come into effect.
active	Express the status of the description, i.e. weather it is active or inactive after it has come into effect
moduleId	Specifies the module to which the description belongs.
conceptId	The concept's ID number that is being described.
languageCode	Specifies the code to which the language is being described in accordance with ISO-639-1.
typeId	Specifies the type of the description whether it is fully specified name or synonym.
term	Specifies the text of the description
caseSignificanceId	Specifies the concept enumeration value which describes the case of the specific description.

Table 2. The description file

The active attribute in a record can be made false ('0'), which then shows that the description is inactive at that point in time. It can also be made true ('1'), so that it takes the meaning that the description is associated with the concept identified by the conceptId field.

Data fields; conceptId, languageCode, and the typeId are immutable data fields of description file, which means they will not change between two rows with the same id. Whenever a change is required to one of these fields, then the current row will be made inactive (by appending a row with the same id and the active field set to false) and a new row with a new id will be appended. Only limited changes may be made to the ' term' field, as defined by editorial rules [3, section 5.5.3.2].

2.5.3 Relationship file

The relationships file stores relationship details between two concepts and contains one relationship per row. For each relationship a unique ID number is stored, together with information regarding which concept the relationship originates from, which concept the relationship links to and the relationship type, e.g., “is a” or “finding site”. It also states whether the relationship is grouped with other relationships, and the relationship's characteristic type is denoted [3, section 5.5.3.3].

Key Fields	
id	Uniquely identifies the relationship.
Data fields	
effectiveTime	Express the date at which the relationship come into effect.
active	Express the status of the relationship, i.e. weather it is active or inactive after it has come into effect

moduleId	Specifies the module to which the relationship belongs.
sourceId	Represents the ID number of the source concept in the current relationship.
destinationId	Represents the ID number of the target concept in the current relationship.
relationshipGroup	Specifies the logically associated group to which the relationship belongs.
typeId	Specifies the type of the relationship
characteristicTypeId	Specifies the characteristic type of the relationship. Either defining, qualifying or additional.
modifierId	Specifies the type of Relationship's description logic restriction.

Table 3. The relationship file

The three tables above, as well as most other tables that are part of RF2 files, also contain columns for information on module and release. The division into modules is, for example, used to indicate that one concept belongs to the international version of SNOMED CT and that another belongs to example the Swedish version. The release handling column is used so that all changes to SNOMED CT are traceable. Concepts, descriptions, relationships and other components of SNOMED CT cannot, therefore be removed completely; it is only possible to mark them as inactive/retired. The release handling columns are used to store information on when amendments and deactivations have been made.

2.6 Release types

The SNOMED CT release file specifications comes in three different types of release files

2.6.1 Full

The full view release file is the most powerful type of terminology service view which allows the server to provide access to any selected version of SNOMED CT from a single representation of the SNOMED CT resource [3, section7.1.1]. This makes full use of the version features in RF2. Alternatively, a full view terminology server can also provide access to changes of components between any two specified points in time representing versions of known interest to its users. The files representing each type of component contain all versions including the component that has been released before.

2.6.2 Snapshot

The simplest of all the release file types is the single snapshot view which provides access to a single or current release version. The files representing each type of component contain one version of all components released up to the time of the snapshot release [3, section7.1.1]. The version that every component holds in a snapshot release is the most recent release version of that component at the time of the snapshot.

2.6.3 Delta

The files representing each type of component contain only component versions created since the past releases. The version in each component of a delta release represents either a new component or a change to an existing component [3, section7.1.1].

2.7 Choosing Release Types

The SNOMED CT users designing a terminology server need to decide whether their server will only provide access to a single current view of the SNOMED CT resource or will also support retrospective views of earlier versions of the terminology [3, section7.1.2]. The single snapshot view is simplest to implement and matches the service most vendors offered with older SNOMED CT release format. A complete view is now possible using Release format 2 and this offers several significant advantages. It supports incremental updates allowing smoother transition as new versions become available. It also allows changes between versions to be detected more easily and can be used to evaluate queries against an earlier version for comparative purposes. People choosing a terminology server need to consider whether a server that only supports a single snapshot view of the current version meets their requirements. If they require access to previous versions a server that supports the full view is likely to be the best long term solution. A server that allows access to multiple discrete snapshots may provide a reasonable interim solution but may be less flexible and less easy to maintain [3, section7.1.2].

2.8 Release Format (RF2) Update Rules

The SNOMED CT contents grow up with the changes coming in every release and these changes might including the addition of new concepts, new Descriptions, and new Relationships between concepts, or updates and retirements of any of these components. The reason for such changes may include findings of new drugs, new procedures, understandings and investigations of health and diseases processes, and new suggestions and ideas raised by those SNOMED CT licensed partners.

According to the SNOMED CT RF2 file structure, each component has an associated fields, ***effectiveTime***, which takes the date at which the component comes into effect and active field, which can take values of true ('1') or false ('0') [3, section 5.5.1.5]. Both these fields in the release file enable the use of historic data to keep a record of any changes to every component, providing full traceability. Once released, a row in any of these files will always remain unchanged.

When a new release version of a component becomes available, the older version will neither be deleted nor replaced, rather a new version of it will be created by changing the current properties, I.e. adding a new row to the applicable file, containing the updated fields, with the *active* field set to true and the timestamp in the ***effectiveTime*** field indicating the nominal date on which the new version was released.

A component can become inactive with so many reasons, in which some of them are discussed in the next sections. Generally, when a component is made inactive, a new row is added with the same id, containing the same data as the final valid version of the component, but with the *active* field set to false (0) and the timestamp in the ***effectiveTime*** field indicating the nominal date of the release in which the version of the component become valid. It is thus possible to see both the current values and any historical values of a component at any point in time.

Records of a release files, always follow the same form of release structure. The content of a component will not be dated with the same date that it appears in. Thus, If a component record is changed a number of times between releases (during an edit and review process), only the most recently amended record will be appended to the release file, not individual records showing each separate edit to the released component.

2.8.1 Concept inactivation and version management

The evolution of SNOMED CT terminology does not change the meaning of a concept. If the meaning of the concept is needed to be changed, may be because it is found to be ambiguous, redundant or incorrect, then the concept is made inactive [3, section 4.1.1.4.1]. When concepts are made inactive, they are never deleted.

SNOMED CT concepts can be found as duplicated and are required to be inactivated. A concept is considered as duplicate or redundant if two concepts are found to hold the same meaning. As a result, SNOMED CT will recognize this possibility and facilitates recognition of equivalent statements and will provide a reference to the continuing, active representation of that concept.

SNOMED CT concepts can also be found as ambiguous and are required to be inactivated. A concept is considered ambiguous if it is found to be referring to two or more meanings or possibly be interpreted into more than one meaning.

In the following part, the history mechanism used to control version of inactive concept files is discussed thoroughly. Note that, the source behind every rule is based on the SNOMED CT technical implementation guide and for more information, it can be referred at, [3, section 5.5.1.6].

Only one concept record with the same id field is current at any point in time. The current record will be the one with the most recent ***effectiveTime*** before or equal to the date under consideration. If the active field of this record is false ('0'), then the concept is inactive at that point in time, otherwise, it is the latest record.

If a concept is inactivated, then a new row is added to the concepts file with the *active* field set to *inactive* (0) and ***effectiveTime*** field set to the *current release date*.

All relationships that have as source the concept to be inactivated will themselves be inactivated by adding a new row to the Relationship file for each relationship, using the previously used ***relationship ID*** and ***destinationID*** (concept), but the same ***sourceID*** (concept) and typeId (e.g IS-a) and the active flag set to inactive;

The following table shows how the concept history mechanism works on a concept file

Id	EffectiveTime	Active	ModuleId	DefinitionStatusId
101291009	20070701	1	| Module 1 |	900000000000074008| Primitive |
101291009	20080101	1	| Module 2 |	900000000000074008| Primitive |
101291009	20080701	1	| Module 2 |	900000000000073002| Defined |
101291009	20090101	0	| Module 2 |	900000000000074008 |Primitive |

Table 4. Concept file history mechanism

As we can see in the above table, the concept with the Id number 101291009 has been added in 2007-07-01 and then in 2008-01-01 it has been moved from module 1 to module 2. Later in 2008-07-01, the concept has been changed from primitive to define and then finally in 2009-01-01 it has been made inactive. Here we can see that the concept's Id number doesn't change as the concept evolves with changes. Also, the active field is changed from 1 into 0 only when the concept is deactivated.

