INTEGRATING TECHNOLOGIES, METHODOLOGIES, AND DATABASES INTO A COMPREHENSIVE TERMINOLOGY MANAGEMENT ENVIRONMENT TO SUPPORT INTEROPERABILITY AMONG CLINICAL INFORMATION SYSTEMS

by

Shaun Cameron Shakib

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Biomedical Informatics

The University of Utah

December 2013

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Shaun Cameron Shakib**

has been approved by the following supervisory committee members:

Stanley Huff	, Chair	**27 April 2012** Date Approved
Bruce Bray	, Member	**27 April 2012** Date Approved
Lisa Albright	, Member	**27 April 2012** Date Approved
Lee Min Lau	, Member	**27 April 2012** Date Approved
Roberto Rocha	, Member	**27 April 2012** Date Approved

and by **Julio C. Facelli** , Chair/Dean of

the Department/College/School of **Biomedical Informatics**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

Controlled clinical terminologies are essential to realizing the benefits of electronic health record systems. However, implementing consistent and sustainable use of terminology has proven to be both intellectually and practically challenging. First, this project derives a conceptual understanding of the scope and intricacies of the challenge by applying informatics principles, practical experience, and real-world requirements. Equipped with this understanding, various approaches are explored and from this analysis a unique solution is defined. Finally, a working environment that meets the requirements for creating, maintaining, and distributing terminologies was created and evaluated.

I dedicate my dissertation work to my family, friends, and colleagues. To my wife Sabine, children Thomas and Sarah, parents Nasser and Jenial, sister Julie, and friends and mentors Lee Min and Lam. Thank you all for your patience, encouragement, and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

The complexity of modern medicine and the volume of available and relevant information that must be processed exceeds the limits of the unaided human mind.[1] Man is not perfectible. No amount of training will make humans into flawless information processors. As a consequence, U.S. physicians provide the recommended care to their patients only about 60 percent of the time, according to a set of reports from the Agency for Healthcare Research and Quality (AHRQ).[2] One large and commonly cited study in Utah and Colorado found that adverse events occur in 2.9 percent of hospitalizations.[3] Another similar study in New York assessed the adverse events at 3.7 percent.[4] Respectively 6.6 and 13.6 percent of these adverse events led to death. Using U.S. hospital admissions in 1997[5] (the approximate timeframe for these studies), the percentages above imply that each year between 44,000 to 98,000 patients die from adverse events of which a very high percent are potentially preventable. In addition to the tragedy of preventable deaths and complications, there is economic impact associated with error and inefficiency. The increase in the cost of health care has exceeded the increase in the consumer price index for the last four decades, and the cost of health care now represents approximately 17.9 percent of the U.S. gross domestic product.[6]

Biomedical Informatics has an important role in addressing this challenge. A key technology domain that falls under Biomedical Informatics is the Electronic Health Record (EHR). Study after study has shown that computerized decision support can

decrease errors, improve patient safety, and decrease costs.[7-9] Our assumption is that this type of computerized decision support and information sharing can only operate reliably on coded and structured medical data. As such, regardless of how the clinical data are collected, they must be either maintained or transformed to a state that is structured and encoded. The goal of this Terminology Management Environment (TME) project is to create a vocabulary server with supporting tools and services, which allows for the collection, use, and exchange of structured, encoded, standardized data. This dissertation:

- establishes why coded terminology is essential for realizing the benefits of EHR systems (Section 2)
- defines semantic interoperability and what it means for health care delivery (Section 3.2)
- describes the challenges and barriers that exist for consistent and sustainable use of coded terminologies in EHR systems (Section 5)
- describes infrastructure options for integrating and managing standard terminologies (Section 7)
- elaborates the detailed requirements for coded terminology used in EHR systems (Section 8)
- develops a strategy and detailed design for creating, maintaining, and distributing coded terminology in a TME that meets the interoperability requirements of EHR use (Sections 9-10)
- evaluates a working system that meets the requirements for creating, maintaining, and distributing coded terminologies used in EHR systems (Section 11).

2. WHY INTEROPERABILITY IS ESSENTIAL TO REALIZE EHR BENEFITS

The need for coded data in the EHR and the specifics around how the data are encoded are dependent on what users want to achieve with the data. Goals for the use of data include:

1. Sharing of data—Implementing a standard method for exchanging clinical data will help to address costly, custom interfaces among disparate systems and the inability to exchange data in a computable form between different care areas, facilities, and organizations. Sharing clinical data must happen at many different levels and for different purposes: communicating reportable diseases to state public health, the United States Centers for Disease Control and Prevention (CDC), and the World Health Organization (WHO); integrating "best of breed" systems within a single facility; transferring a patient's electronic medical record to a new facility; communicating with insures and the government for billing and reimbursement purposes; making data available for research; integrating a patient's personal health record.[10] These are a few examples of the reasons why clinical data are shared. Some are to improve patient care, governments or insurers mandate others, and some drive an organizations business operations.
2. Executing advanced decision support logic and alerts—Humans are good at applying knowledge, context, and past experience to understand the "big picture,"

plan, adapt, and make decisions. However, the consistency and reliability of human decision-making is not perfect.[11] Computer-based decision support and alerting tools are meant to augment human decision-making. They take advantage of a computer's superior ability to quickly and accurately store and retrieve data[12] and apply computable rules to supply warnings and recommendations to decision makers.

3. Sharing of decision support logic and medical knowledge—Knowledge applicable to clinical domains is vast, ever changing, and context-dependent. It is unlikely that a single organization could collect, represent, and maintain all of this information. Creating an environment where this type of knowledge can be authored and distributed among multiple different organizations/systems allows experts to focus on the part of the problem they are best suited to address. They can share and compare approaches to come up with a comprehensive and higher quality model.

Because computers are incapable of automatically establishing context or accounting for variation in text, none of the above can be accomplished without coded and structured data. Computers can only provide accurate and timely information if the underlying data have been normalized to address variations inherent in natural language and free text. The types of variation include those represented orthographically: valid synonyms, abbreviations, lexical variants; and nonorthographically: awkward/incorrect grammar, misspellings, nonstandard codes or abbreviations.[13]

As an example of orthographic variation, the concept "varicella-zoster virus" has all of the following valid synonyms and acronyms: VZV, varicella virus, zoster virus, herpes

zoster virus, human herpes virus type 3, HHV-3, and chickenpox virus. As an example of nonorthographic variation, one study that reviewed 6 million chief complaint records found 379 different misspellings for the word "vomiting."[13] Encoding data using a controlled vocabulary normalizes the text variation and allows computers to operate against codes that represent "agreed-to" definitions of data.

In order for both legacy and future EHR systems to achieve the ultimate goal of providing significant and measureable improvements to population health, it is essential that those systems utilize coded, standard terminology.[8] Every secondary use of data, whether it be summarizing, analyzing, exchanging, or transforming the data, is completely dependent upon the data being structured, normalized, and interoperable.

3. THE STATE OF THE ART WITH REGARD TO INTEROPERABILITY

Having established the importance of interoperability, this section more rigorously defines it and describes the current state of efforts toward achieving interoperability by:

1. Establishing the role of terminology in achieving interoperability (Section 3.1).
2. Defining the layers and levels of interoperability (Section 3.2).
3. Describing the types of code systems and their implications for interoperability and TME design (Section 3.3).
4. Summarizing the existing vocabulary server technologies and where things are failing today with regard to the collection, storage, and exchange of interoperable clinical data (Section 3.4).

3.1. Terminology Use in Interoperable Data Exchange

Interoperability is a broad concept that has many levels and must be addressed across systems architecture, content, and hardware. The TME facilitates a specific level of interoperability at a specific system layer, which will be described in the following sections. Simply defined, interoperability is the ability of one computer system to exchange data with another.[14] However, there are many layers, from the physical to the application, that require interoperability standards and there are levels of interoperability that define the degree of semantics captured in the data that are exchanged. The role for

interoperability in health information systems is to allow systems to do more with data than merely capture, store, and display it in a single, standalone system. Advanced levels of interoperability empower decision makers with actionable information by enabling their computer systems to exchange meaningful data and to interpret and act upon shared data and knowledge.

3.2. Layers and Levels of Interoperability

The interoperability layers are defined by the Open Systems Interconnection Reference Model and define interoperability standards for everything from the physical layer (electrical and physical specifications for devices) to the application layer (semantic conversion between associated application processes).[15] The TME operates at Level 7 or the application layer in this model.

At that application layer, there are three different levels of interoperability. As defined by the National Committee on Vital and Health Statistics (NCVHS) in their July 6, 2000 Report on Uniform Data Standards for Patient Medical Record Information,[16] the interoperability levels are:

1. "Basic" Interoperability
    - Messages can be exchanged between systems, but not interpreted.
2. "Functional" (Syntactic) Interoperability
    - Messages can be exchanged between systems and interpreted, but only to the level of the data fields; in other words, the message structure/format is defined and understood by both sending and receiving systems, but there is no common/computable definition of the data within the fields.
3. "Semantic" Interoperability

- Messages can be exchanged between systems and interpreted by both systems; the structure of the message is defined and the meaning of the data within the data fields is understood. The data can be acted on by the receiving system automatically.[16]

Standard terminologies address semantic interoperability by providing a common set of codes with definitions/descriptions that can be referenced to encode data across disparate systems. The TME helps to implement semantic level interoperability by operationalizing standard terminologies in EHR systems.

3.3. Terminology Use in EHR Systems

Health care delivery operates at the patient-centric and population level and terminology management and interoperability are important at both levels. A patient-centric view of clinical data does not obviate the need for exchange of normalized and standardized data. Aggregating a single patient's data over time requires that the data be collected from multiple different facilities and source systems such as COTS pharmacy systems and laboratory information systems (LIS), and combining those data in a nonredundant and meaningful way.

For example, a patient's list of allergies at Facility A includes shellfish. At Facility B the same patient's allergy list includes shrimp and penicillin, and at Facility C the list is PCN, peanuts, and chocolate. Automatically consolidating the lists and recognizing related and redundant items is enabled through semantic interoperability. This is important for the accurate and concise representation of data in a patient-centric system. Even viewing a patient's data at a single point in time can benefit from interoperability, in

that it gives a clinician the ability to connect data in the patient's record to external knowledge sources or decision support and alerting tools.[17]

Take this hypothetical example: A severely ill infant with suspicion of meningitis is transferred from a rural medical facility to a pediatric hospital. The health information systems at the two facilities are not interoperable and cannot share information regarding the patient. At the rural facility, a Cerebral Spinal Fluid (CSF) sample was collected and laboratory tests performed, but when the baby arrives at the pediatric hospital, none of the previous results from the rural hospital are available to the new physician. The parents insist that testing was performed and may even know, for example, that bacterial meningitis was ruled out. But it does not matter, the new physician must start over and this begins with the collection of a new CSF specimen.

Collecting a CSF sample on an infant can be a traumatic procedure for both the clinician and the child and can have complications that worsen the outcome. Not only does the duplicate testing introduce additional risk, pain, and expense, but also causes a delay in this time critical, life threatening situation.[18] One study found duplicate testing, which they defined as a repeat of the same test within a 12-hour period, in 32% of the cases examined.[19] The example above describes systems that lack interoperability. The records in these systems are incomplete, ambiguous, or not electronic. The following describes information exchange through progressively higher levels of interoperability previously described:

1. "Basic" Interoperability
    - The physician must search through printouts or multiple screens of data to see if the necessary laboratory test results are available.

2. “Functional” Interoperability
    - The receiving system is able to recognize a laboratory test result (versus a medication order, for instance) in another system, but results for a particular test cannot be identified and the values cannot be interpreted or aggregated with results in the receiving system.
    - Automated alerting for duplicate testing, adverse events, or infectious diseases will not work for data collected from other systems.
3. “Semantic” Interoperability
    - The receiving system is able to recognize and interpret a particular laboratory test and its results from another system.
    - Critical information vital to accurate decision-making is readily available to automated decision support and the clinician without additional time, expense, or hardship for the patient.

Beyond its value at the patient-centric level, semantic interoperability has significant implications at the population level. Disease surveillance and early alerting depends heavily on the timely and accurate exchange of clinical information.[20] Without basic interoperability, humans would be required to transfer information person-to-person through phones, fax machines, email, etc. With syntactic interoperability, data from multiple disparate sources can automatically be collected in a single repository, but a human or additional tooling would be required to review, normalize, and interpret the data. With semantic interoperability, the system can recognize a specific disease occurrence without human intervention, and automatic triggers can be set up for alerts.

In addition to the benefits to health care at the patient and population levels, the economic value of interoperable data exchange has been estimated in two recent studies. The state of Maine commissioned a cost-benefit analysis as part of the feasibility study for the Maine Health Information Network Technology (MHINT) system. The analysis concluded that a health information exchange could potentially save the state $42.3-$58.4 million annually in health care costs.[21] In 2005, Walker et al. from the Center for Information Technology Leadership in Boston, MA assessed the value of electronic health information exchange at the national level. They concluded that fully standardized health information exchange and interoperability could potentially result in a net value to the U.S. of $77.8 billion annually.[22]

3.4. Interoperability Standards Efforts to Date

Recently, health care providers have been incentivized through the HITECH Act (part of the American Recovery and Reinvestment Act (ARRA) passed by the U.S. Congress in 2009) to implement EHRs that can achieve "meaningful use." Many of the meaningful use criteria are either enhanced by, or can only be achieved through, standardization and interoperability.[23] In an effort to assess gaps and areas of overlap with regard to standards for Health Information Exchange (HIE), the office of the National Coordinator for Health Information Technology (ONC) has taken on as part of its mission "providing leadership in the development, recognition, and implementation of standards and the certification of Health IT products."[24]

As early as 1994, Rocha et al. first described several of the key components necessary for the practical implementation of a vocabulary server in "Designing a Controlled Medical Vocabulary Server: The VOSER Project."[25] The VOSER Project

introduced the concept of knowledge representation through three components: controlled medical vocabulary, knowledgebase, and medical information models. It also proposed a strategy for integrating multiple source terminologies.

In 1998, Cimino published "Desiderata for controlled medical vocabularies in the twenty-first century" a meta-analysis that compiled best practices with regard to the design of a controlled medical vocabulary.[26]

Since that time, there have been commercial, academic, and government efforts to develop tools, services, and content to address the terminology challenges. The following are vocabulary servers that support mappings between LITs and standards and among standards:

- The Apelon Distributed Terminology System (DTS) is a commercial suite of tools for terminology maintenance and development, but contains no content.[27,28]
- The 3M Healthcare Data Dictionary (HDD) is a runtime vocabulary server that houses a concept-based controlled medical vocabulary with mappings among local terminologies and standards. It contains a "tightly federated" set of terminologies and because of this, it is limited with regard to flexibility of mappings and source native data that it can represent.[29]
- Columbia's Medical Entities Dictionary (MED) is also a concept-based controlled medical vocabulary that took a frames-based approach to represent concepts and mappings in a semantic network.[30,31]
- The Health Language Language Engine® (LE®) is a commercial terminology management environment that allows for content authoring and management of external code systems.[32]

- Clinical Architecture's Symedical® Server is a commercial terminology management environment that allows for the management of external code systems and the creation and maintenance of mappings.[33]

In addition, there are large collaborative efforts to develop, publish, and load health care terminology:

- The Lexical Grid (LexGrid) Project created by the Mayo Clinic provides support for the distributed authoring of terminology and a method to export the content in multiple standard formats—no content.[34]
- Similarly, Biomedical Grid Terminology (BiomedGT) is an open terminology for translational research. It was initially populated with content from the National Cancer Institute (NCI) Thesaurus and made available in a wiki format to facilitate collaborative development—not runtime.[35]

Efforts towards interoperability have accelerated over the last few years. Standard Development Organizations (SDOs) such as HL7 have taken the lead in providing interoperability definitions, and the HL7 version 2.x messaging standard is implemented in information systems worldwide.[36] Most, if not all, clinical information systems are currently compliant to the version 2.x messaging standards, making syntactic interoperability an attainable reality.[37] However, semantic interoperability is still uncommon, for many reasons:

- Clinical data that are not digital; a large volume of data collection is done via transcription to either digital free text or paper and small facilities are either primarily or entirely on paper.

- Digital data that do not encode discrete data elements; many EHR systems are document management systems that store scanned images of paper records with very little to no discrete, structured data.
- Coding that is not granular enough; in the U.S., coding is done primarily for billing and reimbursement purposes, using classification systems like ICD-9-CM. Classification systems lack the granularity to achieve semantic interoperability of clinical data without information loss.[38]
- Challenges implementing standards; there are multiple standards and properly implementing them requires significant expertise.
- Branching ideas/strategies; in the field of knowledge representation, there have been both academic and pragmatic discussions with regard to how to represent clinical data. One such dialogue occurred between Barry Smith (From concepts to clinical reality: An essay on the benchmarking of biomedical terminologies) and James Cimino (In defense of the Desiderata). Smith repeatedly states that his theory is based on "reality" and goes on to describe an approach for knowledge representation involving the use of ontology and "universals,"[39] which are essentially rigorous and formally defined concepts. Cimino rebuts, describing how concepts and "universals" can coexist and why there is a practical need for concepts as an intermediary between instance data and "universals."[40]
- Issues with HL7 standardization and interoperability; if you have seen one HL7 interface you have seen one HL7 interface. That is, the HL7 V2.X family of standards allows a high degree of optionality, and there is a lack of standardization of terminology.

- Issues with Vendor Technology; the approach of each vendor is different: a Cerner system cannot communicate with a Siemens system without translation. Use of local master files, local term tables, and local terminology servers leads to divergent terminology use. Standard practice is that even within customers of the same vendor, each installation makes up its own attributes, terms, and codes.

Terminology is often the last problem to be recognized and addressed in the effort to move to digital health records. Canada, for example, is working to create a pan-Canadian EHR. The Canadian government has described a "blueprint" for their health information highway, which defines the infrastructure for data exchange. A key component of the blueprint is the Health Information Access Layer (HIAL). The HIAL is an interface specification for EHRs and all Canadian regions and provinces are expected to connect to the HIAL.[41] Canada is several years into this project and has built a great deal of infrastructure and defined many of the standards to be used in messaging. However, they have yet to define a practical strategy for regions and provinces to implement and maintain standard terminologies like LOINC and SNOMED CT. Interoperability showcases and "connectathons" in Canada are based on use cases that begin with LOINC and SNOMED CT, neglecting to recognize that most legacy systems are incapable of providing LOINC or SNOMED CT codes.

The challenge of interoperable standards implementations remains. Most EHR development predates current semantic interoperability requirements, and most health care organizations already have clinical information systems in place, a significant financial as well as operational investment. Replacing these legacy EHRs is not a viable option for many organizations. Considerable patient data have already been collected,

encoded, and stored, using the LITs in these information systems. These historical data are critical to continuity of care and optimal outcomes for patients. Switching to encoding new patient data with a different terminology, even a standard, would mean these historical data are no longer compatible with the new data. Thus, while the new data encoded with a standard terminology are semantically interoperable with other external data also coded to the same standard, ironically, the organization's own historical data would not be.

Communication of interoperable data is essential to achieve the best efficiency and outcomes in health care, both for the patient and the population. The examples above illustrate how the lack of semantic interoperability results in duplicate clinical effort (e.g., laboratory testing), additional human intervention (e.g., manual data entry and review), and difficulty in performing advanced decision support and alerting functions. And the cost savings estimates further justify the expense and effort of implementing interoperable health care systems. A practical migration path is needed to help health care organizations gain the ability to achieve semantic interoperability, without imposing undue burden on the organization and that does not result in the loss of historical patient data.

4. TERMINOLOGY THEORY AND STRUCTURE

Having established the importance of semantic interoperability and the role terminology has in achieving it, the following sections define categories of terminology and their characteristics. There are broad categories of terminologies/classification systems used within clinical information systems. Each of these source code systems has different behavior and design characteristics. These differences place numerous and sometimes conflicting demands on the TME. The TME design must be flexible enough to incorporate these different categories of code systems without loss of information. There does not appear to be general consensus on definitions of these categories. So, for the purposes of this dissertation, descriptions will be provided below.

4.1. Local Interface Terminology

An interface terminology has been described as a collection of terms and phrases used to enter data in a computer system.[42] For the purpose of this discussion, the definition of interfaces will be broadened to include both end-user and machine interfaces. Local Interface Terminologies (LIT), or local terminologies, are widely variable, "home-grown" terminologies that supply the codes and displays used within many health information systems. Often these types of terminologies are not well behaved. Examples of problems with local interface terminologies include:

- Not concept-based; term-based resulting in denormalized data because of orthographic and nonorthographic variants; e.g., dyspnea, shortness of breath, disnea, and SOB are various valid and invalid terms that can be used to represent one concept.
- Code removal and/or reuse; codes are deleted and/or codes for deleted or deprecated (inactivated) terms or concepts are reassigned to new terms or concepts. This problem occurs in local and standard code systems; e.g., CPT code "0002T" for "Endovascular repair of infrarenal abdominal aortic aneurysm or dissection; aorto-uni-iliac or aorto-unifemoral prosthesis" was deleted in December 2003 and did not appear in subsequent versions of the code system[43,44]; NDC "00074433501" was for the concept "Liposyn (Fat Emulsions), 10%, IV Solution, 200ml Bag" before July 2002. It was deleted and reintroduced as the concept "Paclitaxel (Paclitaxel, Semi-Synthetic), 6mg/ml, Vial, Injection, 5ml Vial" after July 2002.[29]
- Lack of version control; updates can be ad hoc and often there is no rigorous mechanism for versioning or standard maintenance protocols.
- Ambiguous content; content may have a nonunique or unclear description and no formal definition; e.g., terms such as "Blue" in a specimen domain with no definition.

In many cases, there are not only challenges with regard to how the LIT was created and is maintained, but also with how systems are utilizing the terminology. The LIT is often housed within a table that is referred to as a master file (see Table 1). Master files are referenced by applications to encode/decode data. The LIT may be hardcoded into

the systems, such that replacing the terminology requires rewriting the software. It is also important to consider all of the data collected previously. Data that have been encoded using a LIT are referred to as legacy data.

4.2. Standard or Reference Terminologies and Classification Systems

A standard terminology is one that has wide industry acceptance or use. Standards are obtained from a variety of efforts, cover different domains of clinical and nonclinical content relevant to the EHR, and serve various purposes. Currently, no one terminology or classification system contains everything that is needed for the EHR. Encoding a longitudinal patient record in the EHR requires multiple standards.

Examples of standard terminologies include:

- Systematized Nomenclature of Medicine-Clinical Terms (SNOMED CT®) is a comprehensive clinical terminology. The U.S. Federal Government purchased a perpetual license for the core SNOMED CT® in 2003. SNOMED CT is maintained by the International Health Terminology Standards Development Organisation (IHTSDO).[45]
- Logical Observation Identifiers, Names, and Codes (LOINC®) is a terminology for laboratory tests, results, and clinical observations. It is developed and maintained by the Regenstrief Institute.[46,47]
- RxNORM and RxTERMS[48] are reference terminologies for human clinical drugs that are maintained by the National Library of Medicine (NLM) and distributed via the Unified Medical Language System (UMLS).[49]

- Current Procedural Terminology (CPT™) is a proprietary standard used to encode medical services and procedures. CPT is maintained by the American Medical Association (AMA).[44]
- Examples of classification systems that are considered standards for billing and reimbursement include the International Classification of Diseases, 9th and 10th Editions, Clinical Modification (ICD-9-CM and ICD-10-CM),[50,51] and several different Diagnosis-Related Group (DRG) systems.
- Standards are also developed by consensus industry effort, such as the terminology authored and distributed by Health Level 7 (HL7) to support the HL7 version 2.x and version 3 messaging standards.[36]

4.3. Federated Terminology

Because different code systems are required to support an EHR and since standard, commercial-off-the-shelf (COTS: terminology that is sold as part of another product or as a stand-alone system) and legacy terminologies can be overlapping, it is necessary to have a practical strategy for integrating and maintaining them. Recently, some have described the notion of a single terminology that harmonizes both concepts and knowledgebases/ontolgies from multiple sources (and most importantly, for a particular purpose), as a "federated terminology." Consider the following example:

Objective: Author an alert within a clinical system to fire if a patient who is allergic to penicillins is prescribed a drug that has any of the penicillins as an ingredient.

Challenge: To avoid the ambiguity and variability of free text, the rule is authored against a code system. The immediate problem is which code system to

use. Drugs can be encoded using Medispan, RxNORM, SNOMED CT, National Drug Code (NDC), etc. Each code system has its own way of describing clinical drugs and hierarchies for organizing them.

Solution: Author the alert against a concept-based federated terminology that normalizes the codes and relationships across multiple source terminologies.

Result: Regardless of which terminology was used to collect the drug information, the alert can recognize the concept of the class of penicillins and traverse a "has ingredient" relationship to identify all drugs that have a penicillin as an ingredient.

The UMLS is an example of a federated terminology, but it can be thought of as a "loose federation." The purpose of the UMLS is to act as a reference that compiles multiple code systems and makes them available in one place. UMLS assigns a single Concept Unique Identifier (CUI) for "synonymous" concepts in the various code systems it incorporates.[49] Because it is meant as a reference and not a product to perform automated data transformation, the UMLS uses a loose definition of synonymy.

The 3M Healthcare Data dictionary (3M HDD) is a product that implements a "tight federation" by creating a "single" concept-based controlled medical vocabulary from integrating multiple source terminologies into a single schema. The HDD first assigns a Numeric Concept Identifier (NCID) to a defined concept, and then maps the identifiers of clinically equivalent concepts from the multiple source terminologies to this NCID.[29] Concepts in the HDD can be missing from standard terminologies. Redundant concepts from source terminologies, or even within a single terminology, are not duplicated in the HDD, but would have the identifiers all mapped to a single concept in

the HDD. Similarly, HDD concepts are organized in multiple hierarchies and relationships. The source terminology's native hierarchies and relationships can also be contained in the HDD and indicated as such, similar to the inclusion and indication of native identifiers and descriptions for concepts. The HDD's federated terminology is used in application programming, such as the alert authoring example above, which will be encoded using the NCID for Penicillins. The relationships of the NCID are used to trigger the alert when any medication containing a Penicillin as an ingredient is ordered, and the mappings of the NCID for Penicillins are used to "translate" to the required standard terminology (e.g., SNOMED CT or RxNORM) for data exchange.[29]

In contrast, the TME houses a “flexible federation.” The design of the TME allows for links among integrated terminologies to be purpose-driven and for multiple different types of mappings/links to be established. The exact design that accommodates this behavior in the TME will be shown later.

Table 1: Local Interface Terminologies (LIT). Sample codes and descriptions from a local lab system master file.

Antibiotic Drug Code	Description
ACOM	ANTIBIOTIC COMMENT
AM	AMPICILLIN
AMI	AMIKACIN
AMI10	AMIKACIN 10
AMI2	AMIKACIN 2 MCG/ML
AMI4	AMIKACIN 4 MCG/ML
AMI6	AMIKACIN 6 MCG/ML
AMI8	AMIKACIN 8 MCG/ML
AMPB	AMPHOTERICIN B
AMS	AMPICILLIN/SULBACTAM
AMX	AMOXICILLIN
ASEN	AFB SUSCEPTIBILITY
ASENS	ANAEROBIC SENSITIVITY
ATM	AZTREONAM
AUG	AMOXICIL/CLAVULANATE
AZL	AZLOCILLIN
AZM	AZITHROMYCIN

5. CHALLENGES UTILIZING TERMINOLOGY

To this point, the role of coded clinical terminology and the current state of the art with regard to implementation of terminology in clinical systems has been discussed. Going forward, discussion will focus on the particular approach taken in this project to address the challenges creating and maintaining a large centralized vocabulary server.

As we have discussed, health care organizations are beginning to recognize the many advantages of semantic interoperability. However, to achieve semantic interoperability, political, economic, and technical challenges must be overcome. While recognizing these additional obstacles, the focus of this dissertation will be to address the technical challenges implementing and maintaining a federated terminology in the TME.

There are a wide variety of code systems that must be incorporated in the TME and each requires its own special considerations. Several of the code sets that are in common use and have become "de facto" standards in the U.S. were originally designed for billing and reimbursement, inventory, or summary analytics.[52] Because of this, they are missing important attributes necessary for a controlled medical vocabulary intended to collect clinical data in a longitudinal patient record. For example, these code systems may lack the necessary level of granularity to prevent information loss or proper maintenance and version control policy to avoid practices such as code reuse or deletion. Where the source code system is deficient, the federated terminology housed in the TME must compensate.

The challenges integrating this heterogeneous content into a single system that can supply fit-for-purpose content can be categorized in the following way:

- Integration Challenges – these are challenges normalizing the content for various sources to a single concept-based terminology
- Maintenance Challenges – these are challenges that come about because of the size, complexity, and ever-changing nature of both code systems and clinical knowledge
- Utilization Challenges – these are challenges specific to making use of this content within clinical systems.

Each of these categories will now be further elaborated.

5.1. Integration Challenge

The following subsections will describe the challenges specific to integrating multiple heterogeneous source code systems in the TME.

5.1.1. Multiple formats, update schedules, and content types.

Source code systems are distributed in multiple different formats (see Table 2). Some are only disseminated in printed form while those that are available digitally may be available as one or a combination of the following formats: Portable Document Format (PDF), text delimited files, databases, excel spreadsheets, etc. It is important that this source data, regardless of format, be archived and indexed so that it can be referenced in the future. The content must also be extracted and transformed in a consistent manner so that it can be compared against previous versions, loaded, and mapped in the federated terminology. This requires having custom rules for the transformation and loading of data from various sources and various clinical domains.

5.1.2. Differences in granularity. Granularity describes the level of detail at which a content (code attribute) is described. Finer granularity means a higher level of detail. Generally, classification systems are less granular than terminologies (see Table 3), but even among different terminologies, granularity can vary. Also, within a single terminology, there may be varying levels of granularity.[53] The TME must account for these differences in granularity without discarding information from more granular content or implying additional information for less granular content.

5.1.3. Pre- and postcoordination. "Precoordinated" terms are "molecular" terms that combine multiple concepts or a concept and its modifiers in a single phrase. The precoordination can be done in adherence to an underlying terminology model such as LOINC or in the absence of a specific terminology model. An example of a precoordinated term would be "lower left eyelid laceration." The same "molecular" term can also be generated in "postcoordinated" fashion by assembling the "atoms" that make up the term: lower, left, eyelid, and laceration using a defined syntax or language such as the GALEN Representation and Integration Language (GRAIL).[54]

The pros and cons of precoordination are:

1) **Pro:** Only the logical or "real" precoordinated terms are created and available; in other words, illogical combinations such as "lower left hair laceration" can be prevented. It is much more difficult to do this in a postcoordination approach.

   Con: "Combinatorial explosion" or the notion that attempting to create every logical combination of all the "atoms" explodes into an unmanageable, vast (but not infinite) number of precoordinated concepts.[54]

2) **Pro:** Precoordination provides a fixed reference for an information model. There is only one canonical way of expressing the molecular term, instead of having multiple ways of generating the same phrase/expression.

   Con: Not as flexible or easily extensible—constant need to create new terms as opposed to just assembling known concepts in a new way.

3) **Pro:** The messaging machinery for storing, interpreting, and communicating a precoordinated concept is not complex. It does not have to reference rules or a syntax for the composition or decomposition of the terms.

   Con: Precoordination can result in the inclusion of properties or modifiers in a molecular term that should instead be inserted in other, separate fields in an information model/message.

The pros and cons of postcoordination are the inverse of those of precoordination, but for the sake of completeness, they are described below:

1) **Pro:** Avoids "combinatorial explosion." It is only necessary to create the "atoms" and not every conceivable way that they could be combined.

   Con: It is possible to create illogical combinations of concepts such as "fractured hair." Preventing illogical concepts requires significant effort defining the properties of concepts and, based on those properties, which combinations they can participate in. For example, define hair as a body structure that cannot be fractured or enumerate all body structures that can be fractured.

2) **Pro:** Easier to maintain the referenced terms because there are fewer.

Con: The messaging machinery for storing, interpreting, and communicating a precoordinated concept is more complex. It must reference rules for the composition or decomposition of the terms.

3) **Pro:** More readily extensible and flexible. Assembling known concepts in a new way can create new terms.

Con: There is the risk of storing the same molecular term in multiple different ways, because of the many different ways of defining or parsing out "atoms." For example, "lower | left | eyelid | laceration" or "lower left | eyelid laceration," or "lower | left | eyelid laceration," etc.

The TME must have a means to compose atomic terms in a source terminology into more molecular terms in a target terminology and the decomposition of more molecular terms into their respective atoms.[55]

5.2. Maintenance Challenge

The following sections will describe the challenges maintaining multiple heterogeneous source systems in the TME over time.

5.2.1. Scalability. The every-changing nature and volume of content and the variety of sources require creating a flexible, extensible vocabulary server. The challenge is doing this without making the system so complex that it cannot be maintained or utilized. The TME must be able to grow and integrate new content without changing the data model or becoming overly complex.

5.2.2. Semantic shift and drift. Semantic shift or drift is a change in the meaning of a code—the identifier for a concept in a terminology—over time. This change in meaning is referred to as semantic drift when it happens gradually with only slight

changes accumulating over time. When there is a significant change in meaning at a single point in time, it is referred to as a semantic shift.[29]

As an example of semantic drift, before June 2010, NDC "30768015605" was described as "Vitamin D (Cholecalciferol), 1000 Unit, Tablet, Oral, Sundown Inc., 100 ea. Bottle." As of June 2010, the same NDC was changed to "**Vitamin D_3** (Cholecalciferol), 1000 Unit, **Capsule**, Oral, Sundown Inc., 100 ea. Bottle."[56] One change in the description, going from vitamin D to vitamin D_3, is not drift. Rather it is just more specificity in the display associated with the NDC, since cholecalciferol=vitamin D_3. However, the second change going from "tablet" to "capsule" is a conceptual change in the form of the drug. The change has introduced "drift" in the meaning of NDC "30768015605" (see Table 4 for other examples of sematic drift).

As an example of semantic shift, before July 2002, NDC "00074433501" was assigned to the concept "Liposyn (Fat Emulsions), 10%, IV Solution, IV, Abbott Hospital, 200ml Bag." After July 2002, the same NDC was assigned to an entirely different drug concept, "Paclitaxel (Paclitaxel, Semi-Synthetic), 6mg/ml, Vial, Injection, Abbott Hospital, 5ml Vial."[56] This practice of reusing a code to represent a new concept is a case of semantic shift.

When a standard code is used to encode data directly in the EHR and the meaning of the code changes over time, historical patient data will be interpreted incorrectly. Depending on the nature of the shift/drift, this can have a significant negative impact on patient care and should be considered a patient health risk.

The TME must have a mechanism that addresses semantic shift and drift in source code systems.

5.2.3. Code/concept deletion. Code deletion refers to removal of a code from a terminology. When data are encoded with a standard code that has since been removed by the Standard Developing Organization (SDO), the data are no longer interpretable. For example, before December 31, 2003, CPT code "0002T" was assigned to the concept "Endovascular repair of infrarenal abdominal aortic aneurysm or dissection; aorto-uni-iliac or aorto-unifemoral prosthesis." The code was deleted from subsequent versions of CPT and is no longer available.[57]

The deletion of a code is addressed with the notion of concept permanence. Concept permanence dictates that once a coded concept has been created and referenced in some way or used to store instance data, it cannot be deleted.[26] It is only acceptable to change the status of the concept through some method of deprecation.[58] Deprecation allows for a status to be applied to concepts and concept attributes to indicate that they are no longer active and should be avoided. This mechanism also requires that there be a way of indicating when concepts have been superseded.

For the TME, this means that appropriate metadata for tracking status of concepts and their associated descriptions, relationships, and attributes as well as a means of indicating when these have been superseded, must be maintained.

5.3. Utilization Challenge

The following section will describe the challenges making use of multiple heterogeneous source systems in a single environment.

5.3.1. Establishing an overarching data model. In order for multiple applications using the terminology to be semantically aligned, there must be an overarching data model that enforces consistency in the way concepts are defined and related. Establishing this overarching model is challenging, both because of the heterogeneity of source terminologies that must be integrated in the vocabulary server and the various and at times conflicting requirements of terminology consumers.

The TME must be able to unify multiple source terminologies under an overarching data model.

5.3.2. Maintaining source transparency. When source terminology is transformed and stored in different schema in a vocabulary server, it is important to confirm that information from the source was not lost or changed during the transformation. Maintaining the fidelity of the source terminology and having the ability to extract the transformed terminology and represent it just as the source originally did is referred to as "source transparency."[59]

The TME must adhere to the principle of source transparency.

5.3.3. Lack of comprehensiveness. In many cases, a standard code system cannot provide all the content that is needed to encode the data in a target domain.[29] This is due to a number of factors: changing medical knowledge and events, content that is truly local in nature such as room and bed numbers, granularity differences between the data collected and the concepts available in the terminology, immature standards, etc. Some standards provide for local extensions. Local extensions are codes added by an entity, other than the SDO, to support operational needs, e.g., locally formulated medications. However, not all standards have such a mechanism. The consequence is

that there is always data collected for which there is no standard code available. Table 5 reports LOINC coverage of US Department of Defense local lab terminology represented in the 3M Healthcare Data Dictionary.[60]

The TME must employ a strategy to deal with the lack of comprehensiveness in standards and standards that have no formal mechanism for local extensions.[61]

5.3.4. Historical compatibility. If historical data have been encoded with nonstandard terminologies/LITs, then, encoding new data using standard terminologies from a particular point onward would result in the historical data not being interoperable with the new data. This creates an ironic situation where an organization can achieve semantic interoperability with the outside world, but lacks interoperability with its own legacy data and systems.

5.3.5. Establishing a distinction in the role of the terminology model vs. the information model. An information model describes entities and events, the actions that can be associated with them, and their attributes or properties. Patient data collected according to an information model capture an "instance" in the clinical encounter. The concepts that "instantiates" a model—encoding the patient data—are provided by a medical terminology, and are defined by the terminology model. In theory, a terminology model should be used to represent concepts their attributes (defining relationships among concepts), while an information model formalizes the way concepts interact in an event or observation.[62]

An information model enables syntactic level interoperability. It defines the fields in a message and describes the relationships among clinical events and terminologies in a fashion that gives them meaning and context. While it enforces

accepted truths, it allows uncertainty and even errors. An information model cannot and should not prevent a clinician from making what seems to be an incorrect or uncertain diagnosis, but it will help prevent the storage of illogical or impossible findings such as "fractured body fluid." Information models mediate between data-gathering software and databases and are supported by terminologies.[63]

For example, a terminology model for clinical drugs can be used to define a clinical drug as having an ingredient, strength, form, route, and brand name as the defining attributes. An information model can then include the concept of a clinical drug in a medication administration observation that defines the site of administration, the dose, the prescribing clinician, etc. The key point is that a terminology model is used to help define a concept/entity unambiguously and an information model defines the behavior of the concept/entity or its interaction with other concept/entities.

Terminologies can always encode the "observables"/names in an information model and in many cases, the "values" in a name-value pair. In one name-value pair example, a LOINC code can be used to represent a white blood count. The associated value for the white blood count is numeric and is not encoded. Since this type of data (e.g., numbers, dates, times) is still computable, it would still be considered "encoded." In a slightly different situation, there can be two codes, "skin color" as the observable, and "cyanosis" as the value. In this case, both the name and the value are encoded.

Fields → Observables/Names → Values
Information Model → Terminology → Coded or Noncoded Data

Terminology and information models must be closely associated. In the best case scenario, they are developed together or the development of one informs the development of the other.

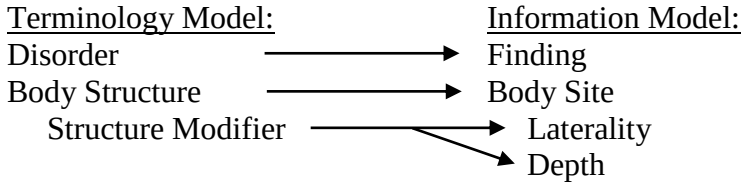
For the precoordinated term previously mentioned, “lower left eyelid laceration,” an information model may be written to capture a finding and it might look like this:

Observation = Finding
Modifier = Body Site
Observation = Body Site
Modifier = Laterality
Modifier = Depth

To properly instantiate this information model with the precoordinated term “lower left eyelid laceration,” there must be a computable way to decompose “lower left eyelid laceration” to:

Finding = “laceration”
Body Site = “eyelid”
Laterality = “left”
Depth = “lower”

Parsing these types of noun phrases is very difficult to do without being able to reference some type of terminology model. Taking the same example, a postcoordinated phrase requires that there be a mapping and harmonization of the terminology model to the information model. For example:



In either case, the work of integrating an information model with terminology that was developed independently is nontrivial.

Overlap between information and terminology models occurs when there is ambiguity with regard to what relationships (laterality, chronicity, severity, etc.) form the defining attributes of a concept. A simple example would be the concept of “left arm.” A terminology model may have included laterality as a defining attribute for an anatomic

concept type, but at the same time laterality has been included as a modifier of an information model that defines body site. This results in laterality ("left") being included in both the molecular concept ("left arm") that has been used to instantiate the body site observation of the information model (Body Site="left arm"), and as an atomic concept "on its own" being used to instantiate the laterality observation of the information model. It is even conceivable to have conflicting laterality information in the two slots.

First Variant:	Second Variant:	Conflicting Variant:
Body Site = "arm"	Body Site = "left arm"	Body Site = "left arm"
Laterality = "left"	Laterality = "left"	Laterality = "right"

Overlap between terminology model and information model creates a situation where it is possible to instantiate the same data in multiple different ways. Allowing the same conceptual data to be encoded/stored in multiple different ways replicates one of the major issues with free text (high variability), drastically reduces semantic interoperability, and results in denormalized data.

The TME must provide rigorous, computable definitions of concepts so that terminology can be properly integrated with information models.

Table 2: Release Schedules and Formats for Some Standard Code Systems.

Standard	Release Schedule	Release Format
ICD-9-CM	yearly; with quarterly updates	Variety of formats released by several independent publishing companies
LOINC	twice a year; approx. Feb and Jul	ASCII Text, Microsoft Access
RxNORM	monthly	UMLS Metathesaurus Relational (MR) or Rich Release Format (RRF) tables
SNOMED CT	twice a year; Jan and Jul	SNOMED CT specific Release Format (RF); transitioning from Release Format 1 (RF1) to a new Release Format 2 (RF2)

Table 3: Granularity of Terminology vs. Classification System. Hierarchy for "benign neoplasm of bone" in SNOMED CT[64] and ICD-9-CM.[50] Each "↳" represents a new level of increasing granularity in the hierarchy.

SNOMED CT (Clinical Terminology)	ICD-9-CM (Classification System)
(no matching granularity)	Benign neoplasm of bone and articular cartilage
benign neoplasm of bone	*(no matching granularity)*
↳ benign neoplasm of bone of lower limb	*(no matching granularity)*
↳ benign neoplasm of long bones of lower limb	↳ Long bones of lower limb
↳ benign neoplasm of femur	*(no matching granularity)*
benign neoplasm of fibula	*(no matching granularity)*
benign neoplasm of metatarsal bone	*(no matching granularity)*
benign neoplasm of tibia	*(no matching granularity)*

Table 4: Other Examples of Semantic Drift. Changes in concept descriptions. Attributes for which the descriptions changed are underlined.

NDC	ORIGINAL DESCRIPTION	NEW DESCRIPTION
24385041396	Calamine (Calamine), Lotion, Topical, Bergen Brunswig, 180ml Bottle	Calamine (Calamine/Zinc Oxide), 8%-8%, Suspension, Topical, Bergen Brunswig, 177ml Bottle
45802014464	Hydrocortisone-Pramoxine (Hc Acetate/Pramoxine HCL), 1%-1%, Cream(gm), Topical, Perrigo Co., 30g Tube	Hydrocortisone-Pramoxine (Hc Acetate/Pramoxine HCL), 1%-1%, Cream/Appl, Rectal, Perrigo Co., 30g Tube
49348003539	Antacid (Mag Hydrox/Al Hydrox/Simeth), 400-400-30, Oral Susp, Oral, Sunmark, 360ml Bottle	Antacid-Antigas (Mag Hydrox/Al Hydrox/Simeth), 400-400-40, Oral Susp, Oral, Sunmark, 360ml Bottle

Table 5: LIT LOINC Coverage. As mentioned earlier, the 3M HDD holds a federated terminology that maps among clinically equivalent concepts in standards and LITs. "Unique Lab Results" = number of unique lab result concepts in the source code system.[60]

Source of Code System	Unique Lab Results	Unique Lab Results with LOINC®	Percent Unique Lab Results with LOINC®
3M HDD Federated Terminology Total	43,664	27,509	63.00%
U.S. Department of Defense LIT	21,171	9,925	46.90%
Commercial Sites LITs	13,400	6,752	50.40%

6. PROJECT AIM: CREATE INFRASTRUCTURE FOR A TERMINOLOGY MANAGEMENT ENVIRONMENT (TME)

At high level, there are three components critical to a system that addresses the challenges described in Section 5 while supporting the structuring, encoding, and integration of clinical data:

1. People, which include subject matter and informatics experts that author and maintain the content. They must:
    - Apply expert knowledge to the creation and management of content
    - Understand terminology principles and best practice
    - Enforce standard policies and procedures
    - Understand the terminology life-cycle, how content is managed over time
2. Applications, which read and write content to and from the terminology server.
3. Infrastructure, which includes the database and services that store and provide access to the content.
    a. Services, supply consistent and controlled access to terminology
    b. Database, houses content, which includes two interdependent constituents:
        i. clinical terminology to achieve semantic consistency (data normalization)

ii. medical information models to achieve syntactic structure

This project considers the complete solution and focuses on the clinical terminology and infrastructure components of a Terminology Management Environment (TME). It is not the goal of this work to develop yet another theory with regard to how to best represent knowledge or a branching strategy for the development of clinical terminologies. The tools, databases, and methods developed in this project provide infrastructure for the maintenance of standard terminologies and mappings among them, while preserving backward compatibility with LITs and legacy systems.

7. DATABASE DESIGN OPTIONS FOR THE TME

This project considered four options for implementing standard terminologies in the TME. Selecting the optimal approach took into consideration the following:

Key Consideration:

- Support for a single, overarching terminology model; ability to transform multiple heterogeneous terminology models into a single reference model.
- Support for source transparency; ability to retain all native information and attributes of integrated code systems.
- Overall scalability and maintainability of the solution.

Additional Considerations:

- Interoperability with legacy data and systems; is it necessary to maintain compatibility with data previously stored using nonstandard codes and/or systems that produce or are dependent on nonstandard codes?
- Cost/effort to implement standards; different approaches will have very different cost and effort to implement. Are the expected immediate and/or long-term benefits of the approach worth the additional cost?
- Cost/effort to stay up-to-date and maintain mappings; if there is a decision to map between LITs and standards, what are the maintenance requirements and how can the effort be optimized?

- Flexibility and extensibility of approach; how does the approach scale and how adaptable is it to changing requirements?

The options for standards implementation are described in the following sections and evaluated against TME key considerations for support for a longitudinal patient record in an EHR. The four options were:

- Option 1: No Integration Model
- Option 2: Loose Integration Model
- Option 3: Tight Integration Model
- Option 4: Hybrid Integration Model

Each of these options will be discussed in detail in the following sections.

7.1. Option 1: No Integration Model

One implementation approach is to use standard terminologies "directly"—meaning data are initially encoded using the standard code. In this case, all the LITs or master files (e.g., list of tests, results, specimens, units) referenced in information systems are replaced with standard codes. Systems then use the standard codes directly without performing any type of translation from LIT to standard.

- Support for a single, overarching terminology model; this option does not support a single reference terminology model unless a single standard code system is used.
- Support for source transparency; this option can support source transparency as long as a proper versioning strategy is implemented.

- Overall scalability and maintainability of the solution; specific to terminology, this solution may require minimal effort to maintain. However, the approach may create additional downstream effort for the consumers of terminology.
- Interoperability with legacy data and systems; because this approach does not maintain any type of link between the original terminology used to encode data and the standard, there is no backward compatibility.
- Cost/effort to implement standards; initial effort to implement may require changes to the applications that reference the terminology. For example, if a legacy system had rules that referenced particular concepts or classes in the LIT, they will have to be changed to reference appropriate concepts/classes in the standard.
- Cost/effort to stay up-to-date and maintain mappings; the effort to stay current with changes in the standards is very use-case-dependent. It may be that the applications referencing the standard can just follow the standard. In other words, as updates are available in the standard, they are applied without attempting to understand or mitigate the impact of any changes. However, as in the example above, if the interaction between the application and the standard is more tightly coupled, rules that reference particular concepts/classes in the standard must be modified if the concept/class undergoes a change (i.e., deleted, deprecated, or undergoes a change in meaning).
- Flexibility and extensibility of approach; this approach is not very flexible, but it is extensible. The design of the standard terminology is dictated by the SDO, limiting flexibility. Changes to the terminology are centrally managed by the

SDO requiring time to implement. However, the approach is scalable. Complexity of dealing with the interaction between terminology and the systems is pushed off to the systems, rather than dealing with it at the level of the terminology.

This is not an appropriate organization-level approach to address standards implementation, but for individual systems that are performing data collection and for which the content available in the standards is adequate, this can be an appropriate and effective way to implement standards.[29] Since the TME is meant to address the needs of an EHR, and integrate multiple disparate systems that have different levels of interaction with terminology, this approach will not work (see Figure 1).

7.2. Option 2: Loose Integration Model

Another implementation approach involves performing a direct mapping between the LIT and each target standard terminology (point-to-point mapping). Since multiple standard terminologies are required to support an EHR, this approach results in a web of many overlapping source LIT to target standard mappings as well as source LIT to target LIT maps (see Figure 3).

- Support for a single, overarching terminology model; this option does not support a single reference terminology model unless a single standard code system is used.
- Support for source transparency; this option can support source transparency as long as a proper versioning strategy is implemented.

- Overall scalability and maintainability of the solution; see the discussion regarding point-to-point mapping. This solution has poor scalability and becomes increasingly difficult to maintain as additional code systems are managed.
- Interoperability with legacy data and systems; since a reference mapping between the LIT and the standard is maintained, new data are interoperable with legacy data, except in the case where there is no equivalent concept in the LIT or standard.
- Cost/effort to implement standards; for three or less mappings between source/target pairs, this type of mapping can be effective. However, as the number of overlapping source/target pairs increase, the number of mappings that must be generated grows at a rate of n(n-1)/2 (see Figure 2).
- Cost/effort to stay up-to-date and maintain mappings; just as the initial effort rapidly grows as the number of source/target pairs grow, so does the number of mappings that must be maintained.
- Flexibility and extensibility of approach; this approach is neither flexible nor extensible. A map requires that there be a corresponding concept in the target terminology. If there is not one, there is no map, and consequently no way to represent a LIT concept for which there is no standard code.

Because of the inability to represent LIT concepts for which there is no equivalent standard code and the inefficiency of both initial and maintenance mapping, this approach is inappropriate for the TME (see Figure 3).

7.3. Option 3: Tight Integration Model

The third option involves creating a reference terminology (mediation layer) to which standards and LITs can be mapped (centralized mapping). In this approach, the legacy systems that collect data continue to use their native code system (LIT). That LIT is linked to a federated terminology through centralized concept mapping (see Figure 4). Standards are similarly mapped to the federated terminology (see Figure 5)

Concept mapping is the process of building links among codes/terms in disparate terminologies in order to integrate them for the purpose of data exchange. It can also serve to normalize a term-based terminology to a concept-based terminology. A concept is a unique, definable, abstract idea that describes a class of entities (i.e., "man"), a category of objects (i.e., "mammal"), and/or the relationships between them (i.e., "is a"). A concept has a specific, known meaning and is labeled using terms. Multiple different terms can be used to label the same concept.

Mapping must also take into account the purpose for which the mappings will be used. Because of this, a mapping done in one context could be considered equivalent where in another context it would be broader or narrower. For example, drugs being mapped for inventory may be matched for equivalence down to the level of manufacturer and packaging, whereas drugs mapped as allergens could be matched for equivalence based on ingredients alone. A third intermediate level of specificity for mapping of drugs would be clinical drugs where the level of granularity could go to ingredient, strength, form and route.

This approach has the following advantages:

- Interoperability with legacy data and systems; since a reference mapping between the LIT and the standard is maintained, new data are interoperable with legacy data.
- Cost/effort to implement standards; mapping effort is centralized, providing significant economy of scale over point-to-point mapping.
- Cost/effort to stay up-to-date and maintain mappings; each mapping between the source and the federated terminology can be maintained separately without impacting other mappings (see Figure 4).
- Flexibility and extensibility of approach; this approach is both flexible and extensible and provides for:
  - full control over the federated terminology that is referenced by internal systems
  - the ability to compensate for semantic drift or shift in standard terminologies
  - the ability to encode local data that do not appear/belong in the standard terminologies

Although this has significant advantages over using the standards directly and point-to-point mappings, it requires transforming the source terminologies to a single reference terminology model. It is difficult to do this and maintain source transparency, the second key consideration for the TME. Consequently, this option was not selected for the TME (see Figure 5).

7.4. Option 4: Hybrid Integration Model

The fourth option combines a loose and tight integration models. In accordance with the loose integration option, each source is maintained in its own namespace. A namespace is an abstract or logical container within the terminology server that segregates content from various sources. However, integration is accomplished through a centralized mapping in a TME Federated Terminology (TFT), rather than point-to-point mappings among the various namespaces (see Figure 4).

This design option was selected for the TME because it maintains many of the advantages of centralized mapping and meets all three key considerations. It provides a single overarching terminology model. It allows for centralized mapping, making the approach more scalable and reducing the number of mappings that must be maintained. It allows for source transparency (see Figure 6).

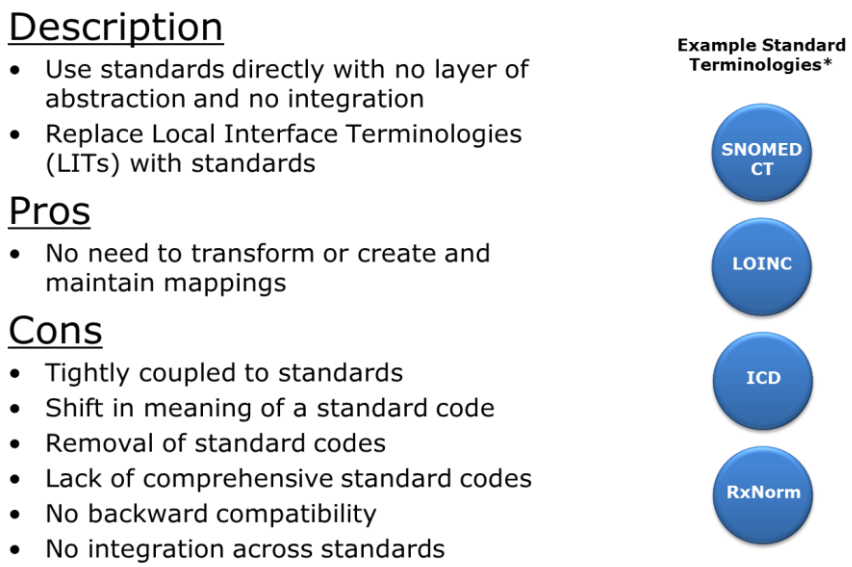


Figure 1: Option 1 - No Integration. This option involves using the standards directly. It fails to meet the first key consideration for TME which is support for a single overarching terminology model. *Systemized Nomenclature of Medicine - Clinical Terms (SNOMED-CT); Logical Objects Identifiers Names and Codes (LOINC); International Classification of Disease (ICD); RxNorm.

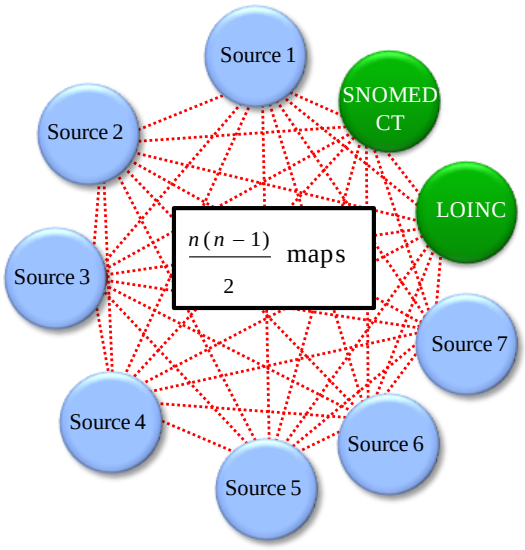


Figure 2: Point-to-Point Mapping. Dotted red lines represent mappings among LITs and standard code systems. In this example, the mapping of nine source code systems results in the creation of thirty-six point-to-point mappings.

Description:

- Content within namespaces is linked through point-to-point mappings
- A namespace is a logical container that segregates source content from other sources in the same database

Pros:

- All source native attributes can be retained
- Allow each source to own changes within the namespace

Cons:

- No single consistent "ontology"
- Point-to-point mappings grow at a (n(n-1)/2) rate; n=number of sources (e.g., 6)

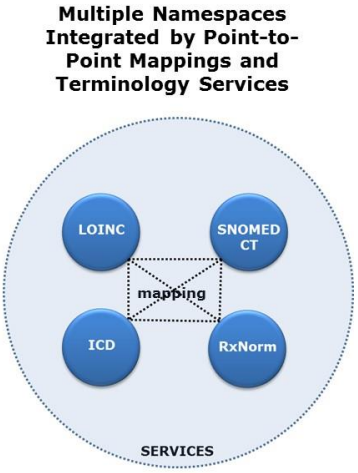


Figure 3: Option 2 - Loose Integration Model. This option involves content integration through point-to-point mapping. It fails to meet the first and third key considerations for TME. There is no overarching model and content is not scalable.

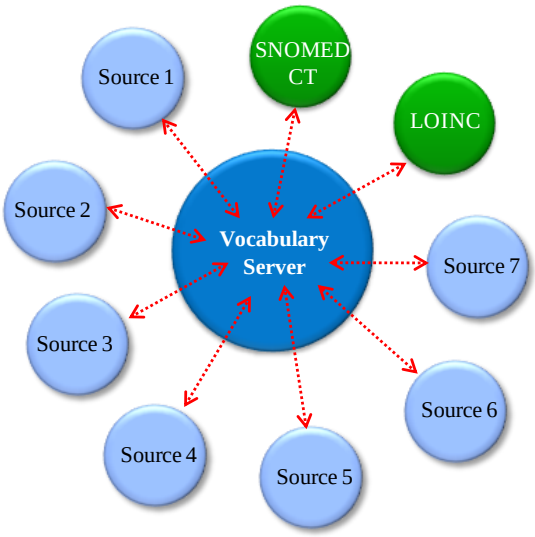


Figure 4: Centralized Mapping. Mapping to a central reference terminology. Dotted red lines represent mappings among source LITs and standards.

Description:

- External source terminologies are "mapped" to a single federated terminology
- Sources do not persist in their own namespaces

Pros:

- Single consistent "ontology"
- (n) mappings; n=number of sources (e.g., 4)

Cons:

- All source native attributes are not retained
- Changes in each source have to be continually identified and reconciled through the mapping process

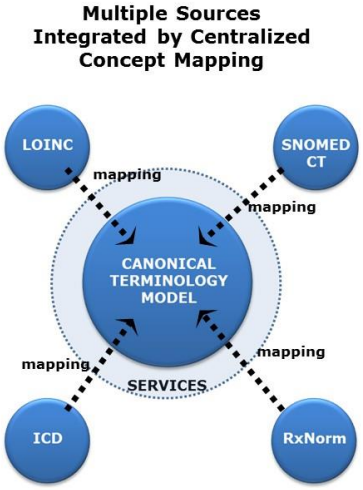


Figure 5: Option 3 - Tight Integration. This option involves content integration through centralized mapping. It fails to meet the second key considerations for TME. Since sources are transformed in the mapping, there is a lack of full source transparency.

Description:

- External source terminologies are mapped to a single terminology
- Sources persist in their own namespaces

Pros:

- Single consistent "ontology"
- All source native attributes can be retained
- (n) mappings; n=number of sources
- Allow each source to own changes within the namespace

Cons:

- Changes in each source have to be continually identified and reconciled through mapping

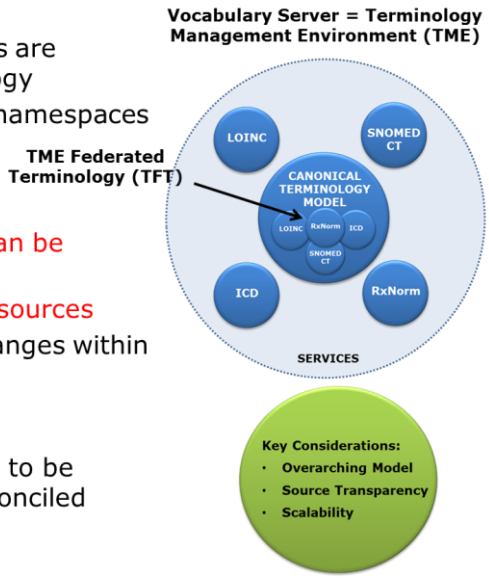


Figure 6: Option 4 - Hybrid Integration. This option involves content integration through centralized mapping while maintaining sources in their own namespaces. It meets all key considerations for TME.

8. COLLECTING AND ANALYSING REQUIREMENTS FOR THE TME

Having established the challenges that must be addressed by the TME and the general approach for managing content in a federated terminology in Section 7, this section defines the TME actors and functional requirements.

8.1. Terminology Actors

A terminology actor is a role that aggregates a set of interactions between a user/system and the federated terminology. In the TME, there are two broad types of interaction: using/referencing the terminology and maintaining/augmenting the terminology. A single user can assume multiple roles. Users can be organizations, applications, services, or individuals. The types of actors were derived empirically, through experience and observation of current vocabulary servers and EHR system and by referencing similar work done by HL7.[65] Terminology actors include: terminology browser, terminology author, clinical investigator, and applications. These roles are not necessarily distinct; rather in most cases, they are overlapping. In the subsequent sections, actors will be defined along with supporting use cases that describe the manner in which they interact with the terminology in the TME.[65,66]

8.1.1. Terminology browser. A *terminology browser* is an actor that must be able to view a federated terminology either to maintain it or to use it (see Figure 7). This requires capability to search for a specific concept using various terms and codes or browse using navigational hierarchies or traversing the semantic network.

Good browsing capability is critical to the TME. Because of the volume and complexity of the content integrated in the federated terminology, it can be challenging to make it accessible for use and maintenance. Using the 3M HDD as an example, as of July 2010 there were 2,206,205 active concepts, 16,711,326 relationships, and 31,513,166 descriptions associated with those concepts.[29] As of January 2008, SNOMED CT had approximately 311,000 active concepts,[64] as of July 2010, RxNORM had approximately 140,000 active concepts,[67] and as of June 2010, LOINC had approximately 56,000 active concepts.[68] Finding a specific concept or set of concepts in such large volumes of content requires advanced searching and filtering capabilities. A *terminology browser* must be able to navigate a semantic network and employ various searching string matching techniques against the federated terminology. This technique must include methods that allow for string matching for a particular concepts or a concept that is used to aggregate other concepts and the ability to place filters and or provided related metadata in the searching. The metadata can be information such as source code system version or other historical information with regard to the content (provenance data) or the intended context of use.