All active descriptions associated with the concept will remain unchanged unless incorrect for the concept;

Rows will be added as needed to the Historical Association Reference Set, a file in which inactive concepts and their active replacements are stored, to model associations from the inactive concept to other concepts;

Active descriptions that are still associated with the inactive concept will be added to the |Description inactivation indicator reference set |, with an associated value of | Concept non-current|.

In general, the relation between a description and a concept is persistent means that If a description is no longer pertinent for a concept, then the description is inactivated. A new corrected description for the concept may be added.

2.9 SNOMED CT Reference Set Mechanism

Whenever a change is made to the contents of SNOMED CT, a mechanism in which its history is recorded is also implemented and this mechanism is known as the SNOMED CT History Mechanism [3, section 5.5.1.6]. This mechanism enables to process records prepared with SNOMED CT regardless of whether the data is expressed using the current release of the terminology or not.

Changes to the SNOMED CT components, such as the retirement and replacements of components shall be recorded in the SNOMED CT component history records by mapping to their proper reference sets [3, section 7.4.2.2]. The main reason for keeping track of these changes in the history record is that because these SNOMED CT components might have been used directly in different health-related applications, or to define the terms used such as personal health records, templates for health data entries or in any health application used for decision support and query analyses. In this manner, these historical records will enable to modify these data as the SNOMED CT is evolving.

The SNOMED CT historical mechanism has also a mechanism that gives references to the current components that are used instead of those already inactivated. These historical references enable to map data to a current equivalent concept and query to include information represented using Inactive concepts [3, section 7.4.2.3].

Furthermore, the SNOMED CT historical mechanism also plays a great role in keeping track of the necessary information during the maintenance of one or more concepts that are required to be transferred from the SNOMED CT Core to an Extension, or vice versa.

The SNOMED CT has two different history mechanism in which it keeps a record of its history, which is discussed below

2.9.1 Association Reference Set

An Association reference set is a Component reference set used to represent associations between components [3, section 5.6.2.11]. Its structure is shown in the following table.

The Association Reference set table contains References from inactive Components to other equivalent or related Components that were current in the Release Version in which that Component was inactivated. Each Reference indicates the nature of the relationship between the inactive and persistent component. Its structure is shown in the following table [3, section 5.6.2.11.2].

Key Fields	
Id	Uniquely identifies the reference set members.
Data Fields	
effectiveTime	The timestamp which stores the date at which the reference set member comes into effect.
active	The state of the referenced member at a specific effectiveTime.
moduleId	Specify the module which contains the reference set member.
refsetId	Specifies to which reference set does this reference set member belongs.
referencedComponentId	Specifies the identifier for the source component of the association.
targetComponentId	Specifies the identifier for the target component of the association.

Table 5. RF2 style association reference set table

The metadata for the association reference set is defined in the following hierarchy [3, section 5.6.2.11.3]

- `900000000000455006 | Reference set |`
    - `900000000000521006 | Association type |`
        - `900000000000522004 | Historical association |`
            - **`900000000000523009 | POSSIBLY EQUIVALENT TO association reference set |`**
            - `900000000000524003 | MOVED TO association reference set |`
            - `900000000000525002 | MOVED FROM association reference set |`
            - `900000000000526001 | REPLACED BY association reference set |`
            - **`900000000000527005 | SAME AS association reference set |`**
            - `900000000000528000 | WAS A association reference set |`
            - `900000000000529008 | SIMILAR TO association reference set |`
            - `900000000000530003 | ALTERNATIVE association reference set |`

2.9.2 Attribute Value Reference Set

A concept that has been made inactive has a default of no reason given for inactivation. If a reason has been recorded for that inactivation, it will be recorded in a particular reason for inactivation refset if not the default should be assumed. When a concept is inactivated, rows will be added as needed to the Historical Association Reference Set, to model associations from the inactive concept to other concepts.

The Attribute value Reference set table mainly contains the reason why a Component has been made inactive [3, section 5.6.2.5]. The reason for the inactivation is represented in numbers under the field "***valueId***". Its structure is shown in the following table.

Key Fields	
Id	Uniquely identifies the attribute-value reference set table.
Data Fields	
effectiveTime	The timestamp which stores the date at which the reference set member comes into effect.
active	The state of the referenced member at a specific time.
moduleId	Specify the module which contains the reference set member.
refsetId	Specifies to which reference set does this reference set member belongs.
referencedComponentId	Specifies a reference to the SNOMED CT component being referenced.
valueId	Specifies the tagged value applied to the referencedComponentId.

Table 6. RF2 style attribute-value reference set table

A 900000000000480006 | Attribute value type reference set | allows to provide additional information about particular concepts, descriptions or relationships. For example, a 900000000000480006 | Attribute value type reference set | is used to indicate the reason ***why a concepts has been inactivated***.

The metadata for the attribute value reference set is defined in the following hierarchy [3, section 5.6.2.5.3]

- `900000000000454005 | Foundation metadata concept |`
    - `900000000000455006 | Reference set |`
        - `900000000000480006 | Attribute value type |`
            - `900000000000488004 | Relationship refinability reference set |`
            - **`900000000000489007 | Concept inactivation indicator reference set |`**
            - `900000000000490003 | Description inactivation indicator reference set |`
            - `900000000000547002 | Relationship inactivation indicator reference set |`

For more possible reasons why a concept can be made inactivated, please refer to the Technical Implementation Guide which are listed in a tabular form under the section "Concept inactivation value". This thesis project is mainly interested in the reasons listed under the following table,

ID	Fully Specified Name	Comment
900000000000482003	| Duplicate component (foundation metadata concept) |	The Concept has been made inactive because it has the same meaning as another Concept.
900000000000484002	| Ambiguous component (foundation metadata concept) |	The Concept has been made inactive because it is inherently ambiguous either because of an incomplete fully specified name or because it has several associated terms that are not regarded as synonymous or partial synonymous.

Table 7. Concept inactivation value (900000000000481005)

2.10 SNOMED CT Version Management System

There are some previously developed applications on SNOMED CT auditing to manage its versions that would result in improving the quality of the terminology. In this section, we will describe some of the already implemented methods used to control the versioning issues involved.

2.10.1 Terminology Version (TV) Manager

TV-Manger [12] is a version control system/application for SNOMED CT used to compare two or more SNOMED CT versions. It was developed by J. Ingenerf and T. Beisiegel at Institute of Medical Informatics, University of Luebeck. It is implemented in Java 6.0 and uses MySQL 5.0 for storing and querying the versions of SNOMED CT. The application presents the hierarchically selected SNOMED CT concepts in a synchronized context [12]. It has the futures such as an access to selected ontologies, structural and quantitative comparisons which helps to display synchronized hierarchies of selected ontologies as well as a number of sub-concepts [12].

Even with the TV-Manager, it is still difficult to perform a comprehensive and efficient study of the area of interest. Until now the TV-Manager is mainly a proof of concept used by some interested terminology experts. Moreover, this application missed to include the history table for being able to traverse inactive concepts in one version to active concepts in another version and vice versa. Furthermore a module for determining and visualizing consequences of new versions for already coded patient data.

2.10.2 Terminology Quality Improvement (TQI)

A terminology quality improvement (TQI) [13] is the model used in the evaluation of healthcare terminologies by assessing the quality of healthcare terminologies and making improvements according to an agreed standard. It involves a serious of terminology life cycle that support the evolution of a terminology and these are; the change request section, editing, and publication [13].

The change request phase is a formal mechanism to submit any changes with respect to a given terminology. Such changes could possibly come from both internal and external terminology users and might include concept addition, revision, and deletion, along with sound rationale. These changes are later taken into consideration for further review, validation and subsequent documentation within a terminology life cycle, requiring the collaboration of terminology developers and users as well as administrative supporting personnel [13]. Following to the proper documentation of the changes, the editing phase would begin which is aimed at activating new concepts, revising existing concept or inactivating concepts based on the documented guideline. Finally, during the publication phase, a new version of the terminologies shall be made available for public use.

3 Method

This chapter mainly describes the working method, in which how this thesis project has been carried out. The working method of this thesis project has been separated into three different phases, which are pre-study, design, implementation, testing and evaluation. The pre-study and design phases are described in this chapter and the three two phases i.e. implementations, testing and evaluation will be discussed thoroughly in the next chapter. The phases and steps of the working methodologies are structured and executed in the listed order of sub-headings.

3.1 Introduction

As the contents of SNOMED CT are constantly revised by its producers, the revised SNOMED CT content shall be made available in succeeding releases to its users. Each release of the content includes a metadata that keeps a record of the contents of the SNOMED CT items such as the current status of each component's content and its history. At the time that the content is changed its metadata is also updated in line with the changes made by the SNOMED CT's producers [3, section 5.6]. It also consists of a record about what is or may be an appropriate replacement for any inactivated content. Some small portion of the concepts also become retired without having any equivalent replacement, thus the metadata keeps indicating the current status of each component.

When a concept is inactivated, it undergoes essential changes of its hierarchical relationships and requires revision on the relationships in order to eliminate any anomalies in the hierarchal structure of the SNOMED CT. This is because the relationship that contains the inactive concept as its source concept will also need to inactivate itself by adding a new row in the Relationship file with the active field set into "0" or "inactive". The result of this relationship inactivation might also effect on how it would be classified using the SNOMED CT hierarchies. The detailed impact of concept inactivation on its relationships can be referred to chapter 4, section 4.2.4 Post-coordinated Expression Update Rules. Here, it is important to make the proper update on the expression which is defined by the concepts. It is also important to follow a common structure to the rules of the update management. As a result, the next sub sections will discuss on the concept inactivation rules that are required to manage the expression changes.

3.2 Pre-study

The first part of the method framework employed in this thesis project is a pre-study of relevant source material.

A study on the SNOMED CT releases was a starting point to evaluate the release format 2 (RF2). During this phase, the proposed literature and other relevant materials have been analysed to get an insight or deep understanding of the release format 2 in each version. As a result, a prototype consists of the update management rules required for the expressions stored in the local expression repository and their implementation was also constructed. These rules basically found to be similar to the RF2 update rules discussed in previous chapters, however, their effect on the expression stored in the local expression repository was different.

The pre-study part of the working method has also made its focus on the phases of design and implementation of the constructed prototype. Furthermore, this chapter also discusses the possible changes to SNOMED CT components.

A system that handles storing new expressions is already in use, and this system has been used as an existing or related work to what this project has planned to accomplish. As a result, this pre-study phase has also involved a study on the existing system to investigate how it is working currently. The study on the current working system has been made by attending a demo on the application followed by interviews and asking questions.

Moreover, other important tasks have also been carried out which includes:

- Existing tools used for providing programming environments
- Programming concepts, frameworks, and tools used in compiler technology.
- Articles, conference papers and other research publications covering relevant topics.

This phase provides fundamental knowledge and understanding of the existing system for completing the implementation and evaluation.

3.2.1 Expression Repository Management

The expression repository is a system that stores locally all the SNOMED CT released component files and SNOMED CT expressions (refer to chapter 2, section 2.2) defined by local Swedish healthcare centers and professionals. It was developed by two University lecturers, Daniel Karlsson and Mikael Nyström at Institutionen för medicinsk teknik (IMT), Department of Biomedical Engineering, Linköping University.

This local expression repository consists of the SNOMED CT component files; concepts, their associated descriptions, the relationships among those concepts and all related references sets as well as expressions table. The “expressions” table is a relational table in the local expression repository is used to store details about expressions of both precoordinated and postcoordinated and contains one expression per row. For each expression, a unique ID number is assigned, together with the rest information of the expression.

Key Fields	
Id	Uniquely identifier of the expression
Data Fields	
startTime	The timestamp of creation of the expression
endTime	The timestamp of expire of the expression, its default value is 'infinity'
expression	Textual representation of the SNOMED CT expression in compositional grammar form.

Table 8. Expressions relational table

When an expression is needed to be entered into the expression table, as it can be seen in the working flow figure 7 below, it is first looked up in the repository of expressions. If the expression is found in the repository, then the unique identifier associated with that expression is stored in the record, however, if the expression is not found in the repository then it is added to the repository with a new unique identifier and the new unique identifier is stored in the record. Once an expression is stored in the repository it will never be deleted. And also expressions that are no longer required can be inactivated by filling out the released date in the ***endTIme*** field in the expressions table, and its identifier will never be reused.

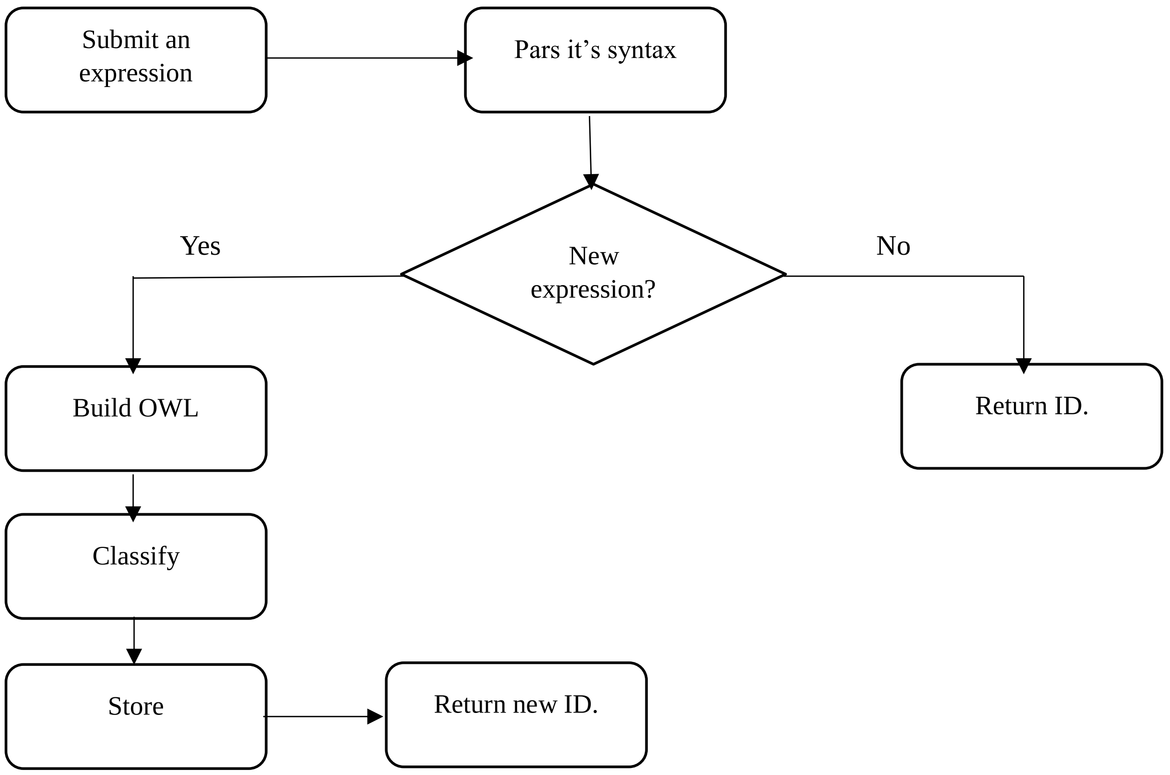

Figure 7. Working flow of the system which stores expressions

3.2.2 SNOMED CT Component changes

This thesis study has been able to identify the following changes when a new SNOMED CT version is released.

4.2.2.1 Addition of new component

Adding a new concept into the SNOMED CT terminology is one of the possible changes that could be distributed as a release file. These types of changes may include new concepts, new descriptions, new relationships between concepts and also new references sets. This new components are required to be added because they are possibly found to be important in understanding of clinical terms, disease and healthcare processes; and also could be followed by the identification of new drugs, analysis, investigations of new threats to health, as well as proposals and work provided by SNOMED CT users.