8.1.2. Terminology author. A *terminology author* must be able to update, version, and extend the federated terminology. *Terminology author* requires all of the "reading" capabilities of *terminology browser*, but must also be able to "write" to the

federated terminology (see Figure 7). The key challenge with *terminology author* is properly managing privileges and the various levels of access control within this role. The content must provide namespaces and other means of organizing, partitioning, and assigning status that can be referenced by privileges to determine access. Access control for *terminology author* must have the following levels:

1) Global Author: This *terminology author* has editing privileges to all content regardless of namespace.
2) Namespace Author: This author has editing privileges to content within a particular namespace only. Namespace Author can add code attribute content to concepts that are in other Global Namespaces within their local namespace (e.g., add relationships to a LOINC concept that exist only in the author's namespace).

There are specific user-level privileges that can be enabled or disabled for Global and Namespace Authors which include:

- Ability to create new concept
- Ability to create new relationships
- Ability to create new representations
- Ability to create new mappings.
- Ability to edit content created by another user.
- Ability to edit content that is not in active status.

Intentionally missing from the list of capabilities of *terminology author* is any ability to delete concepts. Authors can only change the status of a concept or its associated attributes. This approach to managing content is meant to be compliant with the notion

of concept permanence.[26] A federated terminology that is referenced to store patient data in a longitudinal record must have robust auditing capability, specifically the ability to know how the terminology looked at any single point in time. By assigning status to concepts and their associated attributes (relationships, properties, descriptions, etc.), authors can perform "housekeeping" types of updates to the terminology such as inactivating or making obsolete certain relationships or terms, but still maintain the integrity and history of the federated terminology. This is important to support both internal and external dependencies on the content. Examples of dependencies include the ability to decode data that has been stored over time or track changes to a concept that maybe referenced by a rule.

8.1.3. Clinical investigator. A *clinical investigator* is interested in using the instance data encoded by the federated terminology to answer business intelligence queries, perform benchmarking, and for quality assurance and research analytics (see Figure 7). However, understanding the instance data and answering questions with it requires knowledge of the data structure in the federated terminology and the ability to augment/manipulate the data structure to answer new types of questions. A *clinical investigator* must be able to create new classes and relationship types that can be used in queries for indexing and inferenceing, to aggregate data for a specific purpose or link concepts using horizontal or vertical relationships. In order to perform these tasks, a *clinical investigator* must have the concept browsing functions of a *terminology browser* and limited *terminology author* capability.

8.1.4. Applications. During development and deployment, clinical systems must utilize the federated terminology. These *applications* require "read" and "write"

capability and must support both automated and manual referencing and manipulation of the federated terminology (see Figure 7). The Lexical Query Services Specification (LQS) describes use cases that would require the following types of tasks be performed.[66]

Applications reference the federated terminology to perform the following types of tasks:

1. Collect structured/encoded data
    - During data entry, an *application* references a class/pick-list in the federated terminology to help a user populate a field with a concept. The proper place to maintain these classes/pick-lists is in the federated terminology. Doing so allows the lists to be dynamic and up-to-date, without requiring modification of an enumerated list "hard coded" in the clinical systems. *Applications* may also utilize the federated terminology to validate values "hand-entered" by system users.
    - If the data have already been entered in free-text, an *application* like a Natural Language Processing (NLP) engine can be used to apply context and parse out concepts that can be encoded by referencing the federated terminology.[69]
2. Perform mediation services
    - *Applications* such as integration engines translate inbound codes in one code system to an outbound code in another specified code system by referencing the federated terminology.
    - *Applications* may also modify the structure of instance data for export to other systems.
3. Display data

- *Applications* need to be able to pick a particular concept description or set of descriptions from the federated terminology to display to system users both during data entry and when viewing previously collected data.

4. Inferencing
    - *Applications* need to be able traverse relationships among concepts, for example, if the application needs to check if a medication is an antibiotic.

8.2. Functional Requirements for the TME

The Lexical Query Services Specification (LQS) describes a set of use scenarios that serve as a good reference for the minimum set of requirements the TME must support. The following use scenarios are defined in LQS:

1. "Information Acquisition - Using terminology services to aid in the process of entering coded data.
2. Information Display - Using terminology services to translate coded data elements into human- or machine-readable external forms.
3. Mediation - Using terminology services to transform messages or data records from one form or representation into another.
4. Indexing and Inference - Using terminology services to inquire about associations which may or may not pertain between various data elements and to assist in the location of various data record sets, which may contain information relevant to the specific topic or entity.
5. Browsing - Using the terminology services to determine the structure and meaning of a terminology system.

6. Composite Concept Manipulation - Using the terminology services to aid in the entry, validation, translation, and simplification of composite concepts."[66]

The use scenarios are primarily addressed through TME support of CTS Section 11.2; however, they will be referenced in the use-case-based service requests described in Section 11.1. In addition to these LQS use scenarios, there are additional requirements described in following sections that are specific to the integration and maintenance of multiple source terminologies in the TME.

8.2.1. Faithful concept representation. The TME is required to integrate multiple standard code systems and LITs into a single federated terminology. This must be accomplished without losing or altering the meaning of source content attributes and properties. The TME follows a concept-based approach for knowledge representation and is designed to allow creation of additional properties and defining attributes for concepts.[26] Differences in granularity among integrated terminologies can be reconciled by traversing hierarchical relationships. In this way, the TME maintains the original semantics and granularity of the data that are encoded or transformed with concepts in its federated terminology.

Functional Requirement: Source content must be represented in the TME with high fidelity (no change in the original meaning).

8.2.2. Single schema/common terminology model. The content of the TFT is utilized for various purposes by multiple different terminology actors previously described. The point of having a single schema is to integrate the content and reconcile the heterogeneity of the source code systems from the terminology actors.

As an alternative, specification like HL7 Common Terminology Services (CTS) can hide the heterogeneity of multiple disparate data models by using a services layer to normalize the terminology models, but the content from multiple sources is not integrated. Some type of federated terminology is still required for content integration. HL7 CTS is described in further detail in Section 8.2.9.

Functional Requirement: The TME must store, access, link, and augment multiple disparate code systems in a single schema/terminology model.

8.2.3. Support for multiple data/information models. The semantics of concepts in the TME must be explicit and computable in order to be shared among various terminology actors, both people and machines. Concept definitions, relationships, and representations must also be flexible and extensible enough to incorporate new content without losing clinically relevant information. Information models will reference the terminology for coded elements and value sets. The necessary metadata (TME) and content organization (TFT) must be in place to support models.

Functional Requirement: Concept representation in the TME must be expressive enough to transform multiple source terminology models and support medical information models of various systems.

8.2.4. Partitioning. In order to export terminology for a special purpose, generate multiple consistent views of the same underlying data, control access to particular type of content, and create value sets, the TME must have the ability to partition concepts and concept attributes into sets. Any one concept or concept attribute must be able to participate in multiple sets. The granularity of partitioning must be finer than just generating concept sets. Sets of all concept attributes must also be supported.

Examples include creating values sets to support an information model or exporting a subset of content for a domain-specific application.

Functional Requirement: The TME must have the ability to create "content containers" or sets of concepts and/or concept representations.

8.2.5. Version control. The TME must be capable of tracking discrete version information for all TME entities. Versions are associated through metadata linked to each entity and are assigned based on changes in status. In order to make this possible, concepts, relationships, mappings properties, and attributes are all assigned a status that can be associated with additional metadata specific to version. Here are the types of content that are versioned in the TME:

- Concept versioning, inactivation, and expiration management
- Concept representations
- Concept attribute/property versioning
- Concept relationships
- Mappings
- Native source code system version information
- TFT version history

All related data elements of concepts in standard terminologies can change from one version to the next: relationships, surface forms, definitions, codes, and attributes. In addition, it is not always clear when a change in a relationship or surface form (which would result in versioning of the attribute) constitutes a change in the meaning of a concept. These dependencies between a concept and its associated attributes, representations, and relationships must be considered in the versioning process.

Unfortunately, the dependencies are not consistent across namespaces and in many cases will require SME review. Tracking this information and being able to represent how a source looked at any previous point in time is important for semantic interoperability and QC of the federated terminology.

The management process for addressing versioning challenges requires:

- good tracking and auditing capabilities in the TME
- ability to store version information with instance data in the data repository or reliably reference the date and time of data storage to determine the version information using the TME
- ability to assign version information to content that is not versioned by the source
- ability to communicate version information in messages to external systems

Functional Requirement: The TME must be able to provide version information and recreate multiple previous versions of the same content.

8.2.6. Terminology browser use-case-based TME service requests. Essential functional requirements for the TME were established by identifying the capability necessary to support the use cases for each type of terminology actor. A *terminology browser* needs to view and navigate terminology for two general purposes: entity browsing and administrative browsing. Entity browsing is focused on the meaning and design of the terminological content in the federated terminology and the various namespaces. Administrative browsing is focused on associated information about the code systems integrated in federated terminology and the status of work performed by various types of *terminology authors*.

The following is a set of specific example use cases for content browsing followed by bulleted functional requirements:

1. A *terminology browser* needs to find a SNOMED CT code for the finding (term or phrase) "dyspnea."
    - Find a representation of specified type (e.g., concept/code) in an optionally specified code system/domain/namespace using one/many synonymous term(s)/phrase(s) (e.g., search using the term "dyspnea" or the phrase "shortness of breath" as display representations, in the domain of findings).
    - Return a specified representation type for a concept (e.g., return the SNOMED CT code for the concept of "dyspnea").
2. A *terminology browser* needs to find a particular description for the SNOMED CT code "267036007."
    - Find the concept for a code from a particular code system (e.g., return the concept identifier for SNOMED CT code "267036007").
    - Return specified description/code for a concept (e.g., return the "SNOMED CT Fully Specified Name" for the concept of "dyspnea").
3. A *terminology browser* needs to find the "parent" concept of the finding "dyspnea."
    - Find the concept for the term "dyspnea" in the domain of findings; the domain (context of use) is specified to disambiguate the concept of dyspnea as a finding from a diagnosis or a keyword.
    - Navigate the links/relationships among concepts/classes in and among code systems (e.g., "dyspnea" is a "child" of "respiratory finding").

- Return the concept identifier for the right- or left-hand concept in a relationship triplet (relationship triplet = left-hand concept, relationship type, right-hand concept).

4. A *terminology browser* needs to view everything that the federated terminology has documented regarding the finding "dyspnea." This includes concept metadata (e.g., status) as well as related concepts, mappings, and designations.
   - Find the concept for the term "dyspnea" in the domain of findings.
   - Return the links/relationships to and from the concept "dyspnea."
   - Return all of the associated codes and descriptions for the concept "dyspnea."
   - View all information related to a particular concept and how it is currently described in various code systems; in other words, how a concept is represented in the federated terminology, as well as the ability to determine how it was represented in various other terminologies and the native code system it was derived from (e.g., relationships, mappings, description, synonyms, attributes, etc.).
5. A *terminology browser* needs to view the history of how the concept "dyspnea" has been represented in the federated terminology.
   - Return the history of a concept/code/description (e.g., what was the status of code "x" in version "y" of a standard terminology).

The following is set of example use cases for administrative browsing which include capabilities necessary to view information about the code systems integrated in federated terminology and the status of work performed by *terminology authors*:

6. A *terminology browser* needs to determine what code systems are included in the federated terminology.
    - Return a list of code systems that have been mapped to the federated terminology (e.g., LOINC, SNOMED CT, RxNORM, LITs, etc.).
7. A *terminology browser* needs to know what the most current version of LOINC in the federated terminology is.
    - Return version information for the code systems integrated in the federated terminology.
8. A *terminology browser* needs to know row-level (LIT code to federated terminology code) status of a mapping effort from an LIT to the federated terminology.
    - Return metrics for the following types of workflow status reports:
        - status of mapping effort (rows reviewed, rows mapped, rows with outstanding questions)
        - status of mapping questions
        - results of validation during interrater review

8.2.7. Terminology author use-case-based TME service requests. SMEs are a primary user of terminology author functions. The TME must address processes related to the management of the federated terminology by SMEs. This includes workflow functionality for maintenance and extension of content as well as QC processes.

At an operational level, concept mapping functions require:

- Editing environment to link external code systems to the TFT and to integrate LITs and standard code systems with associated codes, terms/descriptions, relationships, and attributes.

- Work queues for tool users
- Administrative reporting functionality around work queue status: pending, in process, completed, etc.
- What about algorithms to find potential matches?

The previously enumerated use cases for a *terminology author* are addressed with the following functions:

1. A *terminology author* needs to be able to browse the federated terminology to perform maintenance and add content.
    - All of the capabilities of a *terminology browser.*
2. A *terminology author* needs to be able to add a new concept "SARS" and associated metadata to the federated terminology in the TFT namespace.
    - Browse to verify that the concept "SARS" does not already exist in the federated terminology.
    - Build a unique concept with appropriate relationships, attributes, properties and descriptions.
    - Create mapping link from the concept in a source code system to the new concept in federated terminology.
3. A *terminology author* needs to be able to add an "is a" relationship between the concept "cytomegalovirus" and the class "virus."
    - Add attributes to an existing concept. Concept attributes include: properties, attributes, relationships, and descriptions.
4. A *terminology author* needs to be able to create a new relationship type "has ingredient."

- Create new types of description, relationships, attributes, and properties for concepts.

5. A *terminology author* needs to be able to update the status of the concept "chickenpox virus" to "inactive" and indicate that it is superseded by the concept "varicella zoster virus."
    - Change the status of concepts and associated metadata.
    - Indicate that an inactive concept should be replaced with another concept.
6. A *terminology author* needs to be able to close a version of the federated terminology.
    - Create both workspace (a version that contains only the work of a single *terminology author*) and global versions of the federated terminology.
7. A *terminology author* needs to be able to add a new namespace to the federated terminology.
    - Obtain code system from the source organization (extract step).
    - Transform source content using a data mapping from the source code system's data model to the federated terminology data model (transform step).
    - Load the source code system into the TME (loading step).
    - Integrate the source code system by mapping it to the federated terminology (mapping step).
8. A *terminology author* needs to be able to update an external code system maintained in the federated terminology.
    - Obtain code system update from the source organization (extract step).

- Transform source content using a data mapping from the source code system's data model to the federated terminology data model (transform step).
- Determine what was changed in the update (diffing step).
- Load changes in the source code system (loading step).
- Reevaluate mappings of all new and modified source content to the federated terminology (mapping step).

8.2.8. Clinical investigator use-case-based TME service requests. The following is a set of example use cases followed by functional requirements to support a *clinical investigator*:

1. A *clinical investigator* needs to identify how often a drug is prescribed to treat a bacterial infection, when susceptibility test results indicate that the isolated organism is resistant to the prescribed drug's active ingredients.
   - Find the concept for a susceptibility test that has a result value of "resistant."
   - Follow a "has analyte" relationship from the susceptibility test to the drug ingredient.
   - Determine if a prescribed drug contains ingredients to which the isolated organism is resistant by following a "has ingredient" relationship from the prescribed drug to ingredients.
2. A *clinical investigator* wants to assess the effectiveness of a particular set of lab test methods vs. other common methodologies.
   - Enumerate the list of methodologies that are of interest.
   - Build a new class that aggregates the methodologies.

8.2.9. Standard Application Programming Interface (API). In order for the TME to be able to support a standard terminology service, it must provide efficient access to computable data representations that are sufficiently granular and comprehensive to provide all the information required by the service. HL7 Common Terminology Services (CTS) version 1.2 was selected as the service to test the TME's ability to support a standard terminology service. Version 2 of the CTS specification was used as a gold standard to validate the TFT schema. This will be discussed in greater detail in Section 11.2.

The HL7 Common Terminology Services (CTS) specification is an Application Programming Interface (API) that specifies a set of common functions that a source code system must provide in order to generate and interpret HL7 version 3 messages.[65] A custom CTS service must be written for each source. The service talks to the standard CTS API and applications access the code systems using standard CTS calls against the CTS API (see Figure 8 and Figure 9). Basically, the complexity of normalizing disparate source data models is shifted from applications to the services. Applications evoke the services using a specified set of input criteria. However, the CTS API does not integrate the content from multiple source code systems. Each code system is housed in its native data model, but the metadata that links/maps them and resolves conflict among the sources must still be stored in ancillary tables.

The CTS version 1.2 specification defines messaging and vocabulary layers between applications that process HL7 messages and the referenced code systems. The messaging layer is a set of functions specific for support of HL7 messages. The vocabulary layer is a more generic set of methods that should be supported by any vocabulary server. The

list of CTS functions that must be supported by the TME along with descriptions can be found in Appendix C. The following are some examples:

1. Validate Code (validateCode): this function is used to determine whether the supplied coded attribute from a code set is valid in the specified vocabulary domain and application context.
2. Validate Translation (validateTranslation): this function is used to determine whether the translation portion of the coded attribute is valid in the specified vocabulary domain and application context.
3. Translate Code (translateCode): this function is used to translate a supplied coded attribute into a target form that is valid in the target application context.
4. Fill in Details (fillInDetails): this function is used to supply additional details for a coded attribute, including all code system names, versions, and display names.
5. Implies (Implies): this function is used to determine whether the parent-coded attribute implies (subsumes) the child.
6. Equivalent (Equivalent): this function is used to determine whether two specified coded attributes are equal.

Functional Requirement: The TME must support standard functionality as described by HL7 Common Terminology Services version 1.2, at a minimum. CTS version 2 will be used for external validation of TME capabilities.

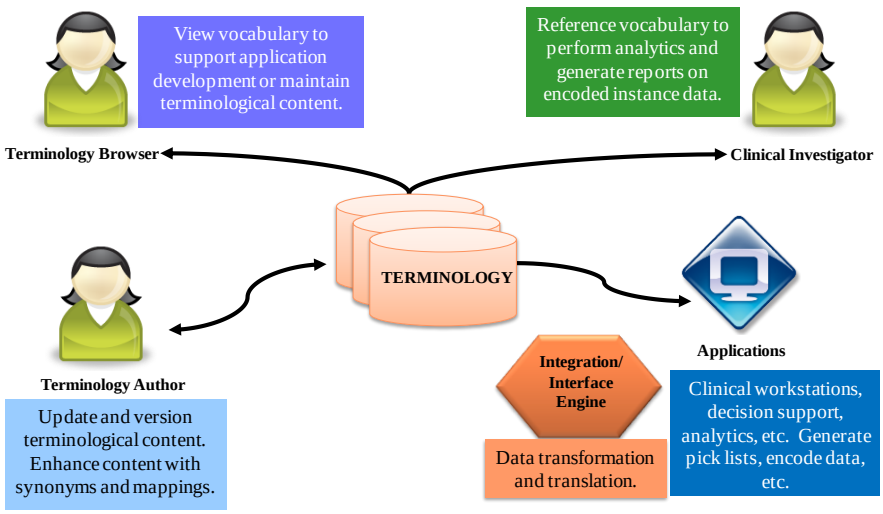


Figure 7: Terminology Actors. Solid black arrows indicate interactions with terminology. Roles are not discrete. For example, a Terminology Author must also interact with the federated terminology as a Terminology Browser.

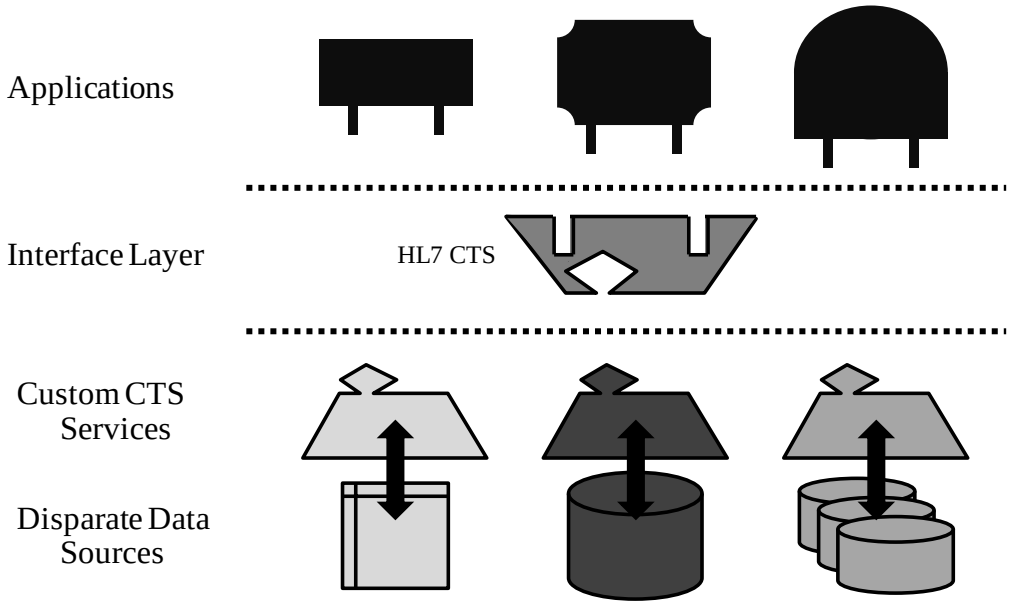


Figure 8: HL7 CTS Application Programming Interface (API). Diagram from Object Management Group (OMG).[65]

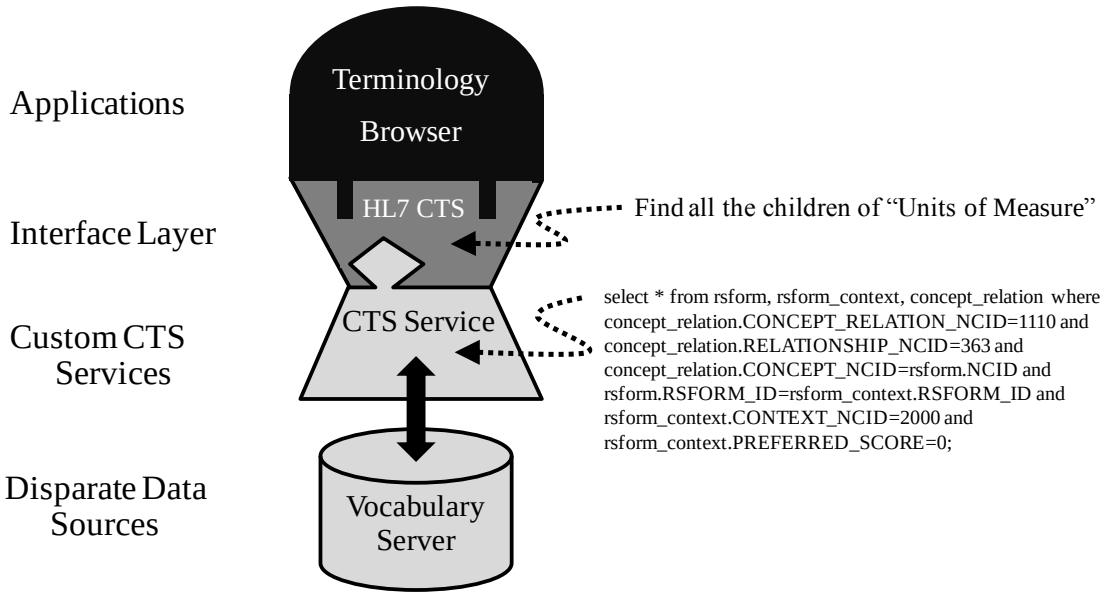


Figure 9: Example CTS Implementation.[65]

9. THE TME LOGICAL AND PHYSICAL DATA MODELS

In Section 9.1, the schema for the TME vocabulary server will first be described as a logical model. In the subsequent section, an Oracle implementation-specific, relational database physical data model is presented.

9.1. TME Database Logical Model

The purpose of the TME logical model is to describe the association among TME elements in a manner that is not implementation specific. The TME vocabulary server design has the following primary elements:

- ENTITY: thing that exists; concepts (abstract ideas) are referred to as entities in the TME[70]
- Entity Representations: Terms, designations, displays, codes used to label entities
- Entity Relationships: Named, directional associations among entities
- Entity Attributes: Defining characteristics of entities
- Entity Mappings: Links among entities for the purpose of transformation
- Properties: All other nondefining characteristics of entities, attributes, representations, and relationships

Entity is the core element of the TME logical model. All other elements further define or modify entity. An entity may have zero to many relationships, mappings,

attributes, and properties. An entity must have at least one representation. All elements in the logical model may have zero to many properties. The association of TME elements is represented visually in Figure 10.

9.2. TME Database Physical Model

1) A staging environment that is used to transform source code systems from their native format during the ETL process.
2) Core tables that house the source terminological content and the TFT along with versioning information.
3) Mapping tables that hold information specific to how content from various sources is cross-linked and provide contextual information about the links such as who performed the mapping, the purpose of the mapping, and the specificity of the mapping.

Every table in the TME schema that holds terminological content has a base and revision instance. The "current" table holds the currently-effective revision of the content. The "revision" table holds previous and future-effective versions of the content. The revision table is populated when content is inserted, updated, or deleted in the base table. This allows for quick performance when referencing content that is current, but still provides a reliable way to look at past and future-effective versions of the content.

A simplified schema of the core tables that does not include the revision tables is depicted in Figure 11. This simple schema is overlaid with the logical model in Figure 12.

9.2.1. ENTITY table.

The ENTITY table holds the basic structural unit of source code systems. ENTITY assigns unique TME Component Identifiers (TCIDs) for

each code/set-of-codes in a source code system and for each concept in the TFT. ENTITY is related to every other core table, except for RSFORM_LINK, through TCID as a foreign key or any field with "_TCID" as the suffix. For sources that are concept-based TCIDs are assigned per concept. For sources that are not concept-based, TCIDs are assigned for each code or set of codes in the source. While the TFT is concept-based, the TME vocabulary server is code-based. TCIDs are assigned for each row of associated data in source code systems. It is only through mapping links to the concept-based federated terminology in the TME that a virtual concept is created and associated with source code system codes.

The ENTITY table includes the following fields (Table 6):

- ENTITY.TCID: TME Component Identifier; TCID is the ENTITY table primary key—a numeric identifier for unique content in the ENTITY table. However, TCID is not unique in the ENTITY revision table (ENTITY_REV), since multiple previous and future-effective versions of the same TCID are tracked in ENTITY_REV. All other fields in the TME that have a suffix of "TCID" are foreign keys that reference a TFT concept in the ENTITY table.
- ENTITY.CID: Component Identifier; CID is an alpha-numeric identifier for unique content in the ENTITY table. The CID cannot contain nonprintable characters or spaces. The CID for all concepts in each source-terminology namespace (identified by entity.SOURCE_TCID) is prefixed by the name of the source code system—e.g., TFTActiveStatus, LOINCGlucose, RxNormGlucose, etc. The CID is critical to the TME in the diffing step of the ETL process. A consistently-produced CID is used to determine if there have been changes to

source content for sources that do not assign identifiers for all the TME-required data elements (e.g., lacking a controlled identifier for description/representations) or have appropriate change control.

- ENTITY.UP_CID: Uppercase Component Identifier; UP_CID is an all uppercase CID. It is used to improve performance by allowing for case-insensitive comparisons of CIDs during diffing. There are instances where term case implies some of the semantics of the underlying concept (e.g., units of measure; m = meters, M = Moles).[71] In these instances, UP_CID is not used for the comparison.
- ENTITY.SCHEMA_TCID: References a TCID from the ENTITY table for the TFT concept that denotes the schema of this entity. Schema concepts are stored in the TFT namespace. The schema is not set by the source code system, and is currently always set to TCID 2 = TFTSchema.
- ENTITY.GENDER_TCID: References a TCID from the ENTITY table for the TFT concept that denotes the gender for gender-specific entities. Gender concepts are stored in the TFT namespace.
- ENTITY.ONTOLOGY_XML: XML string that holds a formal definition of the entity.
- ENTITY.DEFINITION_XML: XML string that holds a human-readable definition of entity.
- ENTITY.SUPERSEDED_BY_TCID: References a TCID from the ENTITY table for the entity that replaces this entity. Since content can never be deleted

from the TME, SUPERSEDED_BY_TCID is a way of referencing the entity that should replace this entity when/if it is deprecated (status changed to "inactive").

The following fields appear in multiple TME tables. They are described here as being associated with a component, where component is an entity, surface form, attribute, property, or link (e.g., in the ENTITY table *COMPONENT* = ENTITY). These field descriptions will not be repeated for subsequent core tables:

- *COMPONENT*.STATUS_TCID: References a TCID from the ENTITY table for the TFT concept that denotes the status of this component. Status concepts are stored in the TFT namespace (e.g., TFTActiveStatus, TFTInactiveStatus, TFTProposedStatus, TFTObsoleteStatus). Status is used in combination with version information in the revision tables to create an audit trail for TME content.
- *COMPONENT*.USAGE_SCORE: Holds a score from a ranking system that is used to roughly evaluate how commonly this component is "used," whether it is instantiated in the patient record or appears in multiple source code systems.
- *COMPONENT*.SOURCE_TCID: References a TCID from the ENTITY table for the TFT concept that denotes the source code system of this component. TFT concepts for source code systems are stored in the TFT namespace (e.g., TFTHDD, TFTLOINC, TFTRxNorm, etc.).
- *COMPONENT*.SOURCE_VER_TCID: References a TCID from the ENTITY table for the TFT concept for the source code system version/release/update where this revision of this component first appeared. If this revision exists in subsequent versions/releases/updates, *COMPONENT*.SOURCE_VER_ID is not updated.

- *COMPONENT*.ADDED_DATE: The date/time (GMT) when this revision of this component was added to the TME. This date/time is stamped as system date by the TME database before insert.
- *COMPONENT*.REVISED_DATE: The date/time (GMT) when this revision of this component was officially recorded in the TME. This is different than the ADDED_DATE, which is the system date/time the row was inserted or updated.
- *COMPONENT*.EFFECTIVE_DATE: The date when this revision of this component became effective in the source code system. This date must be provided by the source code system, or it is set to null.
- *COMPONENT*.EXPIRATION_DATE: The date when this revision of this component stopped being effective in the source code system. This date must be provided by the source code system, or it is set to null.
- *COMPONENT*.VIEW_XML: XML string that holds a delimited list of TCIDs for the "views" in which this component participates. Views are TFT concepts and are stored in the TFT namespace (e.g., TFTDental, TFTER, TFTProjectX, etc.).
- *COMPONENT*.COMMENT_XML: XML string that holds author comments regarding this component (e.g., <submitted_by> IHC </submitted_by > <comment> For Allergen Id domain and Allergen Component Code domain </ comment > <created_by> SC Shakib </created_by >).
- *COMPONENT*.REV_COMMENT_XML: XML string that holds a human-readable comment regarding the nature of the current revision of this component.

- *COMPONENT*.PREVIOUS_RID: References the *COMPONENT_REV*.RID of the previous revision of this component in the corresponding component revision table. If PREVIOUS_RID is null, this is the first revision of this component.
- *COMPONENT*.NEXT_RID: References the *COMPONENT_REV*.RID of the next revision of this entity in the corresponding component revision table. If NEXT_RID is null, this component is the most-future revision. When NEXT_RID is not null for the currently-effective revision stored in the *COMPONENT* table, it indicates that there is a more-future revision of the component that has not yet become effective (a future-effective revision).
- *COMPONENT*.RID: Revision Identifier; Every time an insert or update is performed against a core table in the TME, a trigger writes a copy of the component revision in both the core table (e.g., ENTITY, RSFORM, etc.) and the corresponding revision table (e.g., ENTITY_REV, RSFORM_REV, etc.), and assigns a new RID. RID references the RID_CONTROL table to ensure that the RID is unique TME-wide. There are no deletes allowed in TME; if a component needs to be removed from TME, its status is changed to TFTInactive and it persists in the revision tables.

9.2.2. RSFORM (Related Surface Form) and RSFORM_LINK tables. The RSFORM table holds one-to-many representations (also referred to as surface forms) that are associated with a single entity. In most cases, RSFORM will hold a display and code for a source code system entity. In the case of the TFT, the related representations include displays and codes that are considered synonyms of concepts in the ENTITY table.

The RSFORM table includes the following fields (see Table 7):

- RSFORM.RSFORM_ID: Primary key of the RSFORM table. It uniquely identifies the related surface form in the RSFORM table.
- RSFORM.REPRESENTATION: A text string that is associated with the referenced entity through the RSFORM.TCID. The string is also referred to as a surface form and it can be in any format (e.g., text, numeric, alpha-numeric) and of any type/context (e.g., code, display).
- RSFORM.UP_REPRESENTATION: The surface form in all uppercase characters. This field facilitates case-insensitive comparisons of surface forms and improves performance of string matching during mapping and diffing.
- RSFORM.PREFERRED_REP: A Boolean flag that indicates whether or not this surface form is the preferred representation for the referenced entity in the referenced context.
- RSFORM.CONTEXT_TCID: References a TCID from the ENTITY table for a TFT concept that denotes the type of surface form (e.g., TFTSite1InterfaceCode, TFTSite1DefaultDisplay, TFTSNOMEDCTFullSpecifiedName, etc.).
- RSFORM.CASE_SENSITIVE: Boolean value that flags whether REPRESENTATION is case-sensitive. Sometimes, changes in case imply different meanings. This flag is checked before doing a case-insensitive match of ENTITY.CID or RSFORM.REPRESENTATION.
- RSFORM.LANGUAGE_TCID: References a TCID from the ENTITY table for a TFT concept that denotes the language of this surface form. Language concepts are stored in the TFT namespace (e.g., TFTEnglish, TFTFrench, etc.).

- RSFORM.TCID: Foreign key that links this surface form to an entity in the ENTITY table. Many surface forms can be associated with a single entity.

The RSFORM_LINK table is used to relate representations for a single entity. For example, there may be multiple codes from a single source code system, each with its own display, on one entity. In this case, RSFORM_LINK is used to join each code with its corresponding display. RSFORM_LINK is also used to join lexical variants of a term (e.g., “Mice” | “plural form of” | “Mouse”).

The RSFORM_LINK table includes the following fields (see Table 8):

- RSFORM_LINK.RSFORM_LINK_ID: Primary key of the RSFORM_LINK table. It uniquely identifies the representation link triplet in the RSFORM_LINK table.
- RSFORM_LINK.RSFORM_ID_1: References an RSFORM_ID from the RSFORM table for the first surface form in a representation link triplet (e.g., “Mice”).
- RSFORM_LINK.RSFORM_LINK_TCID: References a TCID from the ENTITY table for a TFT concept that denotes the kind of link between RSFORM_ID_1 and RSFORM_ID_2. Surface form link concepts (i.e., concepts that represent links) are stored in the TFT namespace (e.g., TFTPluralFormOf).
- RSFORM_LINK.RSFORM_ID_2: References an RSFORM_ID from the RSFORM table for the second surface form in a representation link triplet (e.g., “Mouse”).

9.2.3. ENTITY_RELATIONSHIP table. The ENTITY_RELATIONSHIP table is used to provide structure to the TME by establishing meaningful links among entities

in a single name space. Single entities can have multiple different types of relationships to multiple other entities. ENTITY_RELATIONSHIP is used to organize entities in to classes, pick lists, and other logical or operational groupings. The relationship between two entities is a TFT concept that has meaning and direction. Two broad categories of relationship types are maintained in the TME: 1) Hierarchical relationships, which are relationships between two entities where there is inheritance of attributes from the "parent" entity to the "child"; and 2) Semantic relationships, which include any meaningful relationship between two entities but do not imply inheritance.

The ENTITY_RELATIONSHIP table includes the following fields (see Table 9):

- ENTITY_RELATIONSHIP.ENTITY_RELATION_ID: Primary key of the ENTITY_RELATIONSHIP table. It uniquely identifies an entity relationship triplet in the ENTITY_RELATIONSHIP table. It is a foreign key in the ENTITY_RELATIONSHIP revision table (ENTITY_RELATIONSHIP_REV).
- ENTITY_RELATIONSHIP.TCID_1: References a TCID from the ENTITY table for the first entity in a relationship triplet (e.g., "Cytomegalovirus" (first entity) | "is-a" (relationship) | "Virus" (second entity)).
- ENTITY_RELATIONSHIP.RELATIONSHIP_TCID: References a TCID from the ENTITY table for a TFT concept that denotes the kind of relationship between the first entity and the second entity. Relationship concepts (i.e., concepts that represent relationships) are stored in the TFT namespace (e.g., "Cytomegalovirus" (first entity) | "is-a" (relationship) | "Virus" (second entity)).

- ENTITY_RELATIONSHIP.TCID_2: References a TCID from the ENTITY table for the second entity in a relationship triplet (e.g., “Cytomegalovirus” (first entity) | “is-a” (relationship) | “Virus” (second entity)).

9.2.4. ENTITY_LINK table. The ENTITY_LINK table holds mappings among entities in different source code system namespaces and to the TFT. This table is used to hold concept and purpose-built maps among code systems and is referenced for the creation of TFT concepts. It is not intended to store relationships among entities in the same namespace.

The ENTITY_LINK table includes the following fields (see Table 10):

- ENTITY_LINK.ENTITY_LINK_ID: Primary key of the ENTITY_LINK table. It uniquely identifies an entity link triplet in the ENTITY_LINK table. It is a foreign key in the ENTITY_LINK revision table (ENTITY_LINK_REV).
- ENTITY_LINK.TCID_1: References a TCID from the ENTITY table for the first entity in a map link triplet (e.g., “SNOMEDCTCytomegalovirus” (first entity in SNOMED CT namespace) | “is-clinically-equivalent-to” (link type) | “TFTCMV” (second entity in TFT namespace)).
- ENTITY_LINK.ENTITY_LINK_TCID: References a TCID from the ENTITY table for a TFT concept that denotes the kind of link between the first entity and the second entity. Concepts that represent map link types are stored in the TFT namespace (e.g., TFTIsNarrowerThan, TFTIsEquivalentTo, TFTIsBroaderThan).
- ENTITY_LINK.TCID_2: References a TCID from the ENTITY table for the second entity in a map link triplet (e.g., “SNOMEDCTCytomegalovirus” (first

entity in SNOMED CT namespace) | “is-clinically-equivalent-to” (link type) | “TFTCMV” (second entity in TFT namespace)).

- ENTITY_LINK.MAP_SET_TCID: References a TCID from the ENTITY table for a TFT map set concept. MAP_SET_TCID is used to group a set of mappings in a many-to-many or many-to-one mapping. This is used for composition and decomposition of molecular content (content comprised of multiple atomic concepts). It does not specify the order of atomic coded attributes when doing composition.
- ENTITY_LINK.MAP_SCORE: Used for ranking mappings. When more than one mapping of the same type is created between a source entity and multiple target entities, the MAP_SCORE is used to rank them. This is populated either by attribute comparison (more defining attributes in common between source and target equals a higher score) or is based on an SMEs judgment with regard to the accuracy/granularity of the maps.
- ENTITY_LINK.RULE_XML: XML string used to express the rules that were followed to create the map link or conditions for which the mapping is valid.

9.2.5. ENTITY_ATTRIBUTE table. The ENTITY_ATTRIBUTE table holds name-value pairs that are defining attributes of entities. This table is similar in structure to the PROPERTY table in that attributes can be organized hierarchically or grouped into sets and the values in the name-value pair can be specified as being coded, numeric, alpha, or XML. The concept attribute table is the primary reference for semantic mapping. Attribute sets are used to define entities, and then those sets are compared to determine if they match and a map link can be created.

The ENTITY_ATTRIBUTE table includes the following fields (see Table 11):

- ENTITY_ATTRIBUTE.ATTRIBUTE_ID: Primary key of the ENTITY_ ATTRIBUTE table. It uniquely identifies an entity attribute in the ENTITY_ ATTRIBUTE table. It is a foreign key in the ENTITY_ ATTRIBUTE revision table (ENTITY_ ATTRIBUTE _REV).
- ENTITY_ATTRIBUTE.SET_ID: Identifies the attribute set to which this attribute belongs. Used to group attributes. For example, it is necessary to associate a strength with a particular ingredient in drugs that have multiple active ingredients. SET_ID is used to associate each ingredient with the appropriate strength.
- ENTITY_ATTRIBUTE.ATTRIBUTE_TCID: References a TCID from the ENTITY table for a TFT concept that denotes the type of attribute this attribute is. Attribute type concepts are stored in the source terminology namespace in which they are used (e.g., LOINCAttributeAxis1), and are provided by, or derived from, source code system data.
- ENTITY_ATTRIBUTE.VALUE_TYPE_TCID: References a TCID from the ENTITY table for a TFT concept that denotes the type of value associated with this attribute. Value type concepts are stored in the TFT namespace (e.g., TFTCodedValue, TFTAlphaNumericValue, TFTNumericValue, and TFTXMLValue).
- ENTITY_ATTRIBUTE.CODED_VALUE_TCID: References a TCID from the ENTITY table for an entity that denotes the value of this attribute if ENTITY_ATTRIBUTE.VALUE_TYPE_TCID is TFTCodedValue. Coded value

entities are stored in the source terminology namespace in which they are used (e.g., LOINCGlucose), and are provided by, or derived from, source code system data.

- ENTITY_ATTRIBUTE.NUMERIC_VALUE: Populated with a numeric value for the attribute name-value pair, when ENTITY_ATTRIBUTE.VALUE_TYPE_TCID is TFTNumericValue.
- ENTITY_ATTRIBUTE.ALPHA_VALUE: Populated with an alphanumeric value for the attribute name-value pair, when ENTITY_ATTRIBUTE.VALUE_TYPE_TCID is TFTAlphaNumericValue.
- ENTITY_ATTRIBUTE.UP_ALPHA_VALUE: An all uppercase representation of ENTITY_ATTRIBUTE.ALPHA_VALUE used to facilitate matching during diffing and mapping.
- ENTITY_ATTRIBUTE.VALUE_UNIT_TCID: References a TCID from the ENTITY table for a TFT concept that denotes the unit in which the value of this attribute is expressed. Value unit concepts are stored in the TFT namespace (e.g., TFTMilligrams).
- ENTITY_ATTRIBUTE.XML_VALUE: Populated with an XML string for the attribute name-value pair, when ENTITY_ATTRIBUTE.VALUE_TYPE_TCID is TFTXMLValue.
- ENTITY_ATTRIBUTE.PARENT_ATTRIBUTE_ID: ATTRIBUTE_ID of the parent entity attribute if this is an attribute of another entity attribute; null otherwise. This is the primary method of combining several attributes into a hierarchical group.

9.2.6. PROPERTY table. The PROPERTY table stores name-value pairs that are properties of entities, representations, relationships, mapping links, representation links, and other properties. This table is not meant to hold defining attributes of entities but rather additional metadata associated with any component, including things like source code system identifiers. An example of a property would be whether or not a specified representation is the preferred representation. Properties can be organized hierarchically or grouped into sets. The values in the name-value pair can be specified as being coded, numeric, alpha, or XML. The property table gives the flexibility to model additional relevant metadata that may be in a source terminology but is not explicitly called out in the TME schema.

The PROPERTY table includes the following fields (see Table 12):

- PROPERTY.PROPERTY_ID: Primary key of the PROPERTY table. It uniquely identifies properties which can be associated with any component in the TME, including other properties. It is a foreign key in the PROPERTY revision table (PROPERTY_REV).
- PROPERTY.SET_ID: Identifies the property set to which this property belongs.
- PROPERTY.PROPERTY_TCID: References a TCID from the ENTITY table for an entity that describes this property. Property type entities are stored in the source terminology namespace in which they are used and are provided by, or derived from, source code system data (e.g., SNOMEDCTID).
- PROPERTY.VALUE_TYPE_TCID: References a TCID from the ENTITY table for the TFT concept that denotes the type of value this property has. This field is

constrained to the same domain of TFT concepts as ENTITY_ATTRIBUTE.VALUE_TYPE_TCID.

- PROPERTY.CODED_VALUE_TCID: References a TCID from the ENTITY table for an entity that denotes the value of this property if PROPERTY.VALUE_TYPE_TCID is TFTCodedValue. Coded value entities are stored in the source terminology namespace in which they are used and are provided by, or derived from, source code system data.
- PROPERTY.NUMERIC_VALUE: Populated with a numeric value for the property name-value pair, when PROPERTY.VALUE_TYPE_TCID is TFTNumericValue.
- PROPERTY.ALPHA_VALUE: Populated with an alphanumeric value for the property name-value pair, when PROPERTY.VALUE_TYPE_TCID is TFTAlphaNumericValue.
- PROPERTY.UP_ALPHA_VALUE: An all uppercase representation of PROPERTY.ALPHA_VALUE used to facilitate matching during diffing.
- PROPERTY.VALUE_UNIT_TCID: References a TCID from the ENTITY table for a TFT concept that denotes the unit in which the value of this property is expressed. Value unit concepts are stored in the TFT namespace.
- PROPERTY.XML_VALUE: Populated with an XML string for the attribute name-value pair, when PROPERTY.VALUE_TYPE_TCID is TFTXMLValue.
- PROPERTY.PARENT_TABLE_TCID: References a TCID from the ENTITY table for a TFT concept that denotes the TME table of the PROPERTY.PARENT_ID.

PROPERTY.PARENT_ID: ID of the component with which this property modifies or is associated. This can be any of the core table primary keys including PROPERTY_ID. It is combined with PROPERTY.PARENT_TABLE_TCID as a compound foreign key.

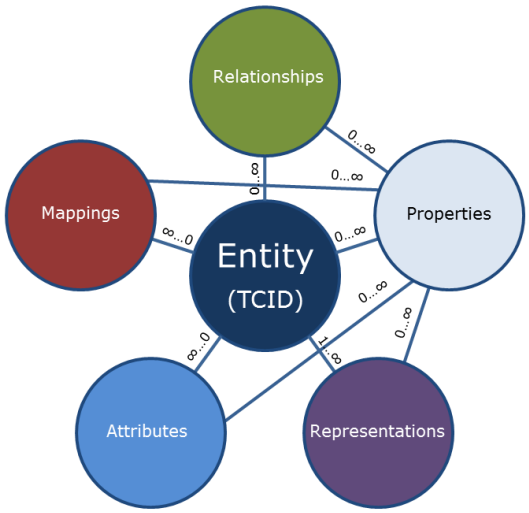


Figure 10: TME Logical Model.

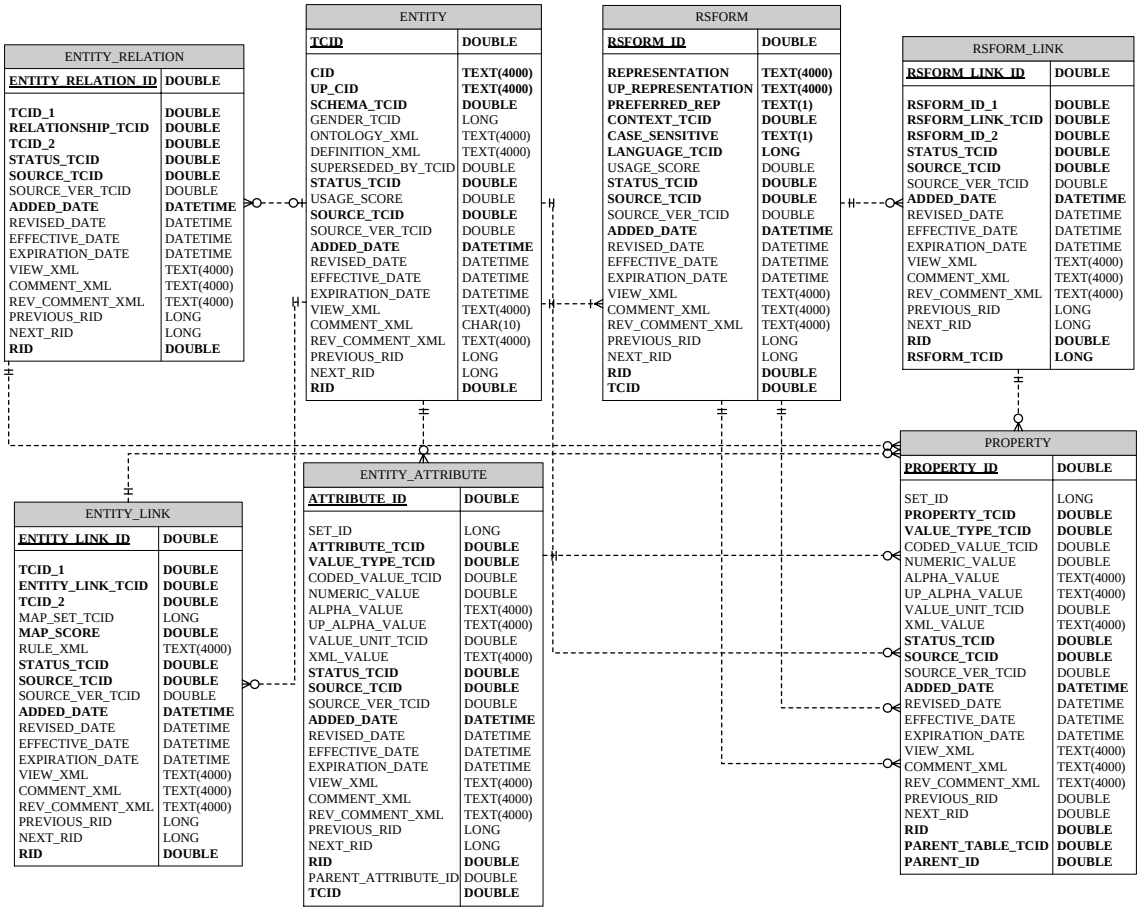


Figure 11: TME Vocabulary Server Core Tables Simplified Schema. This is a simplified TME schema that includes just the core current revision TME tables.

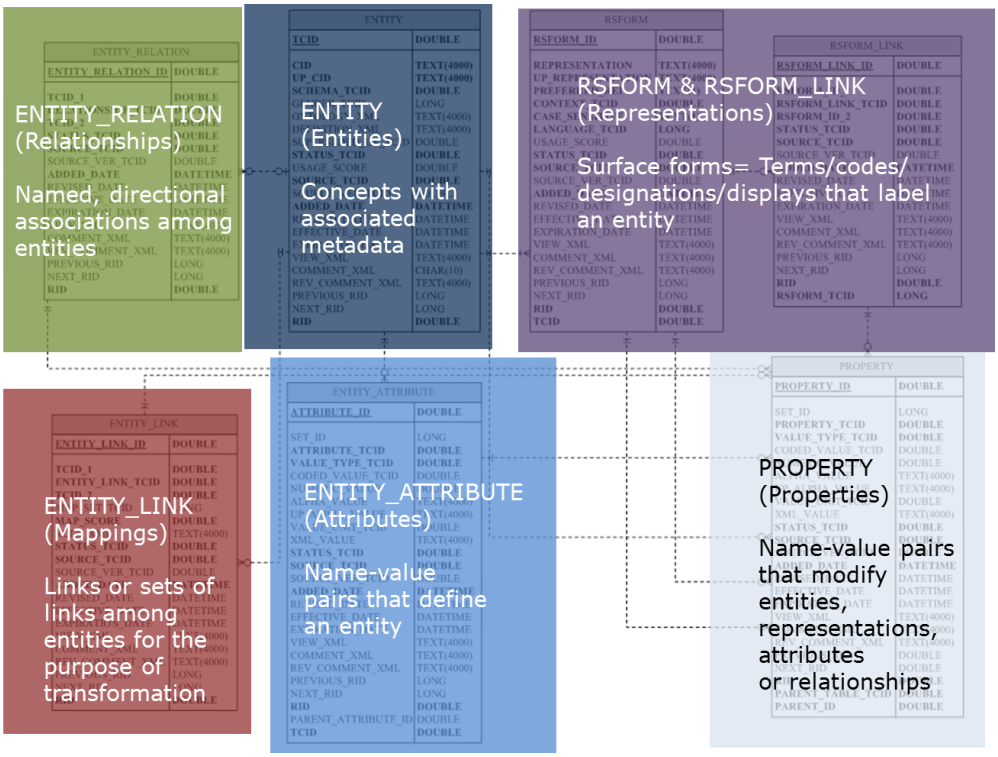


Figure 12: Simplified TME Physical Model with TME Logical Model Overlaid.

Table 6: ENTITY Table. Field names and data types.

ENTITY	
TCID	**DOUBLE**
CID	**TEXT(4000)**
UP_CID	**TEXT(4000)**
SCHEMA_TCID	**DOUBLE**
GENDER_TCID	LONG
ONTOLOGY_XML	TEXT(4000)
DEFINITION_XML	TEXT(4000)
SUPERSEDED_BY_TCID	DOUBLE
STATUS_TCID	**DOUBLE**
USAGE_SCORE	DOUBLE
SOURCE_TCID	**DOUBLE**
SOURCE_VER_TCID	DOUBLE
ADDED_DATE	**DATETIME**
REVISED_DATE	DATETIME
EFFECTIVE_DATE	DATETIME
EXPIRATION_DATE	DATETIME
VIEW_XML	TEXT(4000)
COMMENT_XML	CHAR(10)
REV_COMMENT_XML	TEXT(4000)
PREVIOUS_RID	LONG
NEXT_RID	LONG
RID	**DOUBLE**

Table 7: RSFORM Table. Field names and data types.

RSFORM	
RSFORM_ID	**DOUBLE**
REPRESENTATION	**TEXT(4000)**
UP_REPRESENTATION	**TEXT(4000)**
PREFERRED_REP	**TEXT(1)**
CONTEXT_TCID	**DOUBLE**
CASE_SENSITIVE	**TEXT(1)**
LANGUAGE_TCID	**LONG**
USAGE_SCORE	DOUBLE
STATUS_TCID	**DOUBLE**
SOURCE_TCID	**DOUBLE**
SOURCE_VER_TCID	DOUBLE
ADDED_DATE	**DATETIME**
REVISED_DATE	DATETIME
EFFECTIVE_DATE	DATETIME
EXPIRATION_DATE	DATETIME
VIEW_XML	TEXT(4000)
COMMENT_XML	TEXT(4000)
REV_COMMENT_XML	TEXT(4000)
PREVIOUS_RID	LONG
NEXT_RID	LONG
RID	**DOUBLE**
TCID	**DOUBLE**

Table 8: RSFORM_LINK Table. Field names and data types.

RSFORM_LINK	
RSFORM_LINK_ID	**DOUBLE**
RSFORM_ID_1	**DOUBLE**
RSFORM_LINK_TCID	**DOUBLE**
RSFORM_ID_2	**DOUBLE**
STATUS_TCID	**DOUBLE**
SOURCE_TCID	**DOUBLE**
SOURCE_VER_TCID	DOUBLE
ADDED_DATE	**DATETIME**
REVISED_DATE	DATETIME
EFFECTIVE_DATE	DATETIME
EXPIRATION_DATE	DATETIME
VIEW_XML	TEXT(4000)
COMMENT_XML	TEXT(4000)
REV_COMMENT_XML	TEXT(4000)
PREVIOUS_RID	LONG
NEXT_RID	LONG
RID	**DOUBLE**
RSFORM_TCID	**LONG**

Table 9: ENTITY_RELATION Table. Field names and data types.

ENTITY_RELATION	
ENTITY_RELATION_ID	**DOUBLE**
TCID_1	**DOUBLE**
RELATIONSHIP_TCID	**DOUBLE**
TCID_2	**DOUBLE**
STATUS_TCID	**DOUBLE**
SOURCE_TCID	**DOUBLE**
SOURCE_VER_TCID	DOUBLE
ADDED_DATE	**DATETIME**
REVISED_DATE	DATETIME
EFFECTIVE_DATE	DATETIME
EXPIRATION_DATE	DATETIME
VIEW_XML	TEXT(4000)
COMMENT_XML	TEXT(4000)
REV_COMMENT_XML	TEXT(4000)
PREVIOUS_RID	LONG
NEXT_RID	LONG
RID	**DOUBLE**

Table 10: ENTITY_LINK Table. Field names and data types.

ENTITY_LINK	
ENTITY_LINK_ID	**DOUBLE**
TCID_1	**DOUBLE**
ENTITY_LINK_TCID	**DOUBLE**
TCID_2	**DOUBLE**
MAP_SET_TCID	LONG
MAP_SCORE	**DOUBLE**
RULE_XML	TEXT(4000)
STATUS_TCID	**DOUBLE**
SOURCE_TCID	**DOUBLE**
SOURCE_VER_TCID	DOUBLE
ADDED_DATE	**DATETIME**
REVISED_DATE	DATETIME
EFFECTIVE_DATE	DATETIME
EXPIRATION_DATE	DATETIME
VIEW_XML	TEXT(4000)
COMMENT_XML	TEXT(4000)
REV_COMMENT_XML	TEXT(4000)
PREVIOUS_RID	LONG
NEXT_RID	LONG
RID	**DOUBLE**

Table 11: ENTITY_ATTRIBUTE Table. Field names and data types.

ENTITY_ATTRIBUTE	
ATTRIBUTE_ID	**DOUBLE**
SET_ID	LONG
ATTRIBUTE_TCID	**DOUBLE**
VALUE_TYPE_TCID	**DOUBLE**
CODED_VALUE_TCID	DOUBLE
NUMERIC_VALUE	DOUBLE
ALPHA_VALUE	TEXT(4000)
UP_ALPHA_VALUE	TEXT(4000)
VALUE_UNIT_TCID	DOUBLE
XML_VALUE	TEXT(4000)
STATUS_TCID	**DOUBLE**
SOURCE_TCID	**DOUBLE**
SOURCE_VER_TCID	DOUBLE
ADDED_DATE	**DATETIME**
REVISED_DATE	DATETIME
EFFECTIVE_DATE	DATETIME
EXPIRATION_DATE	DATETIME
VIEW_XML	TEXT(4000)
COMMENT_XML	TEXT(4000)
REV_COMMENT_XML	TEXT(4000)
PREVIOUS_RID	LONG
NEXT_RID	LONG
RID	**DOUBLE**
PARENT_ATTRIBUTE_ID	DOUBLE
TCID	**DOUBLE**

Table 12: PROPERTY Table. Field names and data types.

PROPERTY	
PROPERTY_ID	**DOUBLE**
SET_ID	LONG
PROPERTY_TCID	**DOUBLE**
VALUE_TYPE_TCID	**DOUBLE**
CODED_VALUE_TCID	DOUBLE
NUMERIC_VALUE	DOUBLE
ALPHA_VALUE	TEXT(4000)
UP_ALPHA_VALUE	TEXT(4000)
VALUE_UNIT_TCID	DOUBLE
XML_VALUE	TEXT(4000)
STATUS_TCID	**DOUBLE**
SOURCE_TCID	**DOUBLE**
SOURCE_VER_TCID	DOUBLE
ADDED_DATE	**DATETIME**
REVISED_DATE	DATETIME
EFFECTIVE_DATE	DATETIME
EXPIRATION_DATE	DATETIME
VIEW_XML	TEXT(4000)
COMMENT_XML	TEXT(4000)
REV_COMMENT_XML	TEXT(4000)
PREVIOUS_RID	DOUBLE
NEXT_RID	DOUBLE
RID	**DOUBLE**
PARENT_TABLE_TCID	**DOUBLE**
PARENT_ID	**DOUBLE**

10. MANAGING TERMINOLOGY CONTENT WITHIN THE TME

Now that the types of terminology actors and how they use the terminology have been defined, this section will describe the flow of data through the TME. In the TME, the life cycle of terminology is centered on the process of concept mapping to the federated terminology (TFT) to integrate various source terminologies. This type of mapping also requires the ability to author new content used to organize terminology for a particular purpose or create new concepts, relationships, designations, and/or codes. Concept mapping and new content creation have the following high-level steps:

1. Identify a code or term/set-of-terms used to label a concept in one code system, referred to as the "source."
2. Use code attribute information from the source to determine context and identify the underlying concept(s) for the term/phrase/code; metadata can be just a string of text or a set of supplied/derived attributes.
3. Link the concept in the source code system to an equivalent concept in another code system, referred to as the "target"; in the TME, the target code system is typically the TME's federated terminology (TFT).

Concept mapping is not a process of linking terms; terms are merely used as descriptions/labels that have a many-to-one relationship to concepts. Concept mapping is the process of linking semantically equivalent concepts from a source code system to a

target code system. The mapping links are directional (from source to target) and can be qualified as being either "broader than," "narrower than," or "equivalent."

The early steps in the mapping process are different in detail based on the nature of the source code system (e.g., standard or local terminology) but in general, they follow an Extract, Transform, and Load (ETL) process. Terms are extracted from the source, transformed to the TFT schema, and loaded into a staging environment to prepare for mapping (see Figure 13). Preparing a new source and going through the ETL process is typically one of the most challenging steps in the integration process. The goal of this phase of integration is to achieve semantic alignment of concepts in the source to the TMEs canonical terminology model. It is not to achieve complete data integration or concept mapping. Because of this, not all source code attributes are integrated in the TFT. This first integration step requires understanding the source data schema, reconciling the disparate terminology models of the source and target, and mapping code attribute fields in the source to code attribute fields in the target.[72]

10.1. Customization Mapping: Data Extraction for LIT

Data extraction is the process of querying master files (see Table 1) from source systems to pull data necessary for mapping the LIT into the TME. These master files contain terms that are used to encode data in legacy systems. Multiple master files are referenced to encode a single record in the legacy system. It is typically a custom effort to determine how to link these files/tables and extract the data required for mapping. This description of the data extraction steps is only relevant for local terminologies for which there is often not a formal logical model and context is established either through HL7 messages, source system attributes, or site SMEs.

In many cases, obsolete data remain in master files, along with ambiguous or duplicate terms and invalid/inactivated concepts. All of these idiosyncrasies in the master files must be resolved during the process of validating and grooming the source data. Validation involves checking the status of LIT terms in the legacy system to determine if they are active. Grooming involves normalizing duplicate terms, cleaning up special characters, and expanding acronyms.

Another way to both validate and obtain the necessary data elements in the LIT is by monitoring outbound HL7 messages from the legacy systems. The local interface codes and descriptions can be mined from HL7 fields, which supply context. So, for example, lab results codes can be found in the HL7 observation segment (OBX) field three (OBX3) of a lab message. A utility can monitor HL7 messages for a period of time and collect the data required for mapping. Regenstrief Institute has created a LOINC mapping tool that operates in this manner.[46] The advantage of this approach is that only active and common data elements in the LIT are collected for mapping, reducing the overall mapping effort. The disadvantage is that relevant data that are rarely instantiated or seasonal in nature may not appear in HL7 messages during the period of data collection.

At the end of the data extraction process, lists of relevant terms, in context, have been collected. Context establishes where and how the term is used and how much information is being included in the underlying concept (e.g., the degree to which it may be precoordinated) and makes it possible to determine, in part or whole, the meaning of the underlying concept (e.g., is this a lab test, laboratory result, unit of measure, or specimen and is this term a code or a display). Knowing context is critical to concept mapping; without it, it is not possible to link terms to the appropriate concepts. So, for

example, the term "cold" cannot be mapped without knowing: 1) if "cold" is a code or a display and 2) if it is a temperature (sensory perception), a diagnosis (pulmonary diagnosis), or a finding (upper respiratory viral infection).[73]

In some cases, more information than a term and context for the source concept is required in order to create an accurate mapping to the target. For example, mapping laboratory results to the six-axis LOINC model (LOINC terminology model = <component/analyte> {required} : <kind of property> {required} : <time aspect> {required} : <system type> {required} : <scale> {required} : <method> {optional})[46] requires the following associated information: lab result name, specimen, result type (numeric or alpha), sample results with units of measure, and method. Since the data model in the legacy systems almost never maps directly to the LOINC model, mapping lab results requires deriving and in some cases assuming values to populate the six axes with what can be extracted from the legacy systems. The work to derive additional values and postcoordinate the attribute information is accomplished during the next step in the TME ETL process.