3.2.2.2 Component status change

SNOMED CT components, concepts, descriptions and relationships can be made inactive so that will change their status from active into inactive. The reason for inactivation of these components may differ from one another and could be represented by a range of status values. The following tables summarize the possible reasons why a component can be made inactive [3, section 7.4.2.2].

Reason for Component inactivation	Comment
Ambiguous component	This reason is more specifically applied into concepts and holds a meaning that the concept has either incomplete fully specified name or contains many associated terms that cannot be considered as synonymous.
Duplicate component	This can be applied to both concepts and descriptions and it defines that the component has been found to have the same meaning as another component which results in duplication of a component.
Outdated component	This can be applied to both concepts and descriptions and it defines that the component has been inactive because it is found to be an outdated concept that is no longer used.
Erroneous component	This is a situation when either the concept or description contains an error.
Limited component	A situation when either the concept or description lacks enough status value which doesn't give a stable meaning within SNOMED CT
Pending move	A situation when a component is still active but in a process of moving into a different name space. This can be applied for both concepts and descriptions.
Moved elsewhere	A situation when either the concept or description is moved into a different name space.
Inappropriate component	A situation when a description's term doesn't describe the respective concept.
Concept non-current	A situation when a description for a specific concept is still active, however the respective concept is inactive.

Table 9. Component inactivation Reason

When the status of a concept is changed from active into inactive, it also affects those descriptions as well as relationships associated with it. For example, if a concept is made inactive then it cannot have any active descriptions, similarly, some changes can happen on its relationship with other concepts. The possible effects of an inactive concept are discussed in the next section (section 4.2.4) next to the rules to handle such changes in the local expression repository.

3.2.2.3 Other changes

Most changes to a Component require it to be inactivated and replaced by a new Component with a new SNOMED CT identifier. However, a specified set of minor changes is permitted without inactivating the Component.

For example, a concept might be needed to have an update in its case styles presentation style such as the capital and small cases, its spelling. Such changes will not result in a change of the specified meaning of the concept and are not necessary to inactivate them. Similarly, changes in the case style of description's Fully Specified Name, might not also result from any effect in its associated concept, and may not be necessary to change its status.

3.2.3 Expressions Repository History Mechanism

Addition and retirements of concepts could have a significance effect on the SNOMED CT expression and thus can be categorized as the major SNOMED CT updates that are regularly released in most of its versions. In some cases, it may also include corrections to the hierarchy position of some concepts. This thesis work is going to make its focus on the two major update domains, the reason for inactivation of concepts, the *ambiguity, and redundancy* of concept inactivation [3, section 7.4.2.2] which are described in detail in section 3.6.

In this thesis work project there is no an explict method used to automatically address the reason selection of the SNOMED CT component change which are the most important for this current work. According to the above analysis findings, most of the SNOMED CT component changes affect the SNOMED CT concepts as compared to the descriptions and relationships. These was one of the reason why this work decides to deal with concept inactivation. Morover, some of the concept changes can be handled with full automation however, some of them requires human intervention. As a result of this, this theis work decides to deal with atleast one reason which can be fully automated and one reason which requires human intervention asuming that the rest component changes could possibly be handled or managed in a similar way.

3.2.3.1 Local Expression Repository History

The expression repository history is local data storage where every change in the status of concepts and terms from the expressions, as well as the release version in which the change was made, are locally stored. It is technically defined by merging the two SNOMED CT reference set mechanism discussed in chapter 3, section 3.6, which has entries for each and every inactive concept. For each referenced component a unique ID number is assigned, together with the referenced component, the target component, and its value for the reason why it has been made inactivated.

The schema for the local expression repository history table is shown below

Key Fields	
Id	Uniquely identifies the reference set members.
Data Fields	
effectiveTime	The timestamp which stores the date at which the reference set member comes into effect.
active	The state of the referenced member at a specific time.
moduleId	Specify the module which contains the reference set member.
refsetId	Specifies to which reference set does this reference set member belongs.
referencedComponentId	Specifies a reference to the SNOMED CT component being referenced.
targetComponentId	Specifies the identifier for the target component of the association.
valueId	Specifies the tagged value applied to the referencedComponentId.

Table 10. The local expression repository table

This local repository history table contains a list of inactivated concepts and stores one inactive concept per row. At a normal condition, this table can store only one replacement of an inactivated concept, however, if there exists an inactive concept with more than one replacement, the entry value allow administrators and users of the system to make informed choices for the most appropriate candidate replacement: the table relays certain metadata characteristics held in SNOMED CT for each candidate replacement. These may be used to filter or alter the presentation of possible replacements such as to highlight where a replacement exists under the same or under a different top level hierarchy from the inactive concept. It also includes as metadata a flag indicating a measure of ambiguity in any particular replacement.

3.2.4 Postcoordinated Expression Update Rules

The SNOMED CT uses historical association reference set, which maps inactive concepts into their equivalent replacements are defined under the foundation metadata concept (900000000000522004) [3, Table 216]. This reference sets are unique for each association and the two most important historical relationships [3, section 7.4.2.3] that should be considered for value set linking are discussed below.

900000000000527005 | SAME AS association reference set |:

If a Concept is inactivated because it is found to be a duplicate, the inactive Concept is linked to its active counterpart by mapping its valueId into '900000000000527005 | SAME AS association reference set |'. In other words, any duplicate concept has a 900000000000527005 | SAME AS | Association to the Active concept with the same meaning as the duplicate Concept; this value should be included in the local expression repository history table.

900000000000523009 | POSSIBLY EQUIVALENT TO association reference set |:

If a Concept is inactivated because it is found to be ambiguous, the inactive Concept is linked to one or more active counterparts by a valueId of '900000000000523009 | POSSIBLY EQUIVALENT TO association reference set |'. Thus, any ambiguous concept has 900000000000523009 | POSSIBLY EQUIVALENT TO | Association to either one or even more active concepts, which represent possible disambiguated meanings. This value should be included in the local expression repository history table.

In the following subsection, the rules used to create the local repository concept history table when components are added or changed can be seen. Generally, concepts are never removed, but their status may change from active to inactive. Thus, concepts that become inactive may sometimes subsequently have their status restored to active.

3.2.4.1 Adding a new Concept

When a newly added concept is released then:

1. A new row shall be created in the appropriate distribution table for the new component.

Id	starttime	endtime	moduleid	definitionstatusid
101009	2002-01-31	infinity	900000000000207008	900000000000074008
102002	2002-01-31	infinity	900000000000207008	900000000000074008
104001	2002-01-31	infinity	900000000000207008	900000000000073002

Table 11. Addition of new concepts

This step is as simple as inserting a new record into a relational database table. Here the new concept fields, which can be seen from the above table, take similar value to the previous records, except that the "***endtime***" field is set into "infinity". This shows that the concept is active at its early stage and can be inactivated at any time of release by changing the value infinity into the specific date at which the concept becomes expired.

3.2.4.2 Changing the status of a Concept

When the status of a concept is changed, then:

1. A new row shall be created in the concept table with:

- The ***endtime*** field which has a value of "***infinity***" shall be set into the current release date. And a new version of the concept shall not be created in a new row with the same concept information except the endtime.

Id	starttime	endtime	moduleid	definitionstatusid
101009	2002-01-31	2016-01-31	900000000000207008	900000000000074008

Table 12. Sample inactive concept record

The above table shows that the concept identified

2. A new row shall be created in the local expression repository history table with the following values field:

- referencedComponentId referring to the concept to be changed.
- TargetComponentId referring to the new replacement concept.
- ValueId referring to the reason why the concept is made inactive.
  - ➢ Ambiguous concept

If the reason for the Concept inactivation is found to be an ambiguity, then link the inactive concept to the active one by assigning the reference set value to valueId "POSSIBILITY EQUIVALENT TO". The first row of the following table shows inactive concept and its replacement due to ambiguity.

id	starttime	endtime	moduleid	referencedComponentId	targetComponentId	valueId
30828	2002-01-31	2016-01-31	900000000000207008	244745002	714307000	900000000000484002

Table 13. Sample ambigus inactive concept record

- ➢ Duplicate concept

If the reason for the concept inactivation is found to be duplication, then link the inactive concept to the active one by assigning the reference set value to valueId "SAME AS". The second row of the above table shows inactive concept and its replacement due to duplication.

id	starttime	endtime	moduleid	referencedComponentId	targetComponentId	valueId
30788	2002-01-31	2016-01-31	900000000000207008	115021007	8301000146102	900000000000482003

Table 14. Sample duplicate inactive concept record

3.2.4.3 Effect of component inactivation

The inactivation of concepts can greatly affect other components that the retired concept has been associated with. In the following section, the impact on relationships due to the status change of inactivated concepts can be seen.

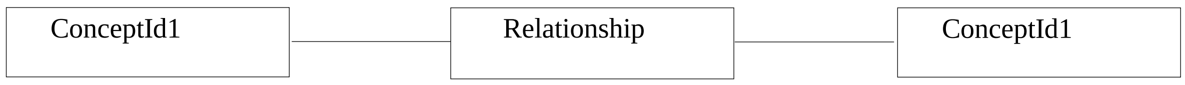

Figure 8. Relationship between two concepts

From the above figure, If the first conceptId1 is inactivated, in other words, if its status is changed from active to inactive, then any of its relationship that define/qualify the conceptId1 will also be inactivated [3, section

6.2.2]. If the inactivation reason of this concept is an “outdated” or “erroneous” or if it is inactivated without any stated reason, then the subtype relationship “IS A”, will be changed into “WAS A”.

Similarly, if the second conceptId2 become inactivated, or if its status is changed from active into inactive, then any of its relationship that defines conceptId2 must also be inactivated or replaced [3, section 6.2.2]. Here, if the reason for inactivation of conceptId2 is duplication, then the value is replaced by a new similar relationship with the current concept that has the same meaning. Also if the reason for inactivation of conceptId2 is ambiguity, then the value may be replaced by a new similar relationship with one of the ambiguous concept.

This relationship type change could have a significant impact on the related data where this hierarchy has been applied such as in the local expression repository. The detail historical Relationships by Domain can be referred in [3, Table 104].

3.2.4.4 Expression change due to concept inactivation

Let us refer back to Table 13 and Table 14, which contains sample inactive concepts due to ambiguity and duplication. Having these two tables we need to search for any expression in our expressions record if any expression contains any of the above two inactive concepts. Thus we shall update by their equivalent replacements.

Id	Starttime	Endtime	Expression
1	2002-01-31	infinity	373873005:**244745002**=(421720008+7946007)
2	2002-01-31	infinity	71388002:**244745002**=129304002,405813007=15497006
3	2002-01-31	infinity	**115021007**:363704007=(24136001:272741003=7771000)
4	2007-01-31	infinity	65801008:**115021007**=66754008,**244745002**= 25876001

Table 15. Sample expressions table which contains inactive concepts

As we can see in the above table, four expressions are appeared to contain the two inactive concepts, thus those four expressions shall be changed. In order to do this, we basically follow the implementation algorithm discussed in section 4.3. Thus, new updated expressions will be inserted following the same structure as in the table below.

Id	Starttime	Endtime	Expression
1	2002-01-31	2016-01-31	373873005:**244745002**=(421720008+7946007)
1	2016-01-31	infinity	373873005:300700305=(421720008+7946007)
2	2002-01-31	2016-01-31	71388002:**244745002**=129304002,405813007=15497006
2	2016-01-31	infinity	71388002:**300700305**=129304002,405813007=15497006
3	2002-01-31	2016-01-31	**115021007**:363704007=(24136001:272741003=7771000)
3	2016-01-31	infinity	**032207001**:363704007=(24136001:272741003=7771000)
4	2007-01-31	2016-01-31	65801008:**115021007**=66754008,**244745002**= 25876001
4	2016-01-31	infinity	65801008:**032207001**=66754008,**300700305**= 25876001

Table 16. Sample active and inactive expressions

Form the above example table we can see that the expressions containing the concept ID 244745002 has been updated by a new expression of equivalent concept ID which is 300700305. Similarly, expressions containing inactive concept ID 115021007 are updated with a new expression of a new concept ID 032207001.

4 Implementation

This chapter is going to be a further description of the previous chapter. It will discuss the update rules designed in the previous chapter in a technical perspective. The main phases that will be discussed in this chapter are the implementation/working flow of the prototype, the results and evaluation. The basic environment settings used in order to implement the work are also described in a separate section. The main results of this project are presented clearly from the implementation point of view attempting for objectivity as far as possible. The results are not analysed, discussed/evaluated here, rather that is included in the coming chapter, discussion to summarize the prototype and the overall idea of the functionality.

4.1 Work-flow of the prototype

Once the prototype that demonstrates the feasibility of the RF2 structure is at hand, the implementation work begins immediately. During this phase of the work, the design for the update rules has been be implemented using Java programming language. The implemented prototype was tested and evaluated by considering sample test cases of on some of the methods defined to implement the updates domains.

The General Pseudo-code flow for updating an expression which contains inactive concept:

1. Get all inactive concepts for a specific release date.
2. Create a local inactive concept association history table based on SNOMED CT Attribute-value Reference Set table and SNOMED CT Association Reference Set table with the following attributes,

   id,starttime, endtime, moduleId, referencedComponentId, targetComponentId, valueId
3. Fill out the local historical association table created above with appropriate value. The field, referencedComponentId refers to the inactive concept, the targetComponentId refers to the replacement of the inactive concept from the Association Reference Set table, and also valueId refers to the reason for concept inactivation from Attribute-value Reference Set table.
4. If the concept is inactivated with a reason as duplication, then, fill out the targetComponentId with the option (possibly one replacement) concept stated in the Association Reference Set table.
5. If the concept is inactivated with a reason as an ambiguity, then, retrieve all possible concept replacements from Association Reference Set table and store them in a separate file for further evaluation by local administrators. An XML file shall be generated/created to store the possible replacements for an ambiguous concept. This XML file shall be then made available for the local administrations for a suitable and single substitution of the concept.
6. Retrieve the proper concept replacements for ambiguous concepts which are chosen by local expression repository administrator and update the targetComponentId field of the local history association table.
7. Retrieve every expression from expressions table which have endtime equals to “infinity” and store them in a temporary dataset.

8. Search for the inactivated concept in the retrieved expressions data set..
9. If the inactivated concept is found in any of the expressions, then, replace them with their proper active concept replacements.
10. For every updated expression, update their endtime into the released date and create a new version of an updated expression with the same Id.
11. If the concept is redefined (primitive to Fully defined) then this will change (but not necessarily) the relationship type, as a result all the child and parent concept hierarchy will change and will result in a different meaning. Since every concept is defined by its relationship it has with other concepts, a concept's fully defined definition could be enough to differentiate the concept from its parents and siblings in the subtype hierarchy, however, If the definition is not sufficient to distinguish the concept from its parents and siblings, the concept would be defined as a primitive [3, section 4.2.1]. For such reason, when a concept is redefined, we need to reclassify our local expression repository.
12. If the inactivated concept is not found in any of the retrieved expressions table, then no expression shall be updated and Exit
13. Load the new SNOMED CT into ontology.
14. Classify the expressions again

The implementation of the algorithm will be discussed in more detail in the next discussion chapter.

4.2 Results

When update files are released, the SNOMED CT uses two ways of historical record methodologies, as a result, these two released historical files are required to be implemented in the local postgre database environment.

Once the update release files are received, the first historical relation, which is Association Reference Set table, is created in the same way as the original file structure. Next, every content of the association reference set file is copied into the table that is already prepared in the local postgre repository. This table basically consists of the following attributes;

associationreferenceset_rf2

id	effectivetime	active	moduleid	refsetid	referencedcomponentid	targetcomponentid
-	-	-	-	-	-	-

Table 17. Association Reference Set relational structure

Here, the attribute referencedComponentId acts as a foreign key of the attribute conceptId from concepts relation, thus the value of the referencedComponentId in the association_reference_set_rf2 tables should be in the conceptId of the concepts relation.