10.2. Data Transformation

Data transformation involves mapping the concept model of the source to the concept model of the federated terminology in the TME. The TME uses a "canonical" model that serve as a model "interlingua."[74] Standard sources like LOINC as well as LITs are mapped to the canonical concept model as well as domain-specific data models within the TFT. In the case of a laboratory results LIT (see Figure 14), it requires extracting the lab result name and associated specimen, result type, sample results with

units of measure, and method, then applying rules to build a data model in the LIT namespace that mirrors the data model of the TFT (see Figure 15).

10.3. Staging and "Diffing"

Results of the data transformation are loaded into a staging environment. The staging environment holds source code systems that have been converted to the TME data model, but have not yet been mapped/integrated in to the TME. Content in the staging environment has not been augmented with additional metadata or linked to the TFT. It is native content that has been transformed into the TME data model.

The staging environment maintains all previous versions of each source code system. This provides a history that can be queried to determine how a particular source looked at any prior point-in-time. It is also used to establish what changes were made in the latest version of a code system. A code system update can result in a variety of changes (additions, modifications, and/or deletions) to concepts, codes, displays, relationships, and definitions. Assessing the delta in new versions of a code system in the TME is referred to as "diffing."

Diffing is the process of comparing a new version of source code systems to the previous version to determine what changed (what is different). Occasionally, source code systems will provide this information, but in most cases, it is not available or it is not provided with enough detail to support the TME mapping process. Unique identifiers are assigned to each attribute of a source code system data element. The identifiers are used as an index for diffing. New concepts or changes that may alter the semantics of an existing concept require mapping review.

10.4. Distributing Work to Appropriate SMEs

The result of diffing is a set of terms from the source code system that are new and must be mapped or have previous mappings to the TME that must be reviewed because of changes in the source. This work goes into mapping queues for the appropriated SMEs. Assignments are tracked at the row level to allow very large mapping tasks to be split between multiple SMEs.

10.5. Initial String Matching

There is initial automated matching of source code system terms to the TME. This involves various types of string matching techniques that are incorporated into a tool called Hypersearch[75]:

- Synonym Matching: Multiple synonymous terms are associated with a single concept (e.g., Varicella, VZV, Chickenpox). The additional synonyms make it easier to find concepts during the matching process.
- Phonetic Algorithms: Soundex is the phonetic algorithm used in the TME. It is a standard capability of Oracle and Microsoft SQL Server (MS SQL) databases. The Soundex algorithm attempts to encode terms that have the same English pronunciation in the same way.[75] This helps to address minor spelling errors in the source terms.
- Lexical Variant Matching: Lexical variants are different forms or spellings of the same term (e.g., mice/mouse, man/men, honor/honour, email/e-mail). The Lexical Variant Generator (LVG),[76] part of the UMLS Specialist Lexical Tools, is used to build an index of variant terms on concepts in the TFT. Source terms are then matched against the index.

At the end of initial automated mapping, three categories of terms that must be further evaluated remain:

1) Terms that match with a low probability score from string matching.
2) Terms for which there are multiple potential matches in the target.
3) Terms for which there are no suggested matches in the target.

The remaining terms must be manually reviewed by SMEs through a process of interrater agreement. For category one and two matches, all candidates are provided to the SMEs for review.

10.6. Attribute Matching

In some domains, automated string matching is sufficient to map the majority of the terms from the source code system. In other domains, like laboratory and pharmacy, a semantic matching technique is applied. Pharmacy, laboratory, and observation data are different from many other clinical terminology domains in that observations, clinical drug concepts, and laboratory results can be readily and precisely defined by a set of attributes.[77] This allows for a more detailed data model for both lab results and drugs that more closely approaches the "universals" and ontology knowledge representation model discussed earlier in Section 3.4.[39] Once an attribute set has been defined for a drug or lab result it can be compared to those that already exist in the TFT.

Using drugs as an example to describe the semantic matching approach, a clinical drug can be defined as having ingredient, strength, form, and route as core attributes (see Figure 16).[78] In addition, drugs have many brand names, abbreviations, synonyms, and packaging information that sometimes need to be taken into account for unique identification. Instead of matching representations for the entire drug concept at once,

attributes are matched. Parsing out the correct attributes from a drug representation is challenging, but once all the attributes have been identified, they can be used to find exact and near (but semantically equivalent) concept matches. This enforces mapping consistency by removing human variability due to case-by-case judgment calls.

Separate, locally developed drug and lab mapping tools were created that utilized synonyms of TME concepts to parse precoordinated strings into a set of attributes.[78] Each drug concept is sent through a parser, which references a knowledge base to identify ingredient, strength, form, and route. The knowledge base of the pharmacy mapping tool supplies rules for parsing and matching each drug attribute. It is organized to support multiple synonyms, brand name to generic ingredient conversions, and form and route hierarchies.

10.6.1. Mapping ingredient. When a brand name is identified in a drug representation, it is translated into the appropriate generic name by referencing the TME. The generic ingredient is then matched against a comprehensive list of ingredients. If no match is found, the tool will attempt to switch the generic and brand names and search again.

10.6.2. Mapping form and route. The form and route hierarchies and synonyms are used to broaden the scope of candidate matches. For example, the term CAPS in a drug refers to capsule. Some synonyms used for matching include: CAP, CAPSULE, CAPSULES, etc. Additional potential matches are identified by referencing the hierarchy for all of the more specific forms of capsule: CAP SEQ, CAP SPRINK, CAP W/DEV, CAP DS PK, CAP MPHASE, CAPSULE DR, CAPSULE CR, CAPSULE SA, etc.

Drawing a clear distinction between form and route is challenging. Often form implies route and vice versa. Preferred attribute scores are used to assign form and route for matching purposes, when there is confusion in differentiating them. For example, injection is often used as both a form and a route. If injection and intravenous both appear in a drug representation, a separate, tool-specific, knowledge base is referenced to determine that intravenous is a preferred route and injection is a preferred form.

In the first mapping review of a large enterprise's formularies, 151,854 unique drug representations were evaluated. Using the semantic matching approach, 50.8% were identified as exact matches to existing TME concepts, 35.5% were approximate matches, and 13.7% were unmatched. Mapping speed was improved by 29% over the previous string matching process and consistency among SMEs was enhanced because of rules enforced by the approach.[78]

10.7. SME Interrater Agreement

The process of interrater agreement involves establishing consensus among SMEs for mapping decisions that require judgment. Its purpose is to reduce variability and increase accuracy. The validation of initial mapping is performed by one SME. A second SME evaluates the first SME's recommended mapping actions (see Figure 17). If the second SME does not agree with the recommended action from the first SME, it is returned to the first SME with a justification. Recommended actions that make more than one loop in the mediation cycle between the first and second SME are escalated for group review. Since the federated terminology is referenced in many different ways by the EHR, and the data in the EHR are used to drive decision making, "small" errors in the terminology are magnified multiple fold in the patient data and can be a patient safety

issue. This risk justifies additional effort to validate mappings. Although accuracy is of critical importance, consistency (reducing variability) can be an even more significant factor over time. If a mapping is incorrect, but consistently incorrect, it can be resolved both in the vocabulary server and in systems that referenced terminology services to store data. If it is only occasionally incorrect, resolving the issue in historical data and the vocabulary server may not even be possible. Achieving consensus, documenting the logic behind mapping decisions, and consistently applying the logic, makes it possible to improve accuracy and data quality over time. Without consistency, accuracy will decline over time.

During the SME review, questions about the source content are collected. Some questions are related to the accuracy/specificity of a proposed mapping. These types of questions are typically addressed to other TME SMEs. Other questions are related to the source code system terms. These types of questions are addressed to the source organizations. Both classes of questions are managed in the Question-Answer process.

10.8. Questions and Answers: Between Source Organization and SMEs

The question-answer process is a cycle that can go through multiple iterations. It can sometimes be very time-consuming and involve multiple individuals across organizations. It is important to capture both the discussion and the final decision so that when similar future questions are raised, the effort to arrive at an answer is not repeated and a consistent solution can be applied. In the TME, the communication between SMEs and source organizations and the final decision are archived and indexed based on the local or standard code used by the source organization.

10.9. Loading and Maintenance in the TFT

Once a final decision is made by an SME, the corresponding mapping action is applied (see Table 13). However, purpose-built mappings are never truly "final." Because of semantic shift/drift in source terminologies and refinement or changes in the definition of the mapping purpose, mappings are constantly being reevaluated in the mapping maintenance process. Mappings can change if the meaning of the concepts from a source change, but the meaning of concepts in the TFT must remain the same.

Loading involves creating the necessary content in the TFT and populating a table in the TME that links source and target code systems. Figure 17 summarizes the concept mapping process.

10.10. Quality Assurance (QA) and Quality Control (QC) of TME

QA and QC methods, processes, and tools are used in the TME to help ensure data quality. The QA measures are focused on processes and the QC measures are focused on the TME content. The goal of these measures is to try to achieve the highest level of accuracy and consistency in data representation, but in a hierarchy of importance, consistency is more important than accuracy:

- Correct and consistent
- Incorrect and consistent
- Correct and inconsistent
- Incorrect and inconsistent

Consistency takes precedence over accuracy because ongoing QC will eventually uncover "errors." Errors are not always unintentional. Because mapping requires

judgment, an error may be a decision that was made at a previous point-in-time that is reconsidered.[79]

10.10.1. QA emphasis on process. QA measures are established among TME SMEs using 3rd party tools such as Microsoft SharePoint. A SharePoint site dedicated to the TME organizes and versions training and procedural documents and acts as a communication hub for SMEs. Issue tracking, topic-specific discussions, and calendars help to formalize communication with regard to process and status of work:

- SME training in informatics principles: SMEs have knowledge of the content area, but require training in informatics to help make consistent and well-informed mapping decisions. Understanding how the TME represents knowledge helps to answer questions such as:
    - Is this a new concept?
    - Is this a property or an attribute?
    - Is this terminology model adequate to identify this concept uniquely?
- Standard Operating Procedure (SOP) Documents: SOPs are detailed descriptions of the steps to follow for any manual TME processes. SOPs are continually updated and must be organized such that searches using phrases and keywords can be used to find documents.

In addition to process that are assimilated by SMEs through instruction, automated and interrater methods are employed through QC measures.

10.10.2. QC emphasis on content. QC measures are implemented in the TME through processes like interrater agreement and domain-specific tools. Some QC is

specific to a current "job." Other QC is ongoing and performed against the entire federated terminology. The TME QC measures include:

- Interrater agreement: Previously described process using two SMEs to map and then validate content changes to TME.
- 3rd Party validation: In some cases, it is possible to get source organizations to validate mappings. The TME mappings are exported in a "human-readable" format and sent to the source organization for validation.
- Intersource agreement: In some cases, there are multiple sources for the same content or publically available mappings. These are used to validate mappings generated by TME SMEs. Discrepancies do not necessary indicate an error. They are used to flag a potential QC issue that must be evaluated.
- Audit History/Change control for all TME content: Every TME table has a corresponding revision table that tracks all changes. Changes can be audited and simple flags such as the number of changes in a particular version can be used to signal a potential QC issue.
- Domain-specific tools: Tools designed for creation and maintenance of a particular type of content (e.g., lab, pharmacy) are tailored to remove manual steps that might be necessary in a more general tool. This type of content control can also be implemented through a terminology model that enforces the creation of required attributes.
- Database Triggers and Constraints: The TME uses an Oracle relational database. Triggers are used to generate content automatically based on certain types of operations. For example, when content is added to a core table, triggers are used

to populate the corresponding revision table. Constraints prevent creation or modification of certain types of content. For example, a TME constraint prevents creation of more than one preferred display of a particular type.

In addition to the QC methods discussed here, implementation of a formal, comprehensive ontology in the TFT is a future quality control measure. Although attribute definitions are utilized in the TFT for semantic matching in the lab and pharmacy domains, a formal ontology has not been implemented. Ontologies are a way representing content that generates formal definitions that can be validated through machine processing.[80] Tools, such as protégé, are available for both the creation and validation of ontologies.[81] Such tools could be used for advanced quality control.

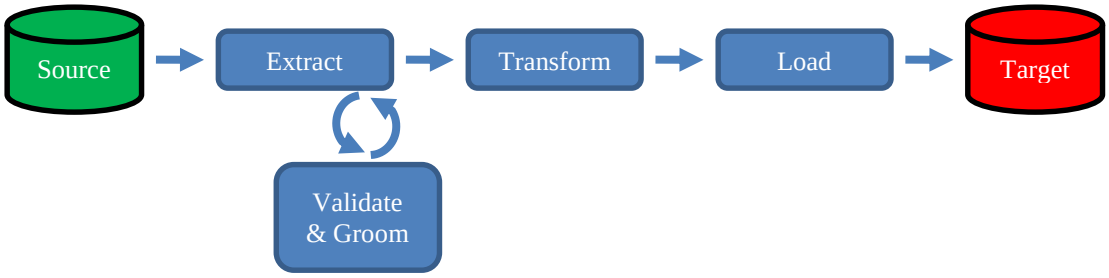


Figure 13: Steps in the TME ETL Process. Starting with source and ending in the TME staging tables (target). Source may be standard or LIT. In the case of LIT, source content will often require some manual grooming and reformatting by Subject Matter Experts (SMEs) before going through the transformation from the source physical model to the TFT physical model.

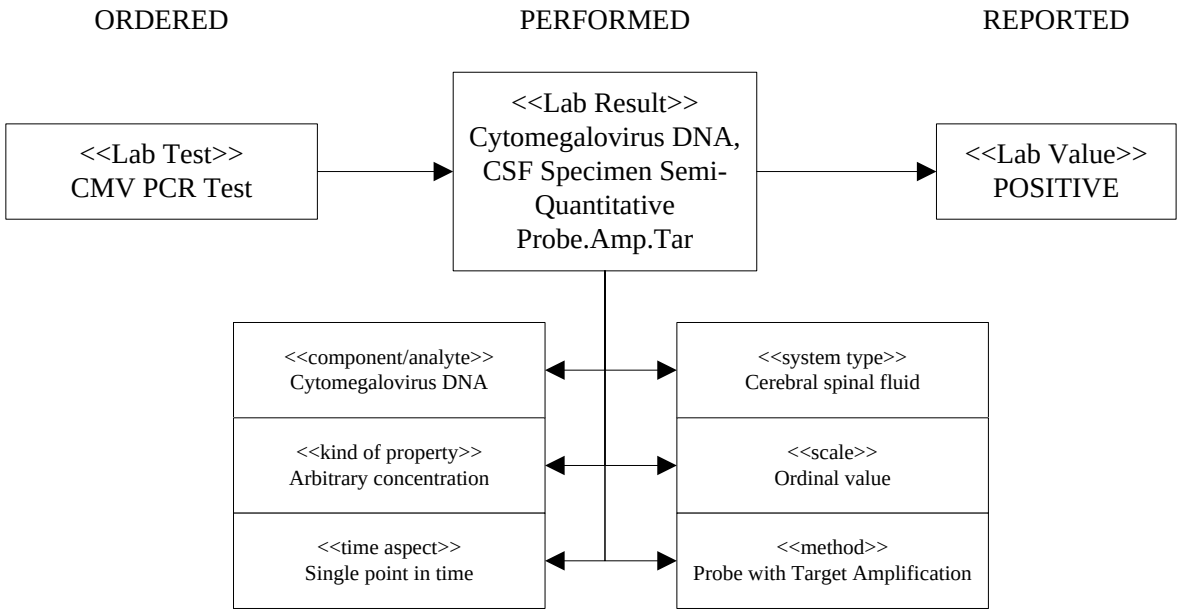


Figure 14: Example Lab Result. A lab test is ordered, a specific lab result is performed, and a lab value is reported.

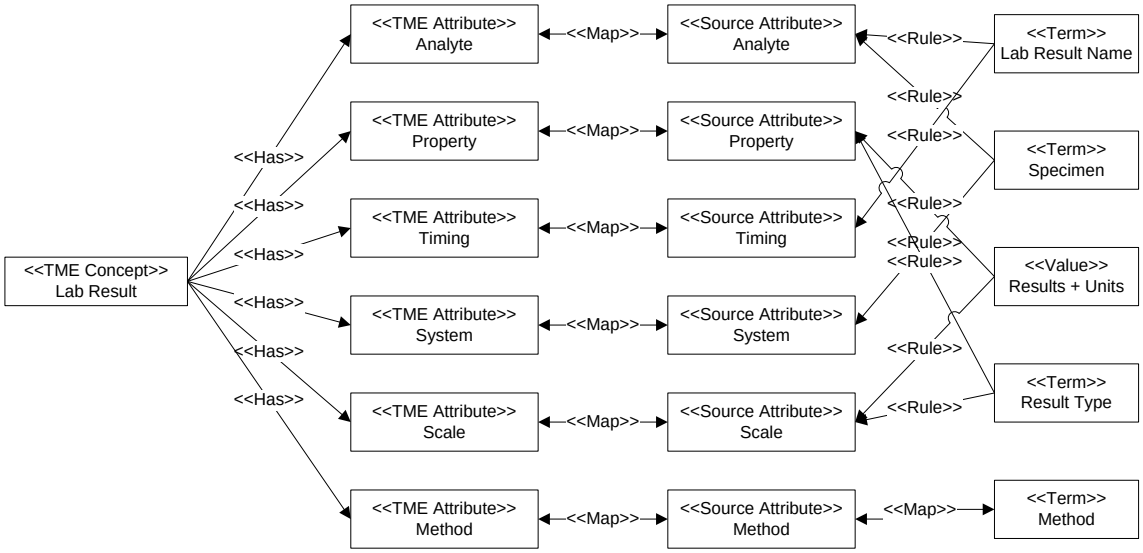


Figure 15: Lab LIT Data Model Mapping. Code attributes from a source lab LIT are on the far right of this diagram. Rules are used to map the source code system code attributes to the TME data model in the sources namespace. The rules sometimes involve simple parsing and in other cases involve inferring values. The source lab LIT is then mapped to the federated terminology by linking source attributes to TME attributes.

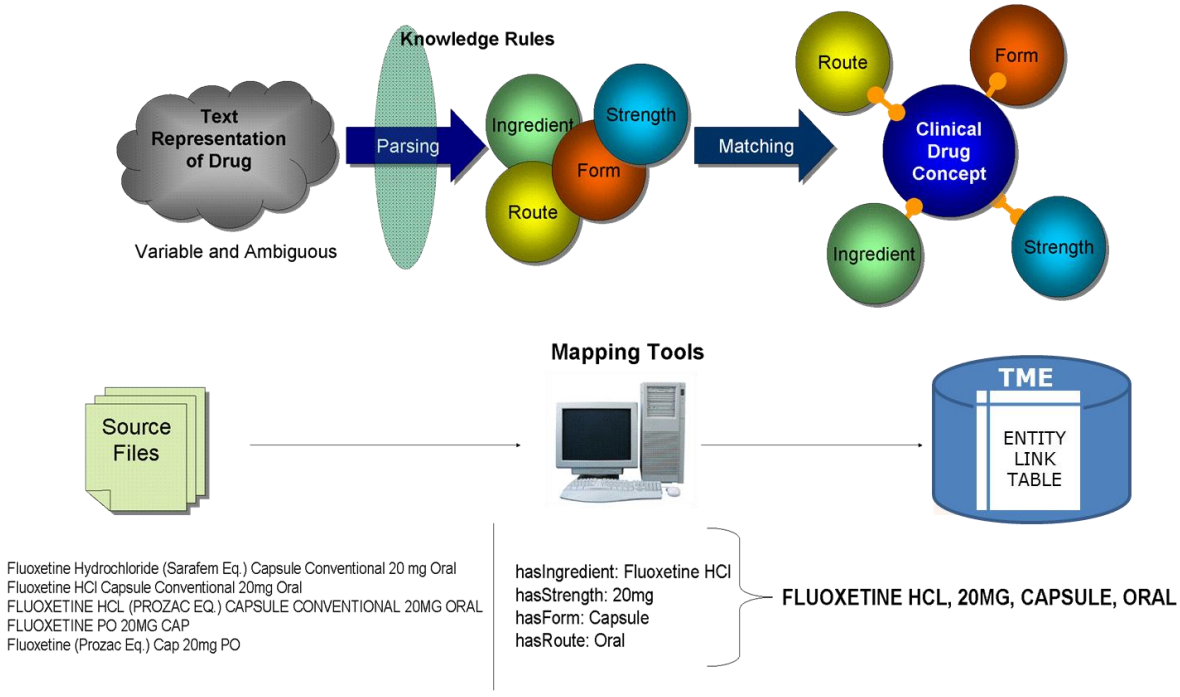


Figure 16: Semantic Matching for Drugs[29,78]

Table 13: Mapping Actions. List of final actions to be taken after SME review.

Mapping Action	Description
Ignore	Take no action with source code/term.
External question	SME has question(s) pending with source organization.
Internal question	SME has question(s) pending with another SME.
Create new TME concept	Build new concept in TFT and create mapping link to source code/term.
Map to existing TME concept	Build a map link between source code/term and existing concept in the TFT.
Remove map link for source code	Map link is determined to be no longer valid (either because refinement in mapping purpose definition, semantic shift/drift in source, or because it was discovered as an error) and is deleted.
Remap Source Code	Existing map link is deprecated and new map link is created to existing TME concept.

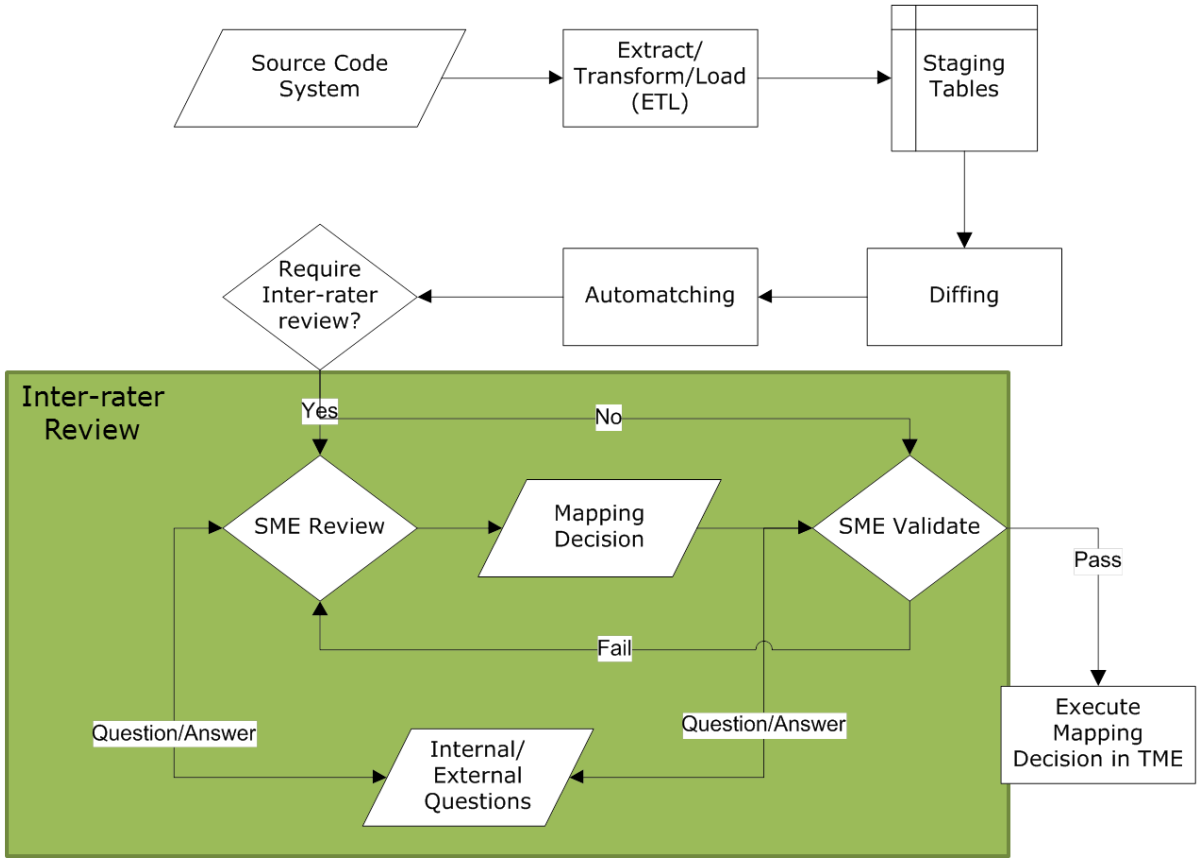


Figure 17: Summary of TME Concept Mapping Workflow.

11. VALIDATION OF DATABASE DESIGN AND INTEROPERABILITY REQUIREMENTS FOR THE TME

The TME was evaluated to determine if it met database design and interoperability requirements. The primary methodology for validation of the database design was source transparency and an assessment of how well the environment met the functional and design requirements specified in Section 8. The methodology for testing the interoperability requirements involved determining support for all HL7 Common Terminology Services version 1.2 (CTS v1.2) methods.

11.1. Database Design Validation

The high-level functional and design requirements from Section 8.2 are restated here in italics followed by a discussion of how the TME addresses each requirement.

11.1.1. Faithful data representation. *Source data must be represented in the TME with high fidelity (no change in the original meaning).*

The methodology employed to confirm faithful data-representation was source transparency. Confirming source transparency involved taking multiple versions of three source code systems (LOINC a clinical terminology with approximately 65,000 concepts, ICD-9-CM a classification system with approximately 14,000 concepts, and the 3M HDD a concept-based reference terminology with approximately 1.6 million concepts) with

varying levels of complexity and integrating them into the TME through the ETL process described in Section 10. Each version of the code system was then run through a "reverse" ETL to extract the code system in its original or native format. The extraction was then compared to the original version of the content to determine if the TME ETL process resulted in any loss or alteration of information in the source. Source system concepts, codes, terms, relationships, and attributes were all included in the evaluation. Testing was done against 3 successive versions of each code system and in all cases, the comparison indicated no changes or information loss. All source system concepts, codes, terms, relationships, and attributes were present and unaltered in the content extracted from the TME.

Based on these tests, it was determined that the TME met the requirement for source transparency. It is important to note that the source transparency test was performed using the respective code system namespaces and not the TFT. The federated terminology does not retain all source code system attributes and does not support source transparency.

11.1.2. Single schema. *The TME must store, access, link, and augment multiple disparate code systems in a single schema/data model.*

The TME houses the TFT along with all other source code systems in a single schema. SNOMED CT, LOINC, RxNORM, and 3M HDD have all been successfully transformed to the TME data model. The database addresses the storage requirement stated above. To address the set of integration requirements (access, linking, and ability to augment) a single schema and an overarching data model, the TFT was incorporated

into the TME as part of the hybrid integration model first described in Section 7.4. The integration requirement validation will be further explored in Section 11.2.

The TME met the requirement for a single schema. The TME schema includes the staging environment and vocabulary server and is detailed in the Appendix.

11.1.3. Support for multiple data/information models**.** *Data representation in the TME must be expressive enough to transform multiple source terminology models and support medical information models of various systems.*

The TME is capable of storing and representing source code system content at an equivalent level of granularity to the source. This was also validated through the source transparency testing. The code system code attributes can be captured either as representations (RSFORM table Section 9.2.2), properties (PROPERTY table Section 9.2.6), or attributes (ENTITY_ATTRIBUTE table Section 9.2.5). Defining attributes of entities are discrete and explicitly represented in the TME. As such, the TME design is capable of supporting multiple data/information models and the TME is at least as good as other external source code systems in its ability to represent knowledge. However, the TME is not meant to manage direct links between information model data elements or observations and terminology. This type of content can be managed in a TME namespace, but it cannot be easily generalized to other information models

Essentially this requirement becomes a test of the generalizability of TME. Since it is not possible to account for all possible implementations, the generalizability of the TME is evaluated based on its support for CTS. The integration validation described in Section 11.2 details TME support for CTS.

11.1.4. Partitioning. *The TME must have the ability to subset data for special purposes or because of special characteristics of the content.*

The TME provides five options for partition data:

1. VIEM_XML: is like a filter that allows for multiple consistent views of the same underlying data. It can be assigned to any component and used to subset/partition entities and their representations, relationships, attributes, and properties.
2. ENTITY_RELATIONSHIP: makes it possible to subset entities by defining a subset and associating member entities with the subset entity. This can be referred to as a concept-based value set.
3. RSFORM_LINK: makes it possible to subset terms by defining a subset and associating member entity terms with the subset entity. This can be referred to as a term-based value set.
4. RSFORM_CONTEXT: makes it possible to subset the types of surface forms associated with an entity.
5. SOURCE_TCID: creates namespaces in the TME for source code systems and can be assigned to any component.

11.1.5. Version control. *The TME must be able to provide version information and retrospectively represent multiple versions of the same content.*

The TME provides extensive version control of all components, meaning entities and their relationships, representations, attributes, and properties are all versioned and archived in revision tables. Version information was used in the test for source transparency.

11.2. Interoperability Requirement Validation Using CTS

The following plan was implemented for testing the TME interoperability requirement using CTS:

1. Assessment of all CTS v1.2 functions to determine which have unique "intent." Unique intent was determined by evaluating method signatures and looking at inputs and outputs.
2. Categorize unique-intent functions as being more content- or design-dependent. In other words, is the function supported by the current design if specific content is present?
3. Author database queries (SQL) against TME designed to return CTS function specified output(s) with defined input parameter(s). Examples of these queries are provided in Sections 11.2.2-11.2.6.
4. Compare TME query results with CTS output description and indicate one of the following:
   - **Fully Supported**: TME SQL output matches CTS function output description
   - **Partially Supported**: Meets the intent of the service; supported by the TME database design, but output is null or incomplete because all values are not instantiated in TME
   - **Not Supported**: TME design and content cannot meet the intent of the CTS function and supply the described output; cannot construct a query against TME
5. Interrater validation of TME Support assessment

- Reviewed by other informaticists and engineers to confirm support assessment.

11.2.1. Assessment and categorization of CTS v1.2 functions. Steps 1 and 2 of the interoperability requirements plan produced the following results:

1. Assessment of all CTS v1.2 functions to determine which have unique "intent"
   - 57 total functions (same functions appear in multiple classes)
   - 39 functions with unique definitions or input/output parameters
   - 35 functions with unique "intent," intent being what the function is attempting to do (not considering overloaded/convenience functions)
2. Categorize 35 unique-intent functions as being more content- or design-dependent
   - 20 functions design-dependent
   - 15 functions content-dependent

The TME supports standard functionality as described by HL7 Common Terminology Services 1 compliance minimum (CTS 2 when published[82]). There are multiple correct ways to address each of the CTS methods in the TME. The following sections illustrate one method for executing each of the functions against the TME, but in many cases, it is not the only way to support the function. Validation and peer review were used to assess the TME's capability. All CTS methods listed in Appendix C are supported by TME design or through content. Some examples are described in further detail in Sections 11.2.2-11.2.6.

11.2.2. CTS validate code. The CTS validateCode method is used to determine whether the supplied coded attribute from a code set is valid in the specified vocabulary domain and application context. There are two CTS named methods for code validation: validateCodeByCodedValue (uses the source code system code) and validateCodeByConceptDescriptor (uses the source code system descriptor or term).[65] These methods can be used to evaluate HL7 message content to determine if the codes in the message are valid. The input for the method is a domain (e.g., "Units of Measure") and a code (e.g., SNOMED CT code = "258770004") or a descriptor (e.g., SNOMED CT Description = "Liter").

- INPUT: Vocabulary Domain ID, Code/Code Descriptor to be validated, Application Context (realm)
- OUTPUT: VALID/INVALID, list of warnings and/or errors (see Figure 18)

The method can be applied against the TME in two ways. It can be used to check whether a code is valid in a source code system in the TME or it can be used to check if a code is valid and present in the TFT. Queries against the source code system namespace use source attributes, properties, and relationships. Queries against the TFT use TFT attributes, properties, and relationships.