The last attribute of the association_reference_set_rf2 table which is targetComponentId indicates the equivalent replacement value for the concept which is made inactive. Hence, by matching the value of the attribute referencedComponentId in association_reference_set_rf2 and the conceptId in concepts table, it would be simple to know the concepts have been made inactive and their equivalent new value (targetComponentId).

The second method of SNOMED CT historical record that we should implement in the local postgres repository is the Attribute Value Reference Set. This file is important in that it contains information why a concept has been made inactive. In a similar way how the association Reference Set table is created, the

Attribute Value Reference Set must also be created in postgres and all the contents of the file should also be copied using an SQL query commands.

The Attribute Value Reference Set relational table consists of the following attributes,

attributevaluereferenceset_rf2

id	effectivetime	active	moduleid	refsetid	referencedcomponentid	valueid
-	-	-	-	-	-	-

Table 18. Attribute Value Reference Set relational structure

Similarly as the Association Reference Set table, the attribute referencedComponentId in the Attribute Value Reference Set also has the same usage; it is the same meaning as the conceptId attribute of concepts table.

This thesis work is mainly concerned on the two important possible values of the attribute valueId, these are

- 900000000000482003 (Duplicate)
- 900000000000484002(Ambiguous)

Ambiguous concept replacement example

Concepts

The following sample table shows a single record taken from the concepts table

Id	starttime	endtime	Moduleid	DefinitionStatusId
244540005	2002-01-31	2016-01-31	900000000000207008	900000000000074008 | Primitive |

Table 19. Semple ambigus concept record

AttributeReferenceSet

Id	effectivetime	active	moduleid	refsetid	referencedComponentId	targetComponentId
80a6b4e6-3208-5a63-ac37-e5cb5eb826da	20160131	1	900000000000207008	900000000000489007	244540005	900000000000484002

Table 20. Sample ambiguous attribute reference set record

AssociationReferenceSet

Id	effectivetime	active	moduleid	refsetid	referencedComponentId	valueId
ec55431d-4804-52ec-9bb1-3a9a26bc941e	20160131	1	900000000000207008	900000000000523009	244540005	714808004
6e0c9af7-ed72-53e6-a4a4-25c219e409d5	20160131	1	900000000000207008	900000000000523009	244540005	714807009

Table 21. Sample ambiguous Association reference set record

Duplicate concept replacement example

Concepts

Id	starttime	endtime	Moduleid	DefinitionStatusId
409437007	2004-07-31	2016-01-31	900000000000207008	900000000000074008

Table 22. Sample duplicate concept record

AttributeReferenceSet

Id	effectivetime	active	moduleid	refsetid	referencedComponentId	targetComponentId
82950ac0-237b-5ab3-8ab3-4a9eb6ea4bd0	20160131	1	900000000000207008	900000000000489007	409437007	**900000000000482003**

Table 23. Sample duplicate attribute reference set record

AssociationReferenceSet

Id	effectivetime	active	moduleid	refsetid	referencedComponentId	valueId
c82288df-5992-57b7-9744-0956bfa378ec	20160131	1	900000000000207008	900000000000527005	409437007	409436003

Table 24. Sample duplicate Association reference set record

Once the Association Reference Set and Attribute Value Reference Set relational tables are implemented in the local postgres repository, then the local history table for expressions can be created. This table is created using SQL query command and its attributes consists of the following,

- id
- starttime
- endtime
- moduleid
- refsetId
- referencedComponentId
- targetComponentId
- valueId

Once the table is created, data must be filled out and the data are normally comes from the above two historical data records. Filling out this table might need to consider some steps of the following

Step1: Fill out id, starttime, endtime, moduleid and referencedcomponentid columns in expressionrephistory relation

```
INSERT INTO
    Expressionrephistory (
             starttime,
             endtime,
       moduleid,
             referencedcomponentid )
SELECT c.starttime, c.endtime, c.moduleid, c.id FROM concepts c
WHERE c.id IN
(SELECT id FROM Concepts WHERE id IN
             (SELECT id FROM Concepts GROUP BY id HAVING COUNT(*) = 1)
             AND endtime='2016-01-31' ORDER BY id
);
```

Here, the ID filed is an auto increment and can start at some commonly agreed number. When selecting an inactive concept's from the *concepts* table the following query has been used

```
SELECT id FROM Concepts WHERE id IN
             (SELECT id FROM Concepts GROUP BY id HAVING COUNT(*) = 1)
             AND endtime='2016-01-31' ORDER BY id /*Assume the endtime is equals to 2016-01-
31
```

id	starttime	endtime	moduleid	definitionStatusId
10281002	2002-01-31	2016-01-31	9000000000000207008	900000000000074008
10281002	2016-01-31	infinity	9000000000000207008	900000000000073002

Table 25. Sample inactive concept record

The table above contains two versions of the same concept, because the id is the same for the two rows in the table. The first version of the concept is the valid version from 2002-01-31 (midnight) to 2016-01-31 (midnight), because the start time for that version is 2002-01-31 and the end time is 2016-01-31. The second version of the concept is the valid version from 2016-01-31 (midnight) and until further notice, because the start time for that version is 2016-01-31 and the end time is infinity.

In this implementation of SNOMED CT, a specific version of a concept is inactivated by setting its endtime to a value less than infinity. If a new version of the concept is inserted and has the same date as start time as the inactivated version of that concept has as end time, then there is a new and current version of the concept and the concept is not inactivated. However, if a specific version of a concept is inactivated by setting its end time to a value less than infinity and no new version of the concept is inserted, then the concept is inactivated.

Step2: Fill out valueid in expressionrephistory relation

```
UPDATE expressionrephistory e
SET valueid= at.valueid
```

```
FROM attributevaluereferenceset_rf2 at

WHERE e.referencedcomponentid= at.referencedcomponentid;
```

Step3: Fill out targetcomponentid in expressionrephistory relation which are not inactivated due to ambiguity

```
UPDATE expressionrephistory e

SET targetcomponentid= a.targetcomponentid

FROM associationreferenceset_rf2 a

WHERE e.referencedcomponentid= a.referencedcomponentid AND e.valueid!
```

At this moment, the valueId field of the table is not completely filled out, it has stored only complete records for those concepts which have a reason or valueId of 900000000000482003 (Duplicate). And this is because duplicate concepts have only one option to be replaced with so that there will not be a need of further experiment.

However, those concepts which have got ambiguity, i.e. 900000000000484002(Ambiguous), they have an option of more than one as an equivalent replacement value, hence, there is a need for human intervention in order to decide which equivalent concept does fits properly. In order to do this, all ambiguous concepts and their possible replacements shall be populated in a separate text or excel file for further discussion and decision with the local repository administrations.

Accordingly, the following SQL query command will generate the inactive ambiguous concepts in a text/excel file format so that it will be easily reported to the administrator

```
Copy (SELECT a.referencedcomponentid, a.targetComponentId

FROM associationreferenceset_rf2 a, attributevaluereferenceset_rf2 at

WHERE a.referencedComponentId=at.referencedComponentId AND at.valueid=900000000000484002) To
'/tmp/ambigous_concepts_replacement.csv' With CSV DELIMITER ' ';
```

Step4: Based on the decision made by the local repository administrators, now it is possible to fill out the targetcomponentid attribute in the expressionrephistory table from the local drive location where the file is stored.

```
void updateambigiousconceptexprepohistory(Map<Long, Long> conceptComponentMap);
```

At this moment the local repository expressions history table is complete and ready to use. The main purpose of preparing this history table was to look up what concepts has been updated, during which time and for what reason that they were updated. Moreover, this table is important in that it can be used as a reference to look at and update the locally defined expressions.

As the main goal of this thesis work is to update those expressions which contain inactive concepts, now it is the best time to get all expression from the expressions table and look for any inactive concept from the local expression history which is already implemented and update them with their equivalent replacements.