First it is necessary to obtain a TCID for the "Units of Measure" domain. The TCID may be known, but it can also be determined by executing getSupportedVocabularyDomains CTS function. This function will return a TCID that has a representation(s) that match the supplied string. Query RSFORM.REPRESENTATION field using the string "Unit" and return the corresponding TCID in the SNOMED CT ($SNOUnits^{TCID}$) or TFT ($TFTUnits^{TCID}$) namespace.

SQL Query:

Select RSFORM.TCID from RSFORM where
RSFORM.UP_REPRESENTATION is like “UNITS” and
SOURCE_TCID = SNOMED CT^{TCID};
or
Select RSFORM.TCID from RSFORM where
RSFORM.UP_REPRESENTATION is like “UNITS” and
SOURCE_TCID = TFT^{TCID};

RSFORM.TCID = SNOUnits^{TCID} or TFTUnits^{TCID} in subsequent queries. This same type of query can be used to obtain TCIDs for strings for other entity structural properties, attributes, and relationships. The query can also be constrained to the domain of “Code System Domain” so that all the values returned will be domains/classes and not other kinds of entities.

Next, execute the validateCode CTS method. Query RSFORM.REPRESENTATION field with string “258770004” (for validateCodeByCodedValue) or “Liter” (for validateCodeByConceptDescriptor) in the SNOMED CT namespace to determine if the code is present, has an RSFORM.STATUS_TCID that equals “Active,” and is a surface form for an entity that is a child of SNOUnits^{TCID} or TFTUnits^{TCID} (see Figure 19).

SQL Query:

Select RSORM.TCID from RSFORM, ENTITY_RELATIONSHIP where
RSFORM.REPRESENTATION = “258770004” and
RSFORM.CONTEXT_TCID = SNOCodeContext^{TCID} and
RSFORM.STATUS_TCID = SNOActive^{TCID} and

- *{Look for the SNOMED CT code “258770004” in the SNOMED CT context and confirm that it is active.}*

RSFORM.TCID = ENTITY_RELATIONSHIP.TCID_1 and
ENTITY_RELATIONSHIP.RELATIONSHIP_TCID = SNOIsA^{TCID} and
ENTITY_RELATIONSHIP.TCID_2 = SNOUnitsTCID;

- *{Join to the ENTITY_RELATIONSHIP table and confirm that the entity with SNOMED CT code “258770004” is a unit of measure. This check is not necessary for SNOMED CT since SNOMED CT codes are unique*

across all domains in the code system. But for other source code systems, uniqueness of code is not guaranteed across domains.}

Result: TME is capable of validating source code systems codes and descriptors.

11.2.3. CTS validate a translation. Validate Translation (validateTranslation): this method is used to determine whether the transformation portion of the coded attribute is valid in the specified vocabulary domain and application context.[65] This method can be used to evaluate a provided transformation between a source and target code system against mappings in the TFT. The input for the method is a domain (e.g., "Units of Measure"), a source and target code system codes or descriptors (e.g., SNOMED CT code = "258770004" is equivalent to UMLS CUI = "C0475211").

- INPUT: Vocabulary Domain ID, Code/Descriptor from source and target code systems for the transformations to be validated, Application Context (realm)
- OUTPUT: VALID/INVALID, list of warnings and/or errors (see Figure 20)

Mappings are held in the TME ENTITY_LINK Table.

SQL Query:

Select RSFORM.TCID from RSFORM, ENTITY_RELATIONSHIP where
RSFORM.REPRESENTATION = "258770004" and
RSFORM.CONTEXT_TCID = SNOCodeContext^{TCID} and
RSFORM.STATUS_TCID = SNOActive^{TCID};

- *{Look for the SNOMED CT code "258770004" in the SNOMED CT context and confirm that it is active.}*
- RSFORM.TCID = SNOLiter^{TCID}

Select RSFORM.TCID from RSFORM, ENTITY_RELATIONSHIP where
RSFORM.REPRESENTATION = "C0475211" and
RSFORM.CONTEXT_TCID = UMLSCUIContext^{TCID} and
RSFORM.STATUS_TCID = UMLSActive^{TCID};

- *{Look for the UMLS CUI code "*C0475211*" in the UMLS CUI context and confirm that it is active.}*
- RSFORM.TCID = UMLSLiter^{TCID}

Once TCIDs have been obtained for both the SNOMED and UMLS codes, the ENTITY_LINK table must be queried to see if any link/transformation exists between the

two TCIDs. TME can validate if any link in either direction exists between the two TCIDs:

Select * from ENTITY_LINK where
ENTITY_LINK.TCID_1 = SNOLiter^{TCID} and
ENTITY_LINK.TCID_2 = UMLSLiter^{TCID} or
ENTITY_LINK.TCID_1 = UMLSLiter^{TCID} and
ENTITY_LINK.TCID_2 = SNOLiter^{TCID};

TME can also validate transformations using only a specific link type (ENTITY_LINK_TCID) or link direction:

Select * from ENTITY_LINK where
ENTITY_LINK.TCID_1 = SNOLiter^{TCID} and
ENTITY_LINK.ENTITY_LINK_TCID = TFTClinicalEquivalent^{TCID} and
ENTITY_LINK.TCID_2 = UMLSLiter^{TCID};

Result: TME is capable of validating transformations by evaluating links between TME code identifiers (TCIDs) in the ENTITY_LINK Table. Not only can TME confirm if a link exists, it can also provide the direction and type of linkage.

11.2.4. CTS translate a code. Translate Code (translateCode): this method is used to transform a supplied coded attribute (code/descriptor) in a specified source code system into a target form that is valid in the target application context.[65] This method can be used by an interface engine to provide mediation services, transforming local codes to standards or one standard to another. The input for the method is a domain (e.g., "Units of Measure"), a source code system code or descriptor (e.g., SNOMED CT code = "258770004"), and a target code system and context (e.g., UMLS CUI).

- INPUT: Vocabulary Domain ID, source Code/Descriptor, and target Code systems and Context
- OUTPUT: Target Coded Attribute (see Figure 21)

Similar to the validateTranslation method the primary TME table for this method is the ENTITY_LINK table which holds mappings. The ENTITY table is included to check

entity status and the RSFORM table is joined to associate terms/descriptors with the codes.

SQL Query:

Select RSFORM.TCID from RSFORM, ENTITY_RELATIONSHIP where
RSFORM.REPRESENTATION = "258770004" and
RSFORM.CONTEXT_TCID = SNOCodeContext^{TCID} and
RSFORM.STATUS_TCID = SNOActive^{TCID};

- *{Look for the SNOMED CT code "258770004" in the SNOMED CT context and confirm that it is active.}*
- RSFORM.TCID = SNOLiter^{TCID}

Once a TCID have been obtained for the SNOMED CT code, the ENTITY_LINK table is queried to transform the code to a UMLS CUI (if such a mapping exists in the TME):

Select RSFORM.REPRESENTATION from RSFORM, ENTITY where
RSFORM.TCID in
(Select TCID_1 from ENTITY_LINK where
ENTITY_LINK.TCID_2 = SNOLiter^{TCID})
or
(Select TCID_2 from ENTITY_LINK where
ENTITY_LINK.TCID_1 = SNOLiter^{TCID}) and
RSFORM.CONTEXT_TCID = UMLSCUIContext^{TCID} and
RSFORM.TCID = ENTITY.TCID
ENTITY.STATUS_TCID = UMLSActive^{TCID};

Transformations can be made more specific by specifying link type (ENTITY_LINK_TCID) and/or link direction.

Result: TME is capable of transforming a source coded attribute to a target coded attribute. It is also possible to traverse relationships in the ENTITY_LINK table or compare attributes in the ENTITY_ATTRIBUTE table and provide the "nearest" target coded attribute.

11.2.5. CTS fill in code details. Fill in Details (fillInDetails): this method is used to supply additional details for a coded attribute, including all code system names,

versions, and display names.[65] The input for the method is a source code system code or descriptor (e.g., SNOMED CT code = "258770004"). The output is additional optional information the TME has regarding the code and associated content. The minimum expected values are code attributes such as displays or descriptions and version information, but TME is capable of supplying a great deal of additional information.

- INPUT: Source Code/Descriptor, Language
- OUTPUT: Coded Attribute Details (see Figure 22)

SQL Query:

Display all descriptions/codes, source, and version information in a namespace:

Select RSFORM.REPRESENTATION, RSFORM.SOURCE_TCID,
RSFORM.SOURCE_VER_TCID from RSFORM where
RSFORM.TCID = (Select RSFORM.TCID from RSFORM where
RSFORM.REPRESENTATION = "258770004" and
RSFORM.CONTEXT_TCID = SNOCodeContext^{TCID} and
RSFORM.STATUS_TCID = SNOActive^{TCID}) and
RSFORM.LANGUAGE_TCID = English^{TCID};

Result: TME is capable providing additional details for a coded attribute, including: descriptions, codes, version information, defining attribute information, properties, related coded attributes, and source code system information.

11.2.6. CTS implies. Implies (Implies): this function is used to determine whether the parent coded attribute implies (subsumes) the child. This involves:

1. Determining if there is any type of relationship between the two coded attributes.
2. If there are relationships, do any of them imply subsumption?

Relationship types are concepts within the TME. They are used to organize, group, and relate content for various purposes. The TME segregates relationships into two classes:

1. Hierarchical relationship: also referred to as vertical relationships. These are relationship types where a parent coded attribute implies the child (e.g., ENTITY_RELATION.RELATIONSHIP_TCID = “is a”). These relationship types are further broken down into subtype:supertype and supertype:subtype relationship types.
2. Nonhierarchical relationship: also referred to as horizontal relationships. These are relationship types where the first coded attribute does not imply the second. (e.g., ENTITY_RELATION.RELATIONSHIP_TCID = “has ingredient”).

The input for the method is a parent code or descriptor (e.g., SNOMED CT code = “282115005,” “SI-derived unit of volume”) and a child code or descriptor (e.g., SNOMED CT code = “258770004,” “Liter”). The output is “TRUE” or “FALSE.

- INPUT: Parent Coded Attribute, Child Coded Attribute
- OUTPUT: True/False (see Figure 23)

SQL Query:

1. Determining if there is any type of relationship between the two coded attributes. For this example, it will be assumed that it is unknown whether this is expected to be a subtype:supertype (child to parent) or supertype:subtype (parent to child) relationship type.

```
Select RELATIONSHIP_TCID from ENTITY_RELATION where
    RELATIONSHIP_TCID in
        (Select RELATIONSHIP_TCID from ENTITY_RELATION where
        ENTITY_RELATION.TCID_1 = SNOLiter^TCID and
        ENTITY_RELATION.TCID_2 = SNOSIVolUnit^TCID)
        or
        (Select RELATIONSHIP_TCID from ENTITY_RELATION where
        ENTITY_RELATION.TCID_1 = SNOSIVolUnit^TCID and
        ENTITY_RELATION.TCID_2 = SNOLiter^TCID);
```

2. If there are relationships, do any of them imply subsumption?

```
Select * from ENTITY_RELATION where
        TCID_2 = TFTHierarchicalRelationType[TCID] and
        RELATIONSHIP_TCID = TFTIsA[TCID] and
        TCID_1 in
                (Select RELATIONSHIP_TCID from ENTITY_RELATION where
                RELATIONSHIP_TCID in
                        (Select RELATIONSHIP_TCID from ENTITY_RELATION
                        where
                        ENTITY_RELATION.TCID_1 = SNOLiter[TCID] and
                        ENTITY_RELATION.TCID_2 = SNOSIVolUnit[TCID])
                        or
                        (Select RELATIONSHIP_TCID from ENTITY_RELATION
                        where
                        ENTITY_RELATION.TCID_1 = SNOSIVolUnit[TCID] and
                        ENTITY_RELATION.TCID_2 = SNOLiter[TCID]));
```

Result: TME is capable of indicating if the relationship between two coded attributes is hierarchical.

11.3. CTS Validation Results and Discussion

It was determined that all 57 CTS v1.2 functions can be fully or partially supported by the TME infrastructure (see Table 14). Of the total functions, 51 (89%) are fully supported and 6 (11%) are partial supported. When these numbers are adjusted for functions that are duplicated across functional areas or have identical inputs and outputs (method signatures or intent), the percent coverage for partial and full support remain the same (89% or 31/35 full support and 11% or 4/35 partial support). It was determined that support for the function was partial either because the intent of the function could be met, but the method signature would be different than what is defined in CTS, or a partial or "null" result was returned in the output.

The 4 functions that are partially supported are:

1. getSupportedMaps: The TME links concepts in disparate code systems through the ENTITY_LINK table. Those links are “flattened” in the TFT for runtime deployment of content. Comprehensive point-to-point mapping is not the specific objective of the TME.
2. getSupportedMatchAlgorithms: Query ENTITY_RELATION for match algorithm entity associated with the TFT "Service" entity. Although TME design supports this service, the content has not been instantiated.
3. lookupValueSet: Query ENTITY_ATTRIBUTE, PROPERTY, and ENTITY_RELATION tables with value set entity TCID to describe how the value set is constructed in the TME. Note: indicated partial support because the current implementation does not provide a value set "definition" (rules for how to construct/populate the intentional value set), it returns enumerated values that demonstrate how the value set has been instantiated.
4. lookupVocabularyDomain: TME can support this function from a design perspective, but supporting content that maps RIM attributes dependencies has not been instantiated in the TFT.

In all partially supported cases, the content was either intentionally not instantiated or the work to populate the content was incomplete – nothing about TME design precluded full support of all CTS v1.2 functions. Based on these results, it was concluded that the TME interoperability requirement was met.

RSFORM	
RSFORM_ID	**DOUBLE**
REPRESENTATION	**TEXT(4000)**
UP_REPRESENTATION	**TEXT(4000)**
PREFERRED_REP	**TEXT(1)**
CONTEXT_TCID	**DOUBLE**
CASE_SENSITIVE	**TEXT(1)**
LANGUAGE_TCID	**LONG**
USAGE_SCORE	DOUBLE
STATUS_TCID	**DOUBLE**
SOURCE_TCID	**DOUBLE**
SOURCE_VER_TCID	DOUBLE
ADDED_DATE	**DATETIME**
REVISED_DATE	DATETIME
EFFECTIVE_DATE	DATETIME
EXPIRATION_DATE	DATETIME
VIEW_XML	TEXT(4000)
COMMENT_XML	TEXT(4000)
REV_COMMENT_XML	TEXT(4000)
PREVIOUS_RID	LONG
NEXT_RID	LONG
RID	**DOUBLE**
TCID	**DOUBLE**

Figure 18: TME Return a Code for a Specified Domain. Fields used in the RSFORM table to supply a TCID for the SNOMED CT Domain of "Units." Yellow highlighted fields are inputs; Green highlighted fields are output.

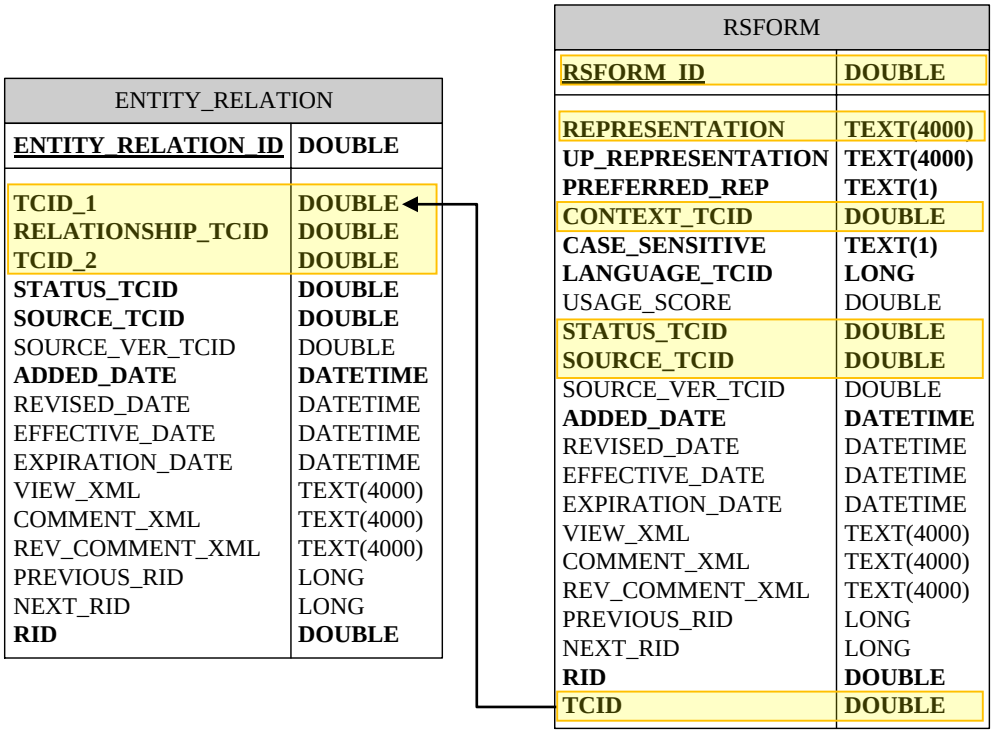


Figure 19: TME Validate a Code is Present and Active in a Specified Code System. Fields used in the RSFORM and ENTITY_RELATION tables to determine if a given code or code descriptor is present (code or description exists), active (STATUS_TCID = "Active"), and in the specified code system (related to code system namespace in ENTITY_RELATION table).

ENTITY_LINK	
ENTITY_LINK_ID	**DOUBLE**
TCID_1	**DOUBLE**
ENTITY_LINK_TCID	**DOUBLE**
TCID_2	**DOUBLE**
MAP_SET_TCID	LONG
MAP_SCORE	**DOUBLE**
RULE_XML	TEXT(4000)
STATUS_TCID	**DOUBLE**
SOURCE_TCID	**DOUBLE**
SOURCE_VER_TCID	DOUBLE
ADDED_DATE	**DATETIME**
REVISED_DATE	DATETIME
EFFECTIVE_DATE	DATETIME
EXPIRATION_DATE	DATETIME
VIEW_XML	TEXT(4000)
COMMENT_XML	TEXT(4000)
REV_COMMENT_XML	TEXT(4000)
PREVIOUS_RID	LONG
NEXT_RID	LONG
RID	**DOUBLE**

Figure 20: TME Validate Transformation. Confirm that a mapping link (ENTITY_LINK_TCID) exists between source code TCID and target code TCID**.**

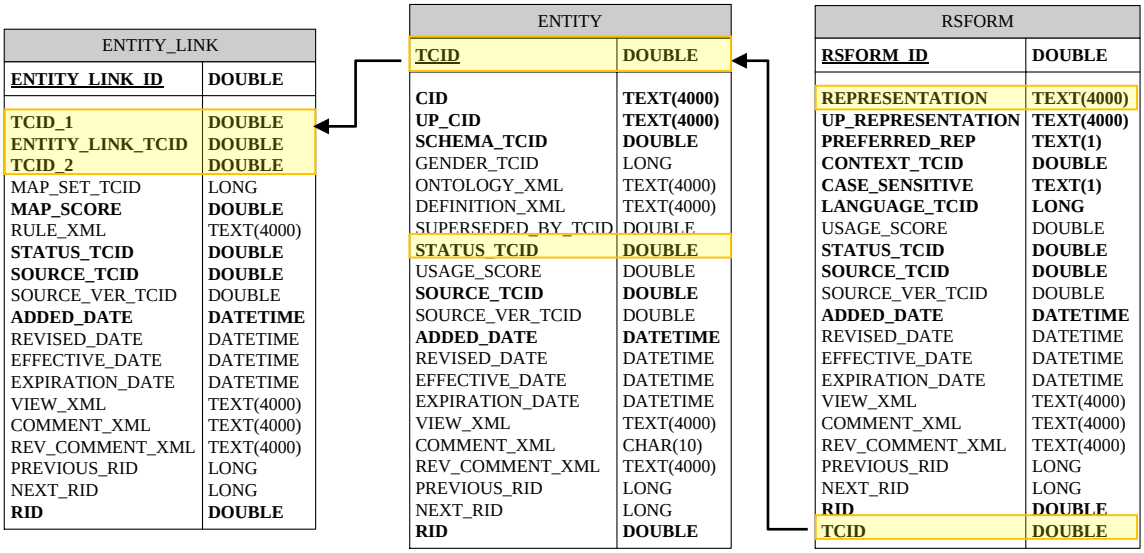


Figure 21: TME Transform Source Code to Mapped Code in Specified Target Code System.

RSFORM	
RSFORM_ID	**DOUBLE**
REPRESENTATION	**TEXT(4000)**
UP_REPRESENTATION	**TEXT(4000)**
PREFERRED_REP	**TEXT(1)**
CONTEXT_TCID	**DOUBLE**
CASE_SENSITIVE	**TEXT(1)**
LANGUAGE_TCID	**LONG**
USAGE_SCORE	DOUBLE
STATUS_TCID	**DOUBLE**
SOURCE_TCID	**DOUBLE**
SOURCE_VER_TCID	DOUBLE
ADDED_DATE	**DATETIME**
REVISED_DATE	DATETIME
EFFECTIVE_DATE	DATETIME
EXPIRATION_DATE	DATETIME
VIEW_XML	TEXT(4000)
COMMENT_XML	TEXT(4000)
REV_COMMENT_XML	TEXT(4000)
PREVIOUS_RID	LONG
NEXT_RID	LONG
RID	**DOUBLE**
TCID	**DOUBLE**

Figure 22: TME Display all Descriptions, Codes, and Version Information in a Given Namespace. RSFORM fields that provide code and descriptor information and are used to link to version and entity tables to return source and version information.

ENTITY_RELATION	
ENTITY_RELATION_ID	**DOUBLE**
TCID_1	**DOUBLE**
RELATIONSHIP_TCID	**DOUBLE**
TCID_2	**DOUBLE**
STATUS_TCID	**DOUBLE**
SOURCE_TCID	**DOUBLE**
SOURCE_VER_TCID	DOUBLE
ADDED_DATE	**DATETIME**
REVISED_DATE	DATETIME
EFFECTIVE_DATE	DATETIME
EXPIRATION_DATE	DATETIME
VIEW_XML	TEXT(4000)
COMMENT_XML	TEXT(4000)
REV_COMMENT_XML	TEXT(4000)
PREVIOUS_RID	LONG
NEXT_RID	LONG
RID	**DOUBLE**

Figure 23: TME How Are Two Entities Related. ENTITY_RELATION fields used to determine if relationship between two entities implies subsumption.

Table 14: CTS Validation Results. Count and percent of total number of CTS functions supported by TME. Result validated indicates that multiple SMEs agree with the assessment.

Description of Category Established for TME Validation	Count (% of Total)	# Fully Supported (% of Count)	# Partially Supported* (% of Count)	# Not Supported (% of Count)	Result Validated
All CTS v1.2 Functions (Across all classes)	57 (100%)	51 (89%)	6 (11%)	0 (0%)	Yes
Functions with unique definitions or input/output	39 (68%)	34 (87%)	5 (13%)	0 (0%)	Yes
Unique functions based on "intent"	35 (61%)	31 (89%)	4 (11%)	0 (0%)	Yes

12. DISCUSSION

This section takes a deeper look at justification for the project approach and characteristics that differentiate it from other systems attempting to do similar things by:

- further reasoning for use of the TFT and the hybrid model as opposed to using standards directly
- characteristics that distinguish TME from other terminology management systems
- significance of this project to the field of biomedical informatics
- potential future work

12.1. Why Not Just Use Standards Directly?

An alternative approach (mentioned in Section 7.1) to the TME is to use standard terminologies directly. Using standards directly entails using the identifiers of standard vocabularies to encode patient data, instead of inserting an abstraction layer (redirecting) through the TME and assigning TCIDs. While standardization and interoperability would be achieved with this approach, other challenges remain.

12.1.1. Dealing with semantic change in external code systems. As discussed in Section 5.1.3, change in the meaning of an encoded concept results in semantic shift (sudden change) or drift (slight alteration over time). Patient data encoded with identifiers that have undergone shift or drift cannot be correctly interpreted. If a code

from a standard terminology is used to encode data and the meaning of the code changes over time, historical patient data will be interpreted incorrectly in a longitudinal patient record. This can happen because of:

- Code reuse: a common problem for many standard code systems, e.g., ICD-9-CM, NDC. This is where a code is deleted and then reintroduced with a different meaning.
- "Adjustment or refinement": these are "small" changes in the display associated with content that can accumulate over time and in certain contexts could be considered a change in meaning.[29]

The federated terminology in the TME (TFT) provides a buffer between the patient data and flux in external terminologies. Concepts created in the TFT follow the principle of concept permanence[26] and never change their meanings so that patient data encoded with TCIDs will never be misinterpreted.

Since external standard codes are associated with TFT TCIDs through the ENTITY_LINK table, as external codes are reused, the mapping in the TME will change accordingly. The ENTITY_LINK table also facilitates the task of remapping ENTITY_LINK_TCID = TFTIsNarrowerThan or TFTIsBroaderThan mappings with successive releases of the source terminology. Mappings that are not designated as equivalent can be reevaluated to determine if any new content in the source would be a more accurate or appropriate map than the current link.

This approach of using a third identifier (TCID) and associating detailed version and context information to mappings in the ENTITY_LINK table does two things to address the issue of semantic shift and drift in source terminologies:

1. Information is never discarded. Source content is encoded with native granularity and compositional structure. The TFT follows the principle of concept permanence and mappings in the ENTITY_LINK table are reevaluated when there is semantic shift or drift in a source terminology.
2. The ENITY_LINK table is used to capture details with regard to the conditions and specificity of the mappings (e.g., ENTITY_LINK.ENTITY_LINK_TCID establishes the specificity of the mapping, ENTITY_LINK.RULE_XML defines conditions for which the mapping is valid, ENTITY_LINK.MAP_SET_ID is used as an index for composition and decomposition). Mappings between source and target can be more easily reevaluated because each link is documented in such a way that categories of mappings can be identified and evaluated.

12.1.2. Dealing with the deletion of standard codes. Standard code systems may retire/deprecate or delete codes. If patient data are stored using the removed code, they will no longer be interpretable. The TME addresses this challenge in two ways: 1) through the revision tables which track changes to the sources and 2) through assigning a stable concept identifier in the TFT namespace (TCID).

The namespace for each source terminology in the TME is kept current with the source. If the source deletes a code or code attribute, it is also deleted in the source namespace. However, the content change will result in entries in the mirror revision tables. So the type of change and when it happens is tracked in the TME. While this provides a means of tracking the changes that occur in source terminologies, it is not a practical way to address the impact these changes may have on the ability to interpret

patient data over-time. Providing a stable third identifier to address this issue is one of the primary purposes of the TFT.

In the TFT, a TCID, once created, is never deleted and its semantics are not altered. Using the TFT TCID to encode and store patient data and not a standard code protects patient data from code deletion.

12.1.3. Lack of comprehensive standard codes. A standard vocabulary may not provide all the codes that correspond to the entire set of data in current use. The set of data that falls outside the coverage of one or multiple standards must be addressed. Some standards such as SNOMED CT provide an "extension" mechanism. SNOMED CT allows third parties to request a local extension namespace that can be used to extend SNOMED CT code coverage to content that is not currently and may never be in the standard.[83]

Local extensions are codes used to encode data when the appropriate, equivalent code is not found in the standard terminology. Local extensions provide concepts needed by a third party for different granularity or compositional structure. This local content is critical for efficient workflow and data capture. However, it is a major contributing factor for why many health care organizations have data interoperability issues.[29]

Both the local extension namespace approach offered by standards and the federated terminology in the TME address the issue of comprehensiveness, but when a standard code is used as a system's internal identifier, the consequence is that the system has no insulation from the standards ontology or "world-view." Issues such as semantic change and nuance in the way content is defined cannot be addressed with this approach. The TFT allows systems to create and maintain their own internal "world-view" and translate

to the appropriate standard for the purpose of data exchange. This approach allows for standards compliance for the sake of interoperability without being constrained to the precise manner in which the standards have modeled and defined content.

12.1.4. Historical patient data. If standards are used directly and no mapping is performed to link legacy codes to the standards, backward compatibility is lost. The historical patient data are important for continuity of care, quality of care delivery, population health management, and outcomes research.[29] Failing to create a mapping would result in the historical data being incompatible with newly collected data and essentially lost to computable clinical or administrative use.

Creating mappings in the TME ENTITY_LINK table maintains backward compatibility and addresses this issue with historical patient data. At any point in time, the association between legacy content and the TFT is kept in semantic alignment though the notion of concept permanence and because of the metadata in the ENTITY_LINK table that explicitly defines the nature of the mapping (see Section 9.2.4).

12.2. How the TME Is Different from other Vocabulary Servers

The TME is composed of a vocabulary server (the set of tables and services that hold the source code systems and the TFT) and a set of terminology management utilities, with supporting applications, processes, documents, and tables (see Figure 24).

Vocabulary server capabilities can be divided into the following broad categories:

- Content Development: creating a terminology; focus is on authoring new content
- Content Integration: linking various source terminologies; focus is on mapping

- Integrated Content Maintenance: keeping multiple integrated source terminologies current and synchronized; focus is on identifying changes, mapping, and reevaluating existing maps
- Content Implementation: strategy for making content operational within a system; focus is on system content requirements—the ability to extract, transform, and load content
- Runtime deployment: supporting applications with terminology services in runtime; focus is on performance

Many of the vocabulary servers available today (listed in Section 3.4) are commercial and most specialize in either content development or runtime deployment. The TME focuses on content integration, maintenance, and implementation.

12.2.1. Content integration in the TME. The TME defines content and associates attributes through the ENTITY, RSFORM, ENTITY_ATTRIBUTE, and PROPERTY tables and establishes context through the ENTITY_RELATIONSHIP table (Section 9.2.3). The context and definition of content is used in the mapping process for content integration (Section 10.5). Additional code attributes, including ENTITY.GENDER_TCID, ENTITY.DEFINITION_XML, and *COMPONENT*.USAGE_SCORE, are used to identify and filter candidate matches in the mapping process. Theses mappings are defined and stored in the ENTITY_LINK table (Section 9.2.4).

The way TME integrates content in the TFT is a key differentiator. This single federated terminology acts as a concept-based "anchor" for the content managed in the TME. There are approaches that attempt to achieve content integration by

incorporating/managing only the subset of standard content that is used within their system through point-to-point maps (discussed in Section 7.2). Such systems only have the ability to interpret the subset of standard codes that map to their LIT. This approach is unidirectional in terms of interoperability. It works for sending data, but not for receiving it.

Consider the following example; the English language is a terminology. The subset of English language, which an individual understands, is his/her vocabulary. In practical terms, semantic interoperability using English as an example cannot require speakers to exchange a list of terms in their vocabulary prior to holding a dialogue and limit the conversation to only those terms that are on both speakers' lists. The more complete a speaker's understanding of the English language is, including terms they may not commonly use, the less information is lost in communication. Similarly, exchange of data using a standard terminology requires sending and receiving systems to "understand" the entire code system, even if their local terminology only maps to a subset of it.

The TME entity is not concept-based. In this way, external code systems can be loaded into their TME namespace without losing native metadata (e.g., properties, attributes, surface forms, etc.). Since the TME contains all the codes from sources that have been integrated along with the associated native metadata, no information beyond the code systems and version needs to be communicated in order to achieve semantic interoperability between systems. In other words, two systems can arrange to use a particular code system for exchanging data and as long as the sending system provides

the code system name and version, along with the code, the TFT can be referenced to interpret and transform the code without loss of information.

The TME loads all of the code attributes of source code systems into their respective namespaces and integrates those codes through mappings that are defined in the ENTITY_LINK table. It permits point-to-point mapping, but there is a canonical centralized mapping to the TFT. With this approach, runtime content generated by the TME can interpret all incoming standard codes and translate to an exact or approximate internal code.

12.2.2. Content maintenance in the TME. Content maintenance involves evaluating subsequent versions of a source terminology for integration in the TME. The key differences between content maintenance and initial content integration are:

1. Identifying where there was change and what type of change it was
2. Identifying what previous mappings must be reevaluated

This area of content maintenance is frequently overlooked. Many organizations and systems think of mapping as a one-time effort and ignore the fact that both source and target terminologies are likely to change over time. When systems do try to address these changes, they typically think of maintenance as addressing things that "error off" in an interface engine. This will happen if an interface engine attempts to translate a code and no mapping exists. These systems think of maintenance as making sure that there is a target code for every source code, but they do not evaluate changes in source or target that may have affected existing mappings.

The TME attempts to keep mappings synchronized without requiring a complete remapping with each subsequent version of a source and/or target terminology as the

source terminologies diverge. Change is tracked at the level of source code attributes in the TME. This means that for any change to a code attribute such as a display or relationship, there will be an entry in the corresponding revision table. The revision tables allow the TME to generate a "snapshot" of what source content looked like in the TME at any past point in time. This is because source code systems may make changes to any single code attribute without versioning the code itself.

Content maintenance in the TME relies heavily on the temporal tracking of entities and their corresponding metadata in the revision tables (Section 9.2 and TME ERD Revision tables described in Appendix A) and "Diffing" (Section 10.3) to identify changes. Once this is done, the source namespace can be updated. The next step is to evaluate mappings between the source namespace and other namespaces. Mapping maintenance involves creating new maps using the content integration steps described in Section 12.2.1 and establishing which existing mappings must be evaluated using the ENTITY_LINK table (Section 9.2.4).

The ENTITY_LINK.ENTITY_LINK_TCID is used to identify the mappings that need to be reevaluated. Instead of remapping all source and target entities, only the links where source or target changed and the ENTITY_LINK_TCID does not equal "TFTIsEquivalentTo" must be reevaluated.

12.2.3. TME content implementation. Content implementation is using clinical terminology in production systems. It involves support for runtime capabilities. Examples of support for these runtime functions are provided in Section 11.2. CTS was used for external validation of the capabilities of the TME (full evaluation is provided in Appendix C and results are summarized in Table 14). However, design requirements for

the TME were taken from the functional requirements described in Section 8.2. The TME design supports all CTS v1.2 requirements. The TME can support runtime capabilities through these services or through direct calls or content may be extracted from the TME and transformed to optimize runtime performance, since runtime implementations may have performance requirements for which the TME is not optimized. The current approach is to export TME content into indexes that are optimized to support particular runtime capabilities and specific system requirements.

The TME facilitates content export with *COMPONENT*.VIEW_XML (Section 9.2). *COMPONENT*.VIEW_XML allows for multiple consistent views of the same underlying content.[26] It is associated with every TME table and can be used to partition TME content for export. It is used to designate the boundaries of namespaces among concepts, designations, and attributes. In the current implementation, the VIEW_XML attribute is manually assigned and static. Future work will entail implementing a strategy for dynamic assignment to a particular view based on criteria.

12.2.4. Additional characteristics of TME content. The TME design has unique characteristics that support modeling content and content integration. Entity and representation links and entity attributes will be discussed as examples.

The TME Enitity Link table stores metadata about the association between two entities (see Table 8). These associations can be created for multiple purposes (e.g., map sets, value sets). The following are examples of unique capabilities enabled by entity links:

- Ability to establish context for mappings--add additional information about mappings between source and target code systems (see Figure 25)

- Ability to express bi- or unidirectional mappings
- Ability to indicate the specificity of the map (i.e., “isEquivalentTo,” “isBroaderThan,” “isNarrowerThan,” etc.)
- Ability to assign mappings for a particular purpose or for a particular user
- Ability to comment on mappings and assign status

Similarly to the TME Entity Link table, the Representation Link table stores metadata about the association between two representations (see Figure 26). These associations can be created for multiple purposes (e.g., lexical variant sets, value sets). The following are examples of unique capabilities enabled by representation links:

- Ability to make explicit links between two representations on the same entity or across entities
- Ability to link between code and display
- Ability to link lexical variants of terms
- Ability to create term-based value sets that link representations across multiple entities

The entity attributes table provides a way to formally define entities by associating attributes with specified data types (see Figure 27). A coded attribute data type specifies that the attribute is another entity. The following are examples of capabilities enabled by entity attributes:

- Ability to formally define entity through associated attributes
- Ability to compare attribute sets source to target for mapping
- Ability to express name-value pairs where values can be coded (TCID), number, string

12.3. Significance of Work

This project builds upon the work of many others who developed approaches and systems to standardize clinical terminology.[25] The unique contributions of this project are:

- the work to derive and describe a conceptual understanding of the scope and intricacies of the challenge
- the approach and solution derived from applying informatics principles, practical experience, and real-world requirements
- a working environment that meets the requirements for creating, maintaining, and distributing terminologies
- a system that address a set of key consideration (source transparency, overarching data model, and scalability) that to-date no single system fully supports

Although the challenge of successfully implementing clinical terminology is widely recognized, it is not well understood. This project explicitly enumerated many of the challenges in justification of an approach that might seem "over-engineered" without a more thorough understanding of the topic.

With the application of informatics principles to a more well-defined problem space, a more strategic rather than tactical solution was defined. It would be possible to engineer multiple discrete, tactical solutions to address each of the functional requirements and challenges defined in the project. However, doing so would have dramatically increased the complexity of the solution and the difficulty of maintaining it and the content within it.

The TME is capable of maintaining a federated terminology and simultaneously integrate code system namespaces. Name value pairs for a range of data types can be defined in the ENTITY_ATTRIBUTE and PROPERTY tables. Additional entity attributes and metadata are maintained within the system to integrate workflow and tooling and formalize the creation, integration, and maintenance of content.

The TME is a key data governance tool that:

- Structures: associates concepts in a meaningful way
- Normalizes: provides a single identifier for the multiple terms and codes associated with a concept in the TFT
- Standardizes: links local terminologies to standards through concept mapping

It does this by meeting the functional requirements of the various consumers and authors of terminology (Section 8.1). It provides this support through the entire life cycle of terminology within an organization addressing ETL, provenance, maintenance, and runtime deployment.

Many terminology servers exist. The challenge is that the complexity of these systems typically prevents them from being utilized by groups or individuals other than those who initially created them. A large number and wide spectrum of individuals must access an enterprise system of this type. If the database, terminology services, content, processes, and tooling are not considered as an integrated environment, the complexity of the system will prevent it from being understood and properly utilized. This work defined the interaction of these various components, described the dependencies and created a system that considers workflow, content integration, and maintenance challenges that to this date have not been adequately addressed.

Although it is difficult to prove the generalizability of the approach and the work involved in creating and maintain content beyond its successful implementation at 3M, the validation was able to establish that other consumers of terminology can interact with the environment using standard APIs.

The ultimate goal of this project is to support efforts to improve clinical outcomes and electronic interoperability of the EHR. Hospitals and providers have an escalating need to maintain a growing amount of standardized, structured data to comply with requirements, interpret massive amounts of complex data, and organize, summarize, and interpret information and outcomes. It is crucial that organizations be proficient in the management, access, and utilization of these data. The TME can act as a key component in an organization's data governance strategy. Combined with the right people, processes, and tools, the TME provides a practical migration path to help health care organizations structure, normalize, and standardize their data. It provides this capability without imposing undue burden on the organization and does not result in the loss of historical patient data.

12.4. Future Work

There are multiple opportunities for additional work in this area. New database designs, matching technologies, and informatics tools are areas that should be continually evaluated and applied to this base framework. The following sections describe some areas of potential future work.

12.4.1. Optimizations to current TME design and content.

With regard to design, some characteristics of the TME schema are aspects of the logical model that found their way into the physical model. It may be possible to simplify the database schema further

by doing things such as merging the PROPERTY and ENTITY_ATTRIBUTE tables. These tables were separated because the logical model made a distinction between defining entity attributes and general properties of code and other code attributes. This distinction in the logical model could be preserved without making two separate tables.

Similarly, version metadata was integrated directly into each of the core TME tables as opposed to breaking out version information into a separate table. This makes it awkward to add additional version metadata (e.g., "Date of deployment") to TME entities.

TFT terminology models are adhered to mostly through tool business logic or SME training. Implementing formal ontology in the TFT would organize the content and provide a valuable QC mechanism with automated validation tools.

12.4.2. More terminology tools. Several applications were created to support the TME,[79,84-87] but tools to manage workflow and map content can always be enhanced, specifically in the area of searching and matching. This capability is not only important for managing terminology, but also for using it. Structured documentation allows clinicians to use picklists for data entry. The picklists reference value sets in the terminology. In some cases, these value sets can get very large (e.g., lists of problems or diagnoses). It is critical to be able to find the appropriate concept in a large list, quickly. Coming up with better ways to find a target concept in large volumes of content is an area for future work.

12.4.3. Better ways to address the ETL. Currently, the initial ETL for integrating complex new source terminology into the TME is one of the most difficult parts of the process. It requires informatics expertise and a very good understanding of both source

and target logical and physical models. It takes time to gain familiarity with the new source data model and then time to reconcile differences and map the models in the ETL process. Another option to explore is promoting exchange of data among vocabulary servers using Resource Description Framework (RDF). RDF is a standard model for data exchange that facilitates data interoperability even when the underlying data models differ.[88] Using RDF as an interlingua would simplify the ETL process allowing source content to be loaded in the appropriate namespace without having to first map all the divergent schemas.

12.4.4. Alignment with CTS v2.0. Common Terminology Services v2.0 (CTS v2.0) is the most current version of the CTS specification. It seeks to expand the original functionality of CTS v1.2 particularly in the area of content administration, authoring, and distribution.[82] At high level, the TME is aligned with CTS v2.0's overall model. However, a detailed analysis similar to the one performed against CTS v1.2 could also be done as another form of external validation.

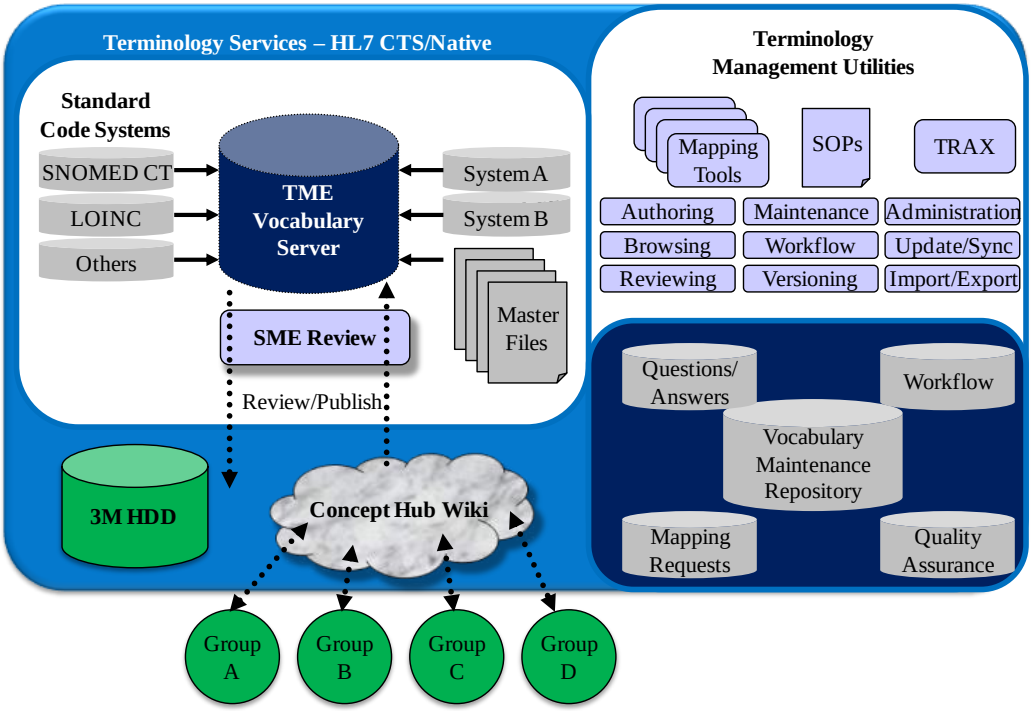


Figure 24: TME Overview. Example TME implementation

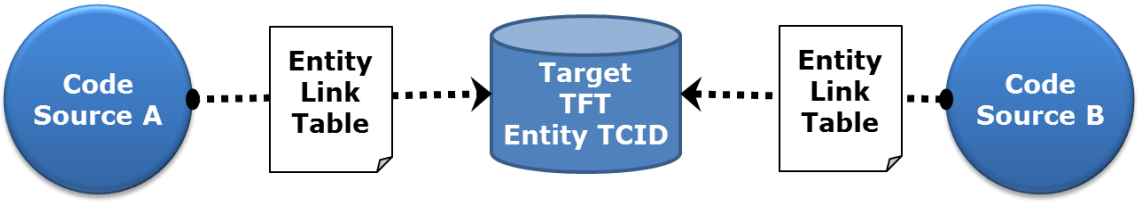


Figure 25: Entity Link Metadata.

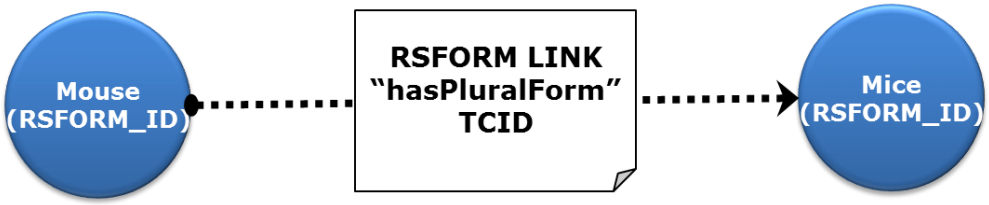


Figure 26: Representation Link Metadata.

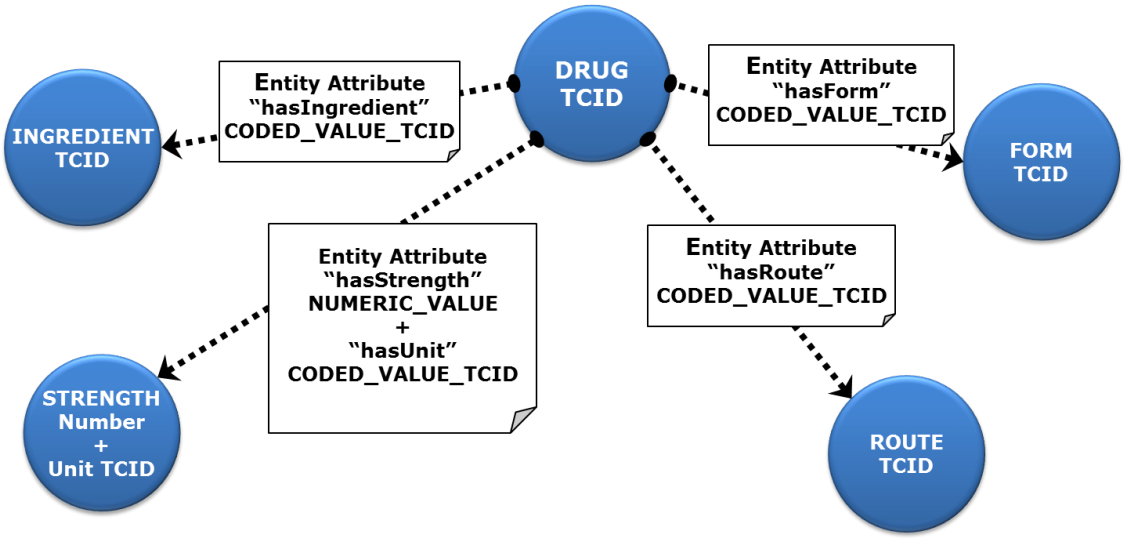


Figure 27: Entity Attribute Example.

13. CONCLUSION

This project has defined the role and benefits of standard terminology and describes the challenges and barriers that exist for consistent and sustainable use of coded terminologies in EHR systems. Various approaches to implementing standard terminologies were explored and finally the TME, a working system that meets the requirements for creating, maintaining, and distributing coded terminologies used in EHR systems, was created and evaluated against defined requirements and the industry standard Common Terminology Services (CTS).

The TME approach involves creation of a federated terminology, the TFT, that integrates multiple standards and local terminologies into a single source of truth for the meaning and structure of clinical content (see Figure 28). In this approach, legacy systems continue to collect data using local codes. The local terminology is linked to standard terminologies through the TME. This approach of creating a centralized vocabulary server that houses a federated terminology and using mappings as the means to integrate both standards and local terminologies across multiple namespaces provides an efficient, flexible, extensible approach for managing fit-for-purpose content and clinical code systems.

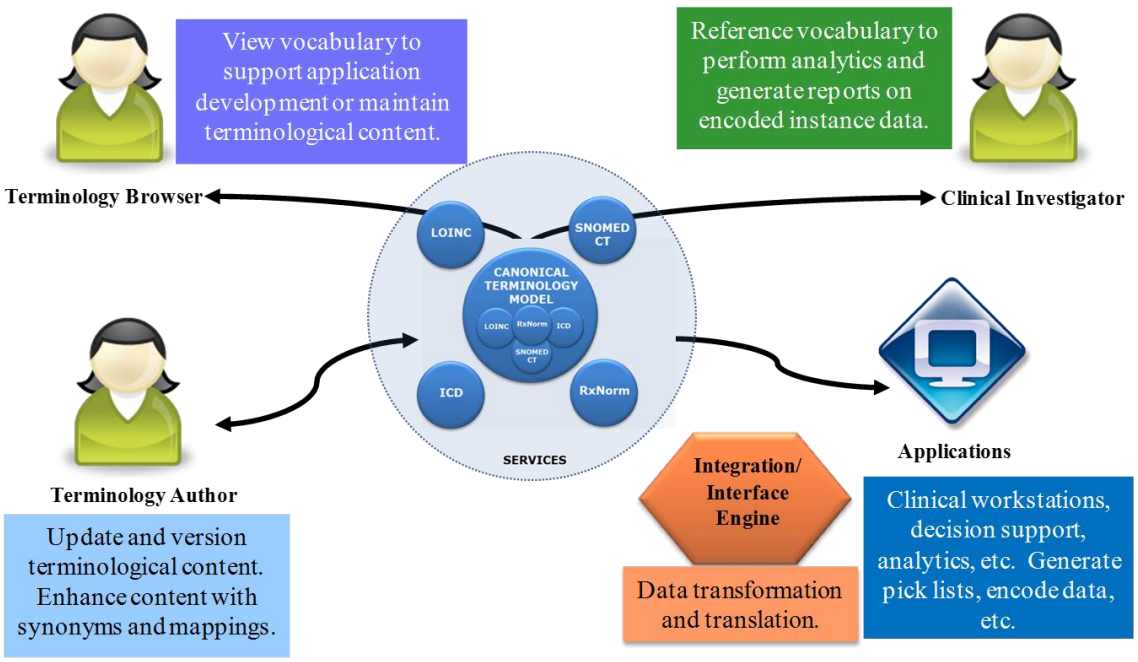


Figure 28: TME Support for Users, Applications, and Interfaces. Summary of TME model and interactions with users and systems.

APPENDIX A

TME CORE TABLES

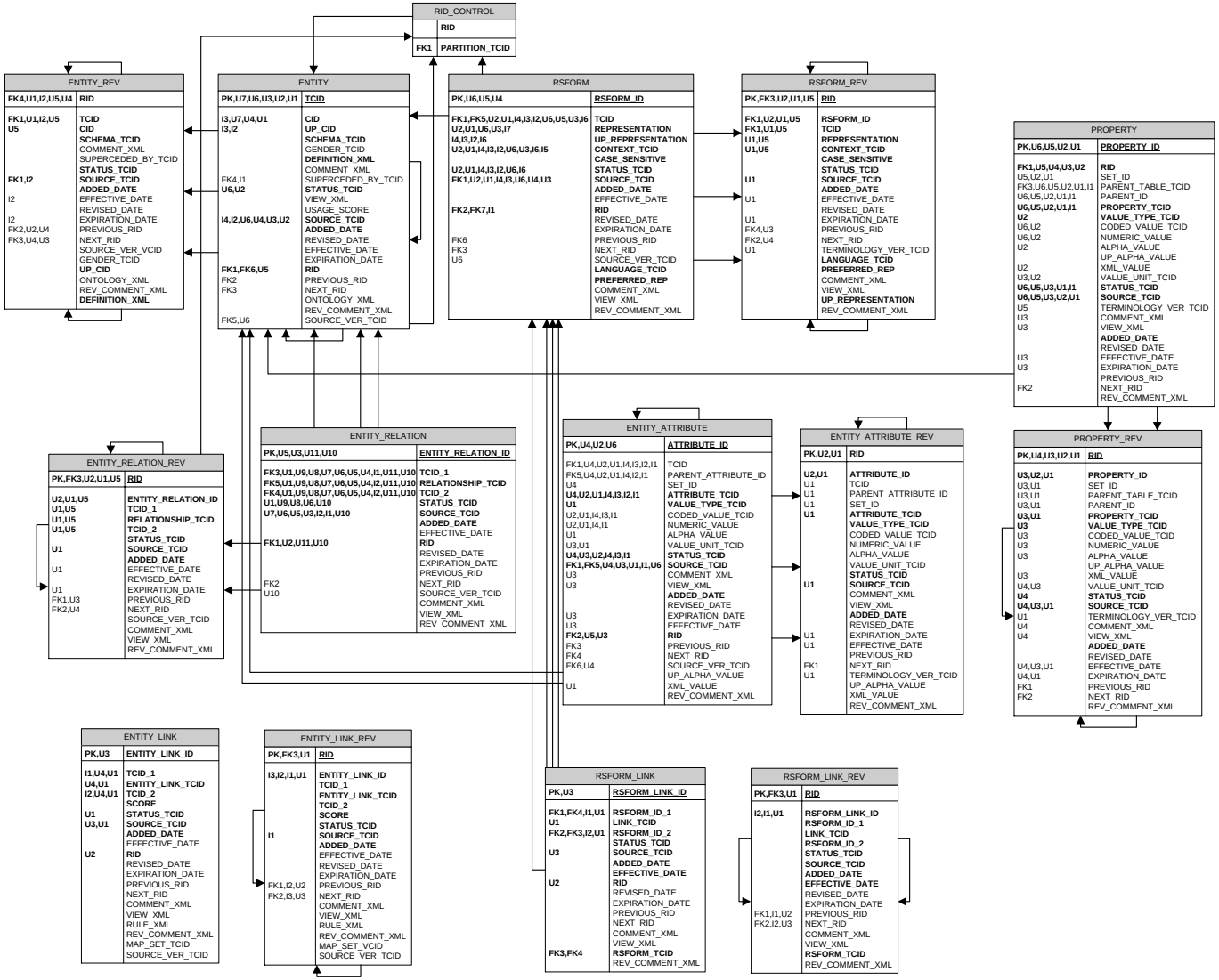


Figure 29: TME ERD Table Description Summary

Table 15: Summary of TME Tables. List of TME table names and descriptions.

TME TABLE NAME	TME TABLE DESCRIPTION
RID_CONTROL (see Table 16 for detailed description)	The RID_CONTOL table is used to generate and maintain unique revision identifiers (RIDs). RIDs are used by the other TME to index changes to content. The RID_CONTROL table can be considered as several same-structured tables combined into one (see PARTITION_TCID description). Because RIDs must be unique across all core TME tables, they are stored in the RID_CONTROL, where designation as a primary key can be used to enforce uniqueness.
ENTITY (see Table 17 for detailed description)	The ENTITY table stores the "most-future" revision of every entity in the TME. This means that it is possible that a revision of an entity in this table will not be the "current-effective" revision and the ENTITY_REV table will need to be referenced in order to determine the most current revision of the entity.
ENTITY_REV (see Table 18 for detailed description)	The ENTITY_REV table stores entity revisions. Anytime an entity is inserted, updated, or deleted in the ENTITY table, a row is created in the ENTITY_REV table. Except where commented otherwise, all fields are identical to the ENTITY table.
RSFORM (see Table 19 for detailed description)	The RSFORM table stores entity related surface forms (RSFORMs) or designations/displays/representations/codes used to label an entity in a specific CONTEXT_TCID.
RSFORM_REV (see Table 20 for detailed description)	The RSFORM_REV table stores entity representation revisions. Anytime an entity representation is inserted, updated, or deleted in the RSFORM table, a row is created in the RSFORM_REV table. Except where commented otherwise, all fields are identical to the RSFORM table.
RSFORM_LINK (see Table 21 for detailed description)	The RSFORM_LINK table is used to stores named associations among entity representations.
RSFORM_LINK_REV (see Table 22 for detailed description)	The RSFORM_LINK_REV table stores representation link revisions. Anytime a representation link is inserted, updated, or deleted in the RSFORM_LINK table, a row is created in the RSFORM_LINK_REV table. Except where commented otherwise, all fields are identical to the RSFORM_LINK table.

Table 15: Continued

TME TABLE NAME	TME TABLE DESCRIPTION
ENTITY_RELATION (see Table 23 for detailed description)	The ENTITY_RELATION table stores all named relationships among entities in the same namespace.
ENTITY_RELATION_REV (see Table 24 for detailed description)	The ENTITY_RELATION_REV table stores entity relationship revisions. Anytime a relationship is inserted, updated, or deleted in the ENTITY_RELATION table, a row is created in the ENTITY_RELATION_REV table. Except where commented otherwise, all fields are identical to the ENTITY_RELATION table.
ENTITY_LINK (see Table 25 for detailed description)	The ENTITY_LINK table stores named links (mapping types) between entities. ENTITY_LINK is not meant to hold relationships among entities. Entity links may cross TME namespaces, while entity relationships are not permitted to cross TME namespaces.
ENTITY_LINK_REV (see Table 26 for detailed description)	The ENTITY_LINK_REV table stores entity link revisions. Anytime an entity link is inserted, updated, or deleted in the ENTITY_LINK table, a row is created in the ENTITY_LINK_REV table. Except where commented otherwise, all fields are identical to ENTITY_LINK.
ENTITY_ATTRIBUTE (see Table 27 for detailed description)	The ENTITY_ATTRIBUTE table stores all attributes of entities. It allows for the data type of the attribute to be specified accommodating name-value pairs.
ENTITY_ATTRIBUTE_REV (see Table 28 for detailed description)	The ENTITY_ATTRIBUTE_REV table Stores ENTITY_ATTRIBUTE revisions. Anytime a attribute is inserted, updated, or deleted in the ENTITY_ATTRIBUTE table, a row is created in the ENTITY_ATTRIBUTE_REV table. Except where commented otherwise, all fields are identical to ENTITY_ATTRIBUTE.
PROPERTY (see Table 29 for detailed description)	The PROPERTY table stores name-value pairs that are properties of entities, representations, relationships, mapping links, representation links, and other properties.
PROPERTY_REV (see Table 30 for detailed description)	The PROPERTY_REV table stores property revisions. Anytime a property is inserted, updated, or deleted in the PROPERTY table, a row is created in the PROPERTY_REV table. Except where commented otherwise, all fields are identical to PROPERTY.

Table 16: RID_CONTROL Table Detail. The RID_CONTOL table is used to generate and maintain unique revision identifiers (RIDs). RIDs are used by the other TME to index changes to content. The RID_CONTROL table can be considered as several same-structured tables combined into one (see PARTITION_TCID description). Because RIDs must be unique across all core TME tables, they are stored in the RID_CONTROL, where designation as a primary key can be used to enforce uniqueness. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
RID	Integer (22)	PK	YES	Unique Revision ID used to identify versions of TME objects (i.e., entities, relationships, representations, attributes, and links). The RID is generated by a java routine that encodes a check-digit into the RID value to be used for some future purpose.
PARTITION_TCID	Integer (22)		YES	The table set in which this RID is used. Table set means the combination of a main TME table and its associated revision table (i.e., ENTITY / ENTITY_REV, ENTITY_RELATION / ENTITY_RELATION_REV, RSFORM / RSFORM_REV, ATTRIBUTE / ATTRIBUTE_REV, ENTITY_LINK / ENTITY_LINK_REV, and RSFORM_LINK / RSFORM_LINK_REV). This is required because RID is used in all partitions.

Table 17: ENTITY Table Detail. The ENTITY table stores the "most-future" revision of every entity in the TME. This means that it is possible that a revision of an entity in this table will not be the "current-effective" revision and the ENTITY_REV table will need to be referenced in order to determine the most current revision of the entity. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
TCID	Integer (22)	PK	YES	TME Component Identifier; TCID is the ENTITY table primary key—a numeric identifier for unique content in the ENTITY table. However, TCID is not unique in the ENTITY revision table (ENTITY_REV), since multiple previous and future-effective versions of the same TCID are tracked in ENTITY_REV. All other fields in the TME that have a suffix of "TCID" are foreign keys that reference a TFT concept in the ENTITY table.
CID	Varchar (4000)		YES	Component Identifier; CID is an alpha-numeric identifier for unique content in the ENTITY table. The CID cannot contain nonprintable characters or spaces. The CID for all concepts in each source-terminology namespace (identified by entity.SOURCE_TCID) is prefixed by the name of the source code system—e.g., TFTActiveStatus, LOINCGlucose, RxNormGlucose, etc. The CID is critical to the TME in the diffing step of the ETL process. A consistently-produced CID is used to determine if there have been changes to source content for sources that do not assign identifiers for all the TME required data elements (e.g., lacking a controlled identifier for description/representations) or have appropriate change control.

Table 17: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
UP_CID	Varchar (4000)		YES	Uppercase Component Identifier; UP_CID is an all uppercase CID. It is used to improve performance by allowing for case-insensitive comparisons of CIDs during diffing. There are instances where term case implies some of the semantics of the underlying concept (e.g., units of measure; m = meters, M = Moles). In these instances, UP_CID is not used for the comparison.
SCHEMA_TCID	Integer (22)		YES	References a TCID from the ENTITY table for the TFT concept that denotes the schema of this entity. Schema concepts are stored in the TFT namespace. The schema is not set by the source code system, and is currently always set to TCID 2 = TFTSchema.
DEFINITION_XML	Clob (0)		YES	XML string that holds human readable definition of entity.
COMMENT_XML	Clob (0)		NO	XML string that holds author comments regarding an entity.
ONTOLOGY_XML	Clob (0)		NO	XML string that holds a formal definition of the entity.
SUPERSEDED_BY_TCID	Integer (22)	FK	NO	References a TCID from the ENTITY table for the entity that replaces this entity.
STATUS_TCID	Integer (22)		YES	References a TCID from the ENTITY table for the TFT concept that denotes the status of this entity.
SOURCE_TCID	Integer (22)		YES	References a TCID from the ENTITY table for the TFT concept that denotes the source code system of this entity.
ADDED_DATE	Date (0)		YES	The date/time (GMT) when this revision of this entity was added to the TME.
EFFECTIVE_DATE	Date (0)		YES	The date when this revision of this entity became effective in the source code system.

Table 17: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
REV_COMMENT_XML	Clob (0)		NO	XML string that holds human readable comment regarding the nature of the current revision of this entity.
RID	Integer (22)	FK	YES	Unique Revision ID used to identify versions of this entity.

Table 18: ENTITY_REV Table Detail. The ENTITY_REV table stores entity revisions. Anytime an entity is inserted, updated, or deleted in the ENTITY table, a row is created in the ENTITY_REV table. Except where commented otherwise, all fields are identical to the ENTITY table. *Primary Key(PK)/Foreign Key(FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
TCID	Integer (22)	FK	NO	See ENTITY table. Note TCID is not required in the revision tables.
CID	Varchar (4000)		YES	See ENTITY table.
SCHEMA_TCID	Integer (22)		YES	See ENTITY table.
DEFINITION_XML	Clob (0)		YES	See ENTITY table.
COMMENT_XML	Clob (0)		NO	See ENTITY table.
ONTOLOGY_XML	Clob (0)		NO	See ENTITY table.
SUPERSEDED_BY_TCID	Integer (22)		NO	See ENTITY table.
STATUS_TCID	Integer (22)		YES	See ENTITY table.
SOURCE_TCID	Integer (22)		YES	See ENTITY table.
ADDED_DATE	Date (0)		YES	See ENTITY table.
EFFECTIVE_DATE	Date (0)		YES	See ENTITY table.
REV_COMMENT_XML	Clob (0)		NO	See ENTITY table.
RID	Integer (22)	PK/FK	YES	See ENTITY table.