While retrieving the expressions from expressions table, it is important to limit the query with only expressions that have endtime field equals to ‘infinity’. This is because, the system that updates the expressions is not interested on expressions which are already retired and out of use.

```
Set<Expression> getAllExpressions() throws DataStoreException;
```

Those inactive concepts *referencedcomponentid* field and their equivalent replacement value *targetcomponentid* from the expressionrephistory table shall be retrived and stored in a temporary data set for comparison with the expressions.

```
 Set<ConceptReplacement> getConceptTarget(String dateStr) throws DataStoreException;
```

Next, once both the expressions and the inactive concepts are made available in a temporary dataset then searching for the inactive *referencedcomponentid* concept shall began in every expressions and if any is found then update them with their equivalent *targetcomponentid* id and store them in a temporary dataset.

```
public void updateAllExpressions() throws DataStoreException;
```

This could lead to finalizing the step by updating the expressions table of endtime attribute with the release date value, and create a new row of the same Id

```
void updateInactiveExpressions(Set<Expression> updatedExp, String dateTimeStr);
```

4.3 Environmental settings

The working method of this thesis project was carried out using the following methods:

- Java and SQL programming language

  In this study, Java programming language in Eclipse IDE has been used to implement the proposed method. SQL programing languge were also used to implement the database quires in PostgresSQL Server.

- PostgresSQL Server

  The records of SNOMED CT data and the other data have been created and handled using the PostgresSQL server. As previously mentioned, SNOMED CT consists of a large dataset. PostgresSQL Server has powerful capability for handling high workloads of similar data.

- Ubuntu desktop operating system

  The operating server used in this study is Linux family, Ubuntu desktop operating system, which can integrate Eclipse IDE and PostgresSQL Server.

4.4 Evaluation

Evaluation of the prototype is be done by considering sample test cases of the defined methods on some of the updates domains. Statistical analysis of the implemented code such as lines of code and speed or performance was not the goal to evaluate the work and has not been considered. However, in some cases, the quality of the code and the optimization of the algorithm have been reviewed to get an improved expressibility and easy understanding of the program.

5 Results and Discussion

This chapter is going to be a discussion on the method and implementation presented in chapter 3 and 4. The discussion section is presented from an implementation process perspective; along this line, the main design decisions made during the process are clearly presented and justified. It will also discuss and evaluate the testing methodologies and how they are actually applied in the work.

5.1 Results

The major findings of this work can be evaluated on the basis of the research question that we have defined at the beginning of this work.

According to the two historical association reference sets, a record in Attribute-value Reference Set indicates the reason for inactivation. Similarly a record in Association Reference Sets contains a reference/mapping from the inactive concepts to their active target concept that could possibly replace them. Based on these two association reference sets, a third local inactive concept historical association relational table has been designed to keep a historic record of the association between inactive concepts and their active replacements including their reason for inactivation. This local historical association was important in that it allows records using these concepts to be identified during information retrieval and also to facilitate mapping of the records to the appropriate active concept Identifier.

When a concept Identifier of an Inactive concept is mapped into an active concept Identifier using these associations, it is recommended that the original retired/ inactive concept Identifier be also retained. This enables future improvements or any corrections of such mappings in cases revised associations are available in a future release of SNOMED CT.

The major update domains can result those active components to be retired and as a result of this, their equivalent replacements to be added. According to this thesis work, not all kinds of change were considered important, many of them have been designated as minor and were not considered as part of this work. The major component changes that this work has paid an attention were the concept retirement/inactivation or changing the status of a concept from active to inactive.

Inactive concepts are given an inactive status and are represented as “0” in the active filed of the file. The inactive status may describe the reason for the inactivation, such as duplicate and ambiguity. When a concept is made inactive due to duplication, the equivalent component is replaced by a new component. In such situation, this work has been able to implement the local history mechanism in a simple and unambiguous manner. This is because; duplicate concepts always appear to have a single component option as their target replacement.

Similarly, in cases of concepts which are found as ambiguous, there might be several associations, in which all of these ambiguous concepts are pointing to a single concept that could represent the intended meaning of the inactive ambiguous concept. These replacements are distributed in the same way as other significant changes. However, one or more new active components are added to replace the component that was ambiguous. This work, has proposed a solution that generates a separate text file which contains a list of all inactive concepts and their ambiguity options. This was just to give the administrators the chance to choose for the active equivalent active component so that a single component can be chosen to replace.

Expressions which were considered to be active until the time of released date has been retrieved and has been checked if any of them has contain any of those retired concepts. As a result, expressions which were found having inactive concepts were made to be expired and a new version of their equivalent expression has been stored.

5.2 Discussion

5.2.1 Retrieval of inactive concepts

The first and most important task to implement the solution was to be able to know which SNOMED CT concepts have been inactivated for a specific time. Here the release date has been made to be read from an XML file which is stored together with other important resources needed to run the system. This release date can be written in the form of DD-MM-YYYY through the help of the local repository administrators. In order to retrieve the inactive concepts from concepts table, we have applied the rule that endtime of the specific concept must be equal to the released date that the local expression repository administrator has passed through the defined xml file. And at the same time, we also had to check if the concept has no newer version of it. Here, it is important that endtime equals to the released date cannot be an option to retrieve inactive concept from concepts table, this is because, a concept can get expired (endtime equals to the released date) with some other reasons as well, for example, a concept can be expired and its endtime field can take the value of the released date when it is redefined, thus another newer version of the same concept will also become available. However, this is not what exactly we are looking for and should not be taken into consideration.

5.2.2 Associating active and inactive concepts

A local history association table has been created in order to keep track of the history of inactive concept replacements as well as for easy identification of the inactive and active concept mappings. This work has made its next focus in the building of this historical association table, the reason for this was because, we noticed that this table can be used for two different table, one for storing the history of an inactive concept during which a specific concept becomes inactivated, the reason for inactivation as well as its proper replacements. The other advantage to create this table was that it could be used to easily refer to search for inactive concepts in the expressions table at the same time to find their replacements during implementation. Here, an option, once we collect the inactive concepts, it was also possible to directly search in the expressions table, however, these would result in an additional job to find the replacement for the inactive concept.

5.2.3 Identification of inactive expressions

The other most important task that this work has dealt with was, identifying which expressions needs to be expired. This has involved, first searching for expressions which are still alive or not yet expired. We were able to identify and retrieve such expressions by executing an SQL query with criteria that matches an expression's endtime equals to infinity. Here, it is important not to retrieve and re-update expressions which are already expired, because, an expired expression has already been updated by creating a new version of it and doing this for a second or more times might change the meaning of the existing hierarchal relationship in the expression. For such reason, this work has chosen to retrieve expressions with endtime equals to infinity only. Since the historical association table was already implemented, it was easy to search for inactive concepts from the table in the retrieved expired expressions. Also, it was simple to find their proper replacement so that a newer version of the expired expression can be created.

5.3 Testing and Evaluation

Based on the above discussion, several methods have been implemented in order to carry out the required tasks, and this section of the chapter will discuss the test cases implemented aligned with their results.

As the local expression repository table might consists of expressions which are made from an inactive concepts, then it is important to inactivate such expressions and create their new version. A test if the method that retrieves every expression that have end-time equals to “infinity” has been implemented in order to verify if it works as expected.

```
public final void testGetEveryExpressions()
```

In order to test this method, two other methods has been created, these methods are;

```
1.insertSampleInactiveTestExpression(startDate,endDate,inactiveExpression);
```

this method is used to insert a sample inactive expression, which means an expression that has endtime different from infinity. It takes three parameters, which are *starttime*, the time during which the expression comes into effect, *endtime*, the date at which the expression becomes expired, and the *expression* itself in a string form.

```
assertFalse("The function did not bring active expression as expected", isExists);
```

Using this method, a sample inactive expression has been inserted and the actual method was not able to retrieve the inactive expression which proves that the method works as expected.

```
2. insertSampleActiveTestExpression(startDate, activeExpression);
```

This second method is used to insert a sample active expression into expressions table, active means that an expression with endtime equals to infinity. It takes two parameters, which are *starttime*, the time during which the expression comes into effect and the *expression* itself in a string form. The endtime has been made to be inserted automatically as it is always takes fixed value which is *infinity*.

```
assertTrue("The function did bring active expression as expected", isExists);
```

Having this method defined, a sample active expression, endtime equals to infinity has been inserted, and the actual method was also executed to check if it can retrieve the active expression inserted. As a result, the assertion was successful, which shows that the actual method has correctly retrieved the active inserted expression together with the rest of other active expressions.

Every time, after a test has been asserted successfully, the sample inserted expression has been made to be removed from the database, as it was inserted for testing purpose only and is not relevant to keep with the actual records.

The local expression history repository table was mainly designed to stores all inactive concepts and their equivalent replacements. The two attribute fields are the most important fields which must be retrieved into a temporary dataset for a comparison with the expressions table records.

In the discussion chapter the method which retrieves these two attribute fields has been discussed thoroughly how it works. This method has been brought into analysis by retrieving every inactive concept and compares them if all are included in what the actual defined method has retrieved.

```
public final void testGetConceptTarget()
```

Based on this; the following method has been implemented and has been invoked to retrive all expired concepts from concepts table and store them in a temporary dataset.

```
getInactivatedConceptIdSet(Date dateStr);
```

In parallel to this method, the actual method which retrieves the inactive concepts and their equivalent replacements has been invoked to retrieve similar records from the local expression repository table and store them in a temporary dataset.

```
sample_concept_target= ds.getInactivatedConceptIdSet("2016-01-31");
```

```
conceptTargetTested = ds.getConceptTarget("2016-01-31");
```

Later in this case, both datasets has been brought into a comparison to verify if every concept which was expired during a specific date was found in the dataset which the actual method has retrieved. As the test case has been asserted successfully, it was proved that the defined method works as expected.

The other and most important method to be tested was the method that updates the ambiguous *targetComponentId* filed of the local expression repository table. Based on the above discussion the goal of this method was to fill out the *targetComponentId* field for those which have ambiguous valueId from a text file chosen by the local expression repository administrator. This was tested and evaluated as follows,

First a sample inactive concept which has been inactivated as a duplicate reason and its equivalent target value has been chosen. And this record has been stored in a temporary Java map collection as key and value ID so that it can be searched in a similar collection temporary set that the actual method has returned. Next the key value of the sample map's data has been searched if it exists in what the actual method has returned; as a result the assertion was passed successfully in that no item has been found in the real data, which work as expected.

6 Conclusions

The goal of this thesis project was to extend the existing system by improving the update management prototype in an electronic health record (EHR) for storing and handling both precoordinated and postcoordinated expressions from the ontology SNOMED CT. In order to achieve this goal, the following research question has been necessary to answer, Which SNOMED CT component changes are considered important for this work? And also what possible update rules can be proposed to manage the release updates in the existing per-coordinated and/or post-coordinated expression?

Although there is no an explictly defiend method to address or identify the important component changes, this theis work has been able to investagate that the concept inactivation change due to ambiguity and duplication could be considered the most important for the moment. An update rules has also been proposed to manage the concept inactivation which results due to ambiguity and duplication of SNOMED CT concepts. Inactive concepts and their proper replacements have been recorded in a local repository history table. This history table is also able to keep record the reason for concept inactivation and can be used as a reference to managing the version of the expressions. Ambiguity normally comes with more than one option for replacement, but this project work did not consider for ambiguous concepts with more than one option as a single target replacement concept was enough to solve the current need. However, this solution may not work always, as there might be cases in which more than one new active concepts that are required to replace a single inactive concept.

Evaluation and testing process of this work shows that the implemented solution have been able to properly manage concept inactivation updates released from SNOMED CT. However, system quality issues such as the performance and speed of the system has not been evaluated, as it was not part of the goal of this work.

Even if the local repository history table is currently handling inactive concepts, as the time goes and when the number of released files increases, its number of records will increase heavily, and this might lower the search engine performance of PostGre database server for updating expressions. From this, we come to conclusion that, It is highly recommended that this table to store only recent inactive components, thus, very old inactive concepts shall be moved out into a separate storage and can be checked only when necessary.

6.1 Future work

This work has provided an update management solution for duplicate and ambiguous concept inactivation update domains. However, these are not the only reasons that could be released from SNOMED CT; there are many other types of updates that could be received. Moreover, this work has provided a solution only for single ambiguous concept replacements which fulfills the current requirement only. Future works shall be considerate to include more futurities and functionalities that supports other update domains and also improve the current prototype to support more than one ambiguous concept replacement.

Bibliography

[1] IHTSDO, "SNOMED CT® Starter Guide," International Health Terminology Standards Development Organization, Copenhagen, 2014.

[2] IHTSDO, "Vendor Introduction to SNOMED CT®," International Health Terminology Standards Development Organization, Copenhagen, 2015.

[3] IHTSDO, "SNOMED CT® Technical Implementation Guide," International Health Terminology Standards Development Organization, Copenhagen, 2015.

[4] IHTSDO, “SNOMED CT® Expression Constraint Language Specification and Guide”, International Health Terminology Standards Development Organization, Copenhagen, 2015.

[5] IHTSDO, “SNOMED CT® Compositional Grammar Specification and Guide”, International Health Terminology Standards Development Organization, Copenhagen, 2015.

[6] IHTSDO, "SNOMED CT® Editorial Guide," International Health Terminology Standards Development Organization, Copenhagen, 2015.

[7] Tang PC, Ash JS, Bates DW, Overhage JM, Sands DZ. Personal health records: definitions, benefits, and strategies for overcoming barriers to adoption. Journal of the American Medical Informatics Association. 2006 Mar 1;13(2):121-6.

[8] IHTSDO, "SNOMED CT® Data Analytics with SNOMED CT”, International Health Terminology Standards Development Organization, Copenhagen, 2016.

[9] Benson, Tim (2012). Principles of Health Interoperability HL7 and SNOMED. London: Springer. ISBN 978-1-4471-2800-7.

[10] F. Baader, I. Horrocks, and U. Sattler. "Description logics as ontology languages for the semantic web". Lecture Notes in Artificial Intelligence, vol. 2605, pp. 228–248, 2005.

[11] I. Horrocks, P.F. Patel-Schneider, and F.v. Harmelen. "From SHIQ and RDF to OWL: The making of a web ontology language". Journal of Web Semantics, vol. 1, pp. 7-26, 2003.

[12] Ingenerf, Josef, and Thomas Beisiegel. "A version management system for SNOMED CT." *Studies in health technology and informatics* 136 (2007): 827-832.

[13] Kim, Tae Youn, Amy Coenen, and Nicholas Hardiker. "A quality improvement model for healthcare terminologies." *Journal of biomedical informatics* 43, no. 6 (2010): 1036-1043.

Glossary

Attribute It express characteristics of concepts.

Component Refers to anything in SNOMED CT identified by an SCTID. Examples of Components include Concepts, Relationships, Descriptions, and Subsets.

Concept A clinical idea to which a unique Concept Identifier has been assigned.

Concept ID/Code: The unique identifier (code) for each SNOMED CT concept. Refer to the SNOMED Technical Reference Guide for a full explanation of how this identifier is structured.

Concept Status A field in the concepts table that specifies whether a concept is in current use. Values include "current", "duplicate", "erroneous", "ambiguous" and "limited".

Description ID: An SCTID that uniquely identifies a Description. Refer to the SNOMED Technical Reference Guide for a full explanation of how this identifier is structured.

Hierarchy An ordered organization of concepts.

History Mechanism SNOMED CT includes some information about the history of changes to concepts and descriptions.

RelationshipID A SCTID that uniquely identifies a Relationship between three concepts: a source concept (ConceptID1), a target concept (ConceptID2), and a relationship type. Each row in the Relationships Table represents a relationship "triplet" (ConceptID1 – RelationshipType - ConceptID2) identified by a RelationshipID. Refer to the SNOMED Technical Reference Guide for a full explanation of how this identifier is structured.

Term A text string that represents the concept. The term is part of the description. There are multiple descriptions per concept.

Value A value which is assigned to an attribute to express characteristics of a concept.