Table 18: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
REVISED_DATE	Date (0)		NO	The date/time (GMT) when this revision of this entity was changed in the TME.
EXPIRATION_DATE	Date (0)		NO	The date/time (GMT) when this revision of this entity expires (status changed to inactive) in the TME.
PREVIOUS_RID	Integer (22)	FK	NO	The RID for the previous revision to this entity.
NEXT_RID	Integer (22)	FK	YES	The RID for the next revision to this entity.

Table 19: RSFORM Table Detail. The RSFORM table stores entity related surface forms (RSFORMs) or designations/displays/representations/codes used to label an entity in a specific CONTEXT_TCID. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
RSFORM_ID	Integer (22)	PK	YES	Primary key of the RSFORM table. It uniquely identifies the related surface form in the RSFORM table.
TCID	Integer (22)	FK	YES	Foreign key that links this surface form to an entity in the ENTITY table. Many surface forms can be associated with a single entity.
REPRESENTATION	Varchar (4000)		YES	A text string that is associated with the referenced entity through the RSFORM.TCID.
UP_REPRESENTATION	Varchar (4000)		YES	RSFORM.REPRESENTATION in all uppercase characters.
CONTEXT_TCID	Integer (22)	FK	YES	References a TCID from the ENTITY table for a TFT concept that denotes the type of surface form.
CASE_SENSITIVE	Varchar (1)		YES	Boolean value that flags whether REPRESENTATION is case-sensitive.

Table 19: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
STATUS_TCID	Integer (22)		YES	References a TCID from the ENTITY table for the TFT concept that denotes the status of this surface form.
SOURCE_TCID	Integer (22)		YES	References a TCID from the ENTITY table for the TFT concept that denotes the source code system of this surface form.
ADDED_DATE	Date (0)		YES	The date/time (GMT) when this revision of this surface form was added to the TME.
EFFECTIVE_DATE	Date (0)		NO	The date when this revision of this surface form became effective in the source code system.
REV_COMMENT_XML	Clob (0)		NO	XML string that holds human readable comment regarding the nature of the current revision of this surface form.
RID	Integer (22)	FK	YES	Unique Revision ID used to identify versions of this surface form.

Table 20: RSFORM_REV Table Detail. The RSFORM_REV table stores entity representation revisions. Anytime an entity representation is inserted, updated, or deleted in the RSFORM table, a row is created in the RSFORM_REV table. Except where commented otherwise, all fields are identical to the RSFORM table. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
RSFORM_ID	Integer (22)	FK	YES	See RSFORM table.
TCID	Integer (22)		YES	See RSFORM table.
REPRESENTATION	Varchar (4000)		YES	See RSFORM table.
CONTEXT_TCID	Integer (22)		YES	See RSFORM table.
CASE_SENSITIVE	Varchar (1)		YES	See RSFORM table.

Table 20: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
STATUS_TCID	Integer (22)		YES	See RSFORM table.
SOURCE_TCID	Integer (22)		YES	See RSFORM table.
ADDED_DATE	Date (0)		YES	See RSFORM table.
EFFECTIVE_DATE	Date (0)		YES	See RSFORM table.
REV_COMMENT_XML	Clob (0)		NO	See RSFORM table.
RID	Integer (22)	PK/FK	YES	See RSFORM table.
REVISED_DATE	Date (0)		NO	The date/time (GMT) when this revision of this representation was changed in the TME.
EXPIRATION_DATE	Date (0)		NO	The date/time (GMT) when this revision of this representation expires (status changed to inactive) in the TME.
PREVIOUS_RID	Integer (22)	FK	NO	The RID for the previous revision to this representation.
NEXT_RID	Integer (22)	FK	YES	The RID for the next revision to this representation.

Table 21: RSFORM_LINK Table Detail. The RSFORM_LINK table is used to stores named associations among entity representations. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
RSFORM_LINK_ID	Integer (22)	PK	No	Primary key of the RSFORM_LINK table. It uniquely identifies the representation link triplet in the RSFORM_LINK table.
RSFORM_ID_1	Integer (22)	FK	No	References an RSFORM_ID from the RSFORM table for the first surface form in a representation link triplet.

Table 21: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
RSFORM_ID_2	Integer (22)	FK	No	References an RSFORM_ID from the RSFORM table for the second surface form in a representation link triplet.
LINK_TCID	Integer (22)	FK	Yes	References a TCID from the ENTITY table for a TFT concept that denotes the kind of link between RSFORM_ID_1 and RSFORM_ID_2.
STATUS_TCID	Integer (22)		No	References a TCID from the ENTITY table for the TFT concept that denotes the status of this link.
SOURCE_TCID	Integer (22)		No	References a TCID from the ENTITY table for the TFT concept that denotes the source code system of this link.
ADDED_DATE	Date (0)		No	The date/time (GMT) when this revision of this link was added to the TME. This date/time is stamped as system date by the TME database before insert.
EFFECTIVE_DATE	Date (0)		No	The date when this revision of this link became effective in the source code system. This date must be provided by the source code system, or it is set to null.
REV_COMMENT_XML	Clob (0)		Yes	XML string that holds author comments regarding this link.
RID	Integer (22)	FK	Yes	Revision Identifier; Every time an insert or update is performed against the RSFORM_LINK table a trigger writes a copy of this revision in both the RSFORM_LINK table and the RSFORM_LINK_REV table.

Table 22: RSFORM_LINK_REV Table Detail. The RSFORM_LINK_REV table stores representation link revisions. Anytime a representation link is inserted, updated, or deleted in the RSFORM_LINK table, a row is created in the RSFORM_LINK_REV table. Except where commented otherwise, all fields are identical to the RSFORM_LINK table. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
RSFORM_LINK_ID	Integer (22)	FK	No	See RSFORM_LINK table.
RSFORM_ID_1	Integer (22)		No	See RSFORM_LINK table.
RSFORM_ID_2	Integer (22)		No	See RSFORM_LINK table.
LINK_TCID	Integer (22)		No	See RSFORM_LINK table.
STATUS_TCID	Integer (22)		No	See RSFORM_LINK table.
SOURCE_TCID	Integer (22)		No	See RSFORM_LINK table.
ADDED_DATE	Date (0)		No	See RSFORM_LINK table.
EFFECTIVE_DATE	Date (0)		No	See RSFORM_LINK table.
REV_COMMENT_XML	Clob (0)		Yes	See RSFORM_LINK table.
RID	Integer (22)	PK/FK	No	See RSFORM_LINK table.
REVISED_DATE	Date (0)		Yes	The date/time (GMT) when this revision of this link was changed in the TME.
EXPIRATION_DATE	Date (0)		Yes	The date/time (GMT) when this revision of this link expires (status changed to inactive) in the TME.
PREVIOUS_RID	Integer (22)	FK	No	The RID for the previous revision to this link.
NEXT_RID	Integer (22)	FK	No	The RID for the next revision to this link.

Table 23: ENTITY_RELATION Table Detail. The ENTITY_RELATION table stores all named relationships among entities in the same namespace. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
ENTITY_RELATION_ID	Integer (22)	PK	No	Primary key of the ENTITY_RELATIONSHIP table. It uniquely identifies an entity relationship triplet in the ENTITY_RELATIONSHIP table. It is a foreign key in the ENTITY_RELATIONSHIP revision table (ENTITY_RELATIONSHIP_REV).
TCID_1	Integer (22)	FK	Yes	References a TCID from the ENTITY table for the first entity in a relationship triplet (e.g., "Cytomegalovirus" (first entity) | "is-a" (relationship) | "Virus" (second entity)).
TCID_2	Integer (22)	FK	Yes	References a TCID from the ENTITY table for the second entity in a relationship triplet (e.g., "Cytomegalovirus" (first entity) | "is-a" (relationship) | "Virus" (second entity)).
RELATIONSHIP_TCID	Integer (22)	FK	Yes	References a TCID from the ENTITY table for a TFT concept that denotes the kind of relationship between the first entity and the second entity. Relationship concepts (i.e., concepts that represent relationships) are stored in the TFT namespace (e.g., "Cytomegalovirus" (first entity) | "is-a" (relationship) | "Virus" (second entity)).
STATUS_TCID	Integer (22)		No	References a TCID from the ENTITY table for the TFT concept that denotes the status of this relation
SOURCE_TCID	Integer (22)		No	References a TCID from the ENTITY table for the TFT concept that denotes the source code system of this relation.

Table 23: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
TCID_1_SOURCE_TCID	Integer (22)		Yes	References a TCID from the ENTITY table for the TFT concept that denotes the source code system of this relation.
TCID_2_SOURCE_TCID	Integer (22)		Yes	References a TCID from the ENTITY table for the TFT concept that denotes the source code system of the first entity in the relationship triplet.
ADDED_DATE	Date (0)		No	References a TCID from the ENTITY table for the TFT concept that denotes the source code system of the second entity in the relationship triplet.
EFFECTIVE_DATE	Date (0)		No	The date when this revision of this relationship became effective in the source code system. This date must be provided by the source code system, or it is set to null.
REV_COMMENT_XML	Clob (0)		Yes	XML string that holds author comments regarding this relationship.
RID	Integer (22)	FK	No	Revision Identifier; Every time an insert or update is performed against the ENTITY_RELATION table a trigger writes a copy of this revision in both the ENTITY_RELATION table and the ENTITY_RELATION_REV table.

Table 24: ENTITY_RELATION_REV Table Detail. The ENTITY_RELATION_REV table stores entity relationship revisions. Anytime a relationship is inserted, updated, or deleted in the ENTITY_RELATION table, a row is created in the ENTITY_RELATION_REV table. Except where commented otherwise, all fields are identical to the ENTITY_RELATION table. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
ENTITY_RELATION_ID	Integer (22)	FK	No	See ENTITY_RELATION table.
TCID_1	Integer (22)		No	See ENTITY_RELATION table.
TCID_2	Integer (22)		No	See ENTITY_RELATION table.
RELATIONSHIP_TCID	Integer (22)		No	See ENTITY_RELATION table.
STATUS_TCID	Integer (22)		No	See ENTITY_RELATION table.
SOURCE_TCID	Integer (22)		No	See ENTITY_RELATION table.
TCID_1_SOURCE_TCID	Integer (22)		Yes	See ENTITY_RELATION table.
TCID_2_SOURCE_TCID	Integer (22)		Yes	See ENTITY_RELATION table.
ADDED_DATE	Date (0)		No	See ENTITY_RELATION table.
EFFECTIVE_DATE	Date (0)		No	See ENTITY_RELATION table.
REV_COMMENT_XML	Clob (0)		Yes	See ENTITY_RELATION table.
RID	Integer (22)	PK/FK	No	See ENTITY_RELATION table.
REVISED_DATE	Date (0)		Yes	The date/time (GMT) when this revision of this relationship was changed in the TME.
EXPIRATION_DATE	Date (0)		Yes	The date/time (GMT) when this revision of this relationship expires (status changed to inactive) in the TME.
PREVIOUS_RID	Integer (22)	FK	No	The RID for the previous revision to this relationship.
NEXT_RID	Integer (22)	FK	No	The RID for the next revision to this relationship.

Table 25: ENTITY_LINK Table Detail. The ENTITY_LINK table stores named links (mapping types) between entities. ENTITY_LINK is not meant to hold relationships among entities. Entity links may cross TME namespaces, while entity relationships are not permitted to cross TME namespaces. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
ENTITY_LINK_ID	integer(22)	PK	No	Primary key of the ENTITY_LINK table. It uniquely identifies an entity link triplet in the ENTITY_LINK table. It is a foreign key in the ENTITY_LINK revision table (ENTITY_LINK_REV).
TCID_1	integer(22)	FK	Yes	References a TCID from the ENTITY table for the first entity in a map link triplet (e.g., "SNOMEDCTCytomegalovirus" (first entity in SNOMED CT namespace) | "is-clinically-equivalent-to" (link type) | "TFTCMV" (second entity in TFT namespace)).
TCID_2	integer(22)	FK	Yes	References a TCID from the ENTITY table for the second entity in a map link triplet (e.g., "SNOMEDCTCytomegalovirus" (first entity in SNOMED CT namespace) | "is-clinically-equivalent-to" (link type) | "TFTCMV" (second entity in TFT namespace)).
ENTITY_LINK_TCID	integer(22)	FK	Yes	References a TCID from the ENTITY table for a TFT concept that denotes the kind of link between the first entity and the second entity. Concepts that represent map link types are stored in the TFT namespace (e.g., TFTIsNarrowerThan, TFTIsEquivalentTo, TFTIsBroaderThan).
SCORE	integer(22)		No	Value that estimates quality of the entity link
RULE_XML	clob(0)		Yes	XML string that expresses the circumstances (rule) under which the link applies.
STATUS_TCID	integer(22)		No	References a TCID from the ENTITY table for the TFT concept that denotes the status of this link.

Table 25: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
SOURCE_TCID	integer(22)		No	References a TCID from the ENTITY table for the TFT concept that denotes the source code system of this link.
ADDED_DATE	date(0)		No	References a TCID from the ENTITY table for the TFT concept that denotes the source code system for the link.
EFFECTIVE_DATE	date(0)		No	The date when this revision of this link became effective in the source code system. This date must be provided by the source code system, or it is set to null.
REV_COMMENT_XML	clob(0)		Yes	XML string that holds author comments regarding this link.
RID	integer(22)	FK	Yes	Revision Identifier; Every time an insert or update is performed against the ENTITY_LINK table a trigger writes a copy of this revision in both the ENTITY_LINK table and the ENTITY_LINK_REV table.

Table 26: ENTITY_LINK_REV Table Detail. The ENTITY_LINK_REV table stores entity link revisions. Anytime an entity link is inserted, updated, or deleted in the ENTITY_LINK table, a row is created in the ENTITY_LINK_REV table. Except where commented otherwise, all fields are identical to ENTITY_LINK. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
ENTITY_LINK_ID	integer(22)	FK	No	See ENTITY_LINK table.
TCID_1	integer(22)		No	See ENTITY_LINK table.
TCID_2	integer(22)		No	See ENTITY_LINK table.
ENTITY_LINK_TCID	integer(22)		No	See ENTITY_LINK table.
SCORE	integer(22)		No	See ENTITY_LINK table.

Table 26: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
RULE_XML	clob(0)		Yes	See ENTITY_LINK table.
STATUS_TCID	integer(22)		No	See ENTITY_LINK table.
SOURCE_TCID	integer(22)		No	See ENTITY_LINK table.
ADDED_DATE	date(0)		No	See ENTITY_LINK table.
EFFECTIVE_DATE	date(0)		No	See ENTITY_LINK table.
REV_COMMENT_XML	clob(0)		Yes	See ENTITY_LINK table.
RID	integer(22)	PK/FK	No	See ENTITY_LINK table.
REVISED_DATE	date(0)		Yes	The date/time (GMT) when this revision of this link was changed in the TME.
EXPIRATION_DATE	date(0)		Yes	The date/time (GMT) when this revision of this link expires (status changed to inactive) in the TME.
PREVIOUS_RID	integer(22)	FK	No	The RID for the previous revision to this link.
NEXT_RID	integer(22)	FK	No	The RID for the next revision to this link.

Table 27: ENTITY_ATTRIBUTE Table Detail. The ENTITY_ATTRIBUTE table stores all attributes of entities. It allows for the data type of the attribute to be specified accommodating name-value pairs. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
ATTRIBUTE_ID	integer(22)	PK	No	Primary key of the ENTITY_ ATTRIBUTE table. It uniquely identifies an entity attribute in the ENTITY_ ATTRIBUTE table. It is a foreign key in the ENTITY_ ATTRIBUTE revision table (ENTITY_ ATTRIBUTE _REV).

Table 27: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
SET_ID	integer(22)		Yes	Identifies the attribute set to which this attribute belongs. Used to group attributes. For example, it is necessary to associate a strength with a particular ingredient in drugs that have multiple active ingredients. SET_ID is used to associate each ingredient with the appropriate strength.
TCID	integer(22)	FK	Yes	Foreign Key to ENITY table.
PARENT_ATTRIBUTE_ID	integer(22)	FK	Yes	ATTRIBUTE_ID of the parent entity attribute if this is an attribute of another entity attribute; null otherwise. This is the primary method of combining several attributes into hierarchical group.
ATTRIBUTE_TCID	integer(22)		No	References a TCID from the ENTITY table for a TFT concept that denotes the type of attribute this attribute is. Attribute type concepts are stored in the source terminology namespace in which they are used (e.g., LOINCAttributeAxis1), and are provided by, or derived from, source code system data.
VALUE_TYPE_TCID	integer(22)		No	References a TCID from the ENTITY table for a TFT concept that denotes the type of value associated with this attribute. Value type concepts are stored in the TFT namespace (e.g., TFTCodedValue, TFTAlphaNumericValue, TFTNumericValue, and TFTXMLValue).

Table 27: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
CODED_VALUE_TCID	integer(22)		Yes	References a TCID from the ENTITY table for an entity that denotes the value of this attribute if ENTITY_ATTRIBUTE.VALUE_TYPE_TCID is TFTCodedValue. Coded value entities are stored in the source terminology namespace in which they are used (e.g., LOINCGlucose), and are provided by, or derived from, source code system data.
NUMERIC_VALUE	integer(22)		Yes	Populated with a numeric value for the attribute name-value pair, when ENTITY_ATTRIBUTE.VALUE_TYPE_TCID is TFTNumericValue.
ALPHA_VALUE	varchar(4000)		Yes	Populated with an alphanumeric value for the attribute name-value pair, when ENTITY_ATTRIBUTE.VALUE_TYPE_TCID is TFTAlphaNumericValue.
UP_ALPHA_VALUE	varchar(4000)		Yes	An all uppercase representation of ENTITY_ATTRIBUTE.ALPHA_VALUE used to facilitate matching during diffing and mapping.
XML_VALUE	clob(0)		Yes	Populated with an XML string for the attribute name-value pair, when ENTITY_ATTRIBUTE.VALUE_TYPE_TCID is TFTXMLValue.
VALUE_UNIT_TCID	integer(22)		Yes	References a TCID from the ENTITY table for a TFT concept that denotes the unit in which the value of this attribute is expressed. Value unit concepts are stored in the TFT namespace (e.g., TFTMilligrams).

Table 27: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
STATUS_TCID	integer(22)		No	References a TCID from the ENTITY table for the TFT concept that denotes the status of this attribute.
SOURCE_TCID	integer(22)		No	References a TCID from the ENTITY table for the TFT concept that denotes the source code system of this attribute.
ADDED_DATE	date(0)		No	References a TCID from the ENTITY table for the TFT concept that denotes the source code system for the attribute.
EFFECTIVE_DATE	date(0)		No	The date when this revision of this attribute became effective in the source code system. This date must be provided by the source code system, or it is set to null.
REV_COMMENT_XML	clob(0)		Yes	XML string that holds author comments regarding this attribute.
RID	integer(22)	FK	Yes	Revision Identifier; Every time an insert or update is performed against the ENTITY_ATTRIBUTE table a trigger writes a copy of this revision in both the ENTITY_ATTRIBUTE table and the ENTITY_ATTRIBUTE_REV table.

Table 28: ENTITY_ATTRIBUTE_REV Table Detail. The ENTITY_ATTRIBUTE_REV table Stores ENTITY_ATTRIBUTE revisions. Anytime an attribute is inserted, updated, or deleted in the ENTITY_ATTRIBUTE table, a row is created in the ENTITY_ATTRIBUTE_REV table. Except where commented otherwise, all fields are identical to ENTITY_ATTRIBUTE. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
ATTRIBUTE_ID	integer(22)	FK	No	See ENTITY_ATTRIBUTE table.
SET_ID	integer(22)		Yes	See ENTITY_ATTRIBUTE table.
TCID	integer(22)		Yes	See ENTITY_ATTRIBUTE table.
RSFORM_ID	integer(22)		Yes	See ENTITY_ATTRIBUTE table.
CONCEPT_RELATION_ID	integer(22)		Yes	See ENTITY_ATTRIBUTE table.
CONCEPT_LINK_ID	integer(22)		Yes	See ENTITY_ATTRIBUTE table.
RSFORM_LINK_ID	integer(22)		Yes	See ENTITY_ATTRIBUTE table.
PARENT_ATTRIBUTE_ID	integer(22)		Yes	See ENTITY_ATTRIBUTE table.
ATTRIBUTE_TCID	integer(22)		No	See ENTITY_ATTRIBUTE table.
VALUE_TYPE_TCID	integer(22)		No	See ENTITY_ATTRIBUTE table.
CODED_VALUE_TCID	integer(22)		Yes	See ENTITY_ATTRIBUTE table.
NUMERIC_VALUE	integer(22)		Yes	See ENTITY_ATTRIBUTE table.
ALPHA_VALUE	varchar(4000)		Yes	See ENTITY_ATTRIBUTE table.
XML_VALUE	clob(0)		Yes	See ENTITY_ATTRIBUTE table.
VALUE_UNIT_TCID	integer(22)		Yes	See ENTITY_ATTRIBUTE table.
STATUS_TCID	integer(22)		No	See ENTITY_ATTRIBUTE table.
SOURCE_TCID	integer(22)		No	See ENTITY_ATTRIBUTE table.
ADDED_DATE	date(0)		No	See ENTITY_ATTRIBUTE table.
EFFECTIVE_DATE	date(0)		No	See ENTITY_ATTRIBUTE table.
REV_COMMENT_XML	clob(0)		Yes	See ENTITY_ATTRIBUTE table.
RID	integer(22)	PK/FK	No	See ENTITY_ATTRIBUTE table.
REVISED_DATE	date(0)		Yes	The date/time (GMT) when this revision of this attribute was changed in the TME.

Table 28: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
EXPIRATION_DATE	date(0)		Yes	The date/time (GMT) when this revision of this attribute expires (status changed to inactive) in the TME.
PREVIOUS_RID	integer(22)	FK	No	The RID for the previous revision to this attribute.
NEXT_RID	integer(22)	FK	No	The RID for the next revision to this attribute.

Table 29: PROPERTY Table Detail. The PROPERTY table stores name-value pairs that are properties of entities, representations, relationships, mapping links, representation links, and other properties. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
PROPERTY_ID	integer(22)	PK	No	Primary key of the PROPERTY table. It uniquely identifies properties which can be associated with any component in the TME, including other properties. It is a foreign key in the PROPERTY revision table (PROPERTY_REV).
SET_ID	integer(22)		Yes	Identifies the property set to which this property belongs.
PARENT_TABLE_TCID	integer(22)	FK	Yes	References a TCID from the ENTITY table for a TFT concept that denotes the TME table of the PROPERTY.PARENT_ID.
PARENT_ID	integer(22)	FK	Yes	ID of the component that this property modifies or is associated with. This can be any of the core table primary keys including PROPERTY_ID. It is combined with PROPERTY.PARENT_TABLE_TCID as a compound foreign key.

Table 29: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
PROPERTY_TCID	integer(22)		No	References a TCID from the ENTITY table for an entity that describes this property. Property type entities are stored in the source terminology namespace in which they are used and are provided by, or derived from, source code system data (e.g., SNOMEDCTID).
VALUE_TYPE_TCID	integer(22)		No	References a TCID from the ENTITY table for the TFT concept that denotes the type of value this property has. This field is constrained to the same domain of TFT concepts as ENTITY_ATTRIBUTE.VALUE_TYPE_TCID.
CODED_VALUE_TCID	integer(22)		Yes	References a TCID from the ENTITY table for an entity that denotes the value of this property if PROPERTY.VALUE_TYPE_TCID is TFTCodedValue. Coded value entities are stored in the source terminology namespace in which they are used and are provided by, or derived from, source code system data.
NUMERIC_VALUE	integer(22)		Yes	Populated with a numeric value for the property name-value pair, when PROPERTY.VALUE_TYPE_TCID is TFTNumericValue.
ALPHA_VALUE	varchar(4000)		Yes	Populated with an alphanumeric value for the property name-value pair, when PROPERTY.VALUE_TYPE_TCID is TFTAlphaNumericValue.
UP_ALPHA_VALUE	varchar(4000)		Yes	An all uppercase representation of PROPERTY.ALPHA_VALUE used to facilitate matching during diffing.

Table 29: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
XML_VALUE	clob(0)		Yes	Populated with an XML string for the attribute name-value pair, when PROPERTY.VALUE_TYPE_TCID is TFTXMLValue.
VALUE_UNIT_TCID	integer(22)		Yes	References a TCID from the ENTITY table for a TFT concept that denotes the unit in which the value of this property is expressed. Value unit concepts are stored in the TFT namespace.
STATUS_TCID	integer(22)		No	References a TCID from the ENTITY table for the TFT concept that denotes the status of this property.
SOURCE_TCID	integer(22)		No	References a TCID from the ENTITY table for the TFT concept that denotes the source code system of this property.
ADDED_DATE	date(0)		No	References a TCID from the ENTITY table for the TFT concept that denotes the source code system for the property.
EFFECTIVE_DATE	date(0)		No	The date when this revision of this property became effective in the source code system. This date must be provided by the source code system, or it is set to null.
REV_COMMENT_XML	clob(0)		Yes	XML string that holds author comments regarding this property.
RID	integer(22)	FK	Yes	Revision Identifier; Every time an insert or update is performed against the ENTITY_PROPERTY table a trigger writes a copy of this revision in both the ENTITY_ PROPERTY table and the ENTITY_ PROPERTY_REV table.

Table 30: PROPERTY_REV Table Detail. The PROPERTY_REV table stores property revisions. Anytime a property is inserted, updated, or deleted in the PROPERTY table, a row is created in the PROPERTY_REV table. Except where commented otherwise, all fields are identical to PROPERTY. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
PROPERTY_ID	integer(22)	FK	No	See PROPERTY table.
SET_ID	integer(22)		Yes	See PROPERTY table.
PARENT_TABLE_TCID	integer(22)		Yes	See PROPERTY table.
RSFORM_LINK_ID	integer(22)		Yes	See PROPERTY table.
PARENT_ID	integer(22)		Yes	See PROPERTY table.
PROPERTY_TCID	integer(22)		No	See PROPERTY table.
VALUE_TYPE_TCID	integer(22)		No	See PROPERTY table.
CODED_VALUE_TCID	integer(22)		Yes	See PROPERTY table.
NUMERIC_VALUE	integer(22)		Yes	See PROPERTY table.
ALPHA_VALUE	varchar(4000)		Yes	See PROPERTY table.
XML_VALUE	clob(0)		Yes	See PROPERTY table.
VALUE_UNIT_TCID	integer(22)		Yes	See PROPERTY table.
STATUS_TCID	integer(22)		No	See PROPERTY table.
SOURCE_TCID	integer(22)		No	See PROPERTY table.
ADDED_DATE	date(0)		No	See PROPERTY table.
EFFECTIVE_DATE	date(0)		No	See PROPERTY table.
REV_COMMENT_XML	clob(0)		Yes	See PROPERTY table.
RID	integer(22)	PK/FK	No	See PROPERTY table.
REVISED_DATE	date(0)		Yes	The date/time (GMT) when this revision of this property was changed in the TME.
EXPIRATION_DATE	date(0)		Yes	The date/time (GMT) when this revision of this property expires (status changed to inactive) in the TME.
PREVIOUS_RID	integer(22)	FK	No	The RID for the previous revision to this property.
NEXT_RID	integer(22)	FK	No	The RID for the next revision to this property.

APPENDIX B

TME STAGE TABLES

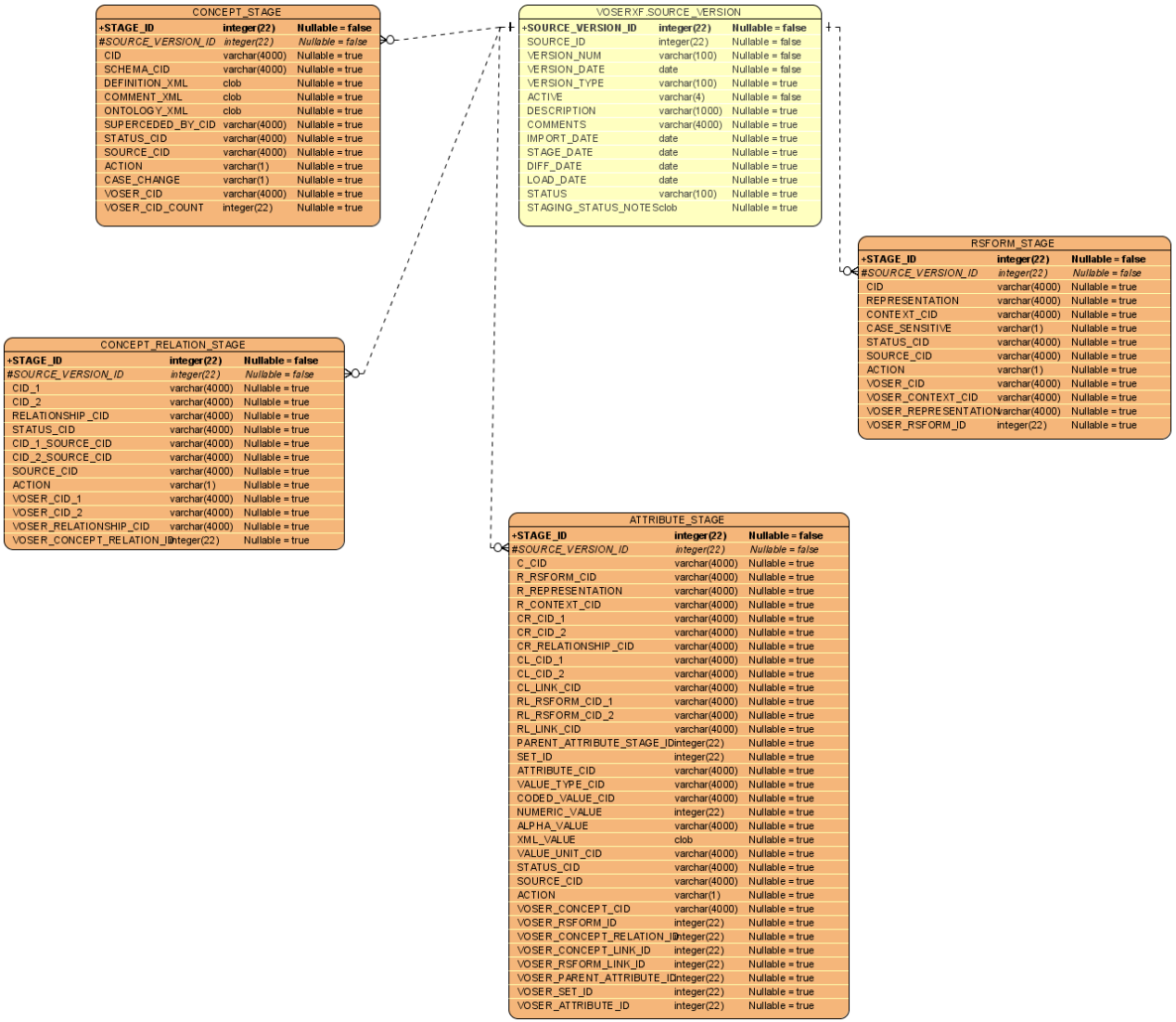


Figure 30: TME Stage Entity Relationship Diagram (ERD)

Table 31: TME Stage ERD Summary.

TABLE NAME	TABLE DESCRIPTION
ENTITY_STAGE	The ENTITY_STAGE table temporarily stores concepts from source vocabulary versions/releases during the ETL process.
TMEXF.SOURCE_VERSION	The TMEXF.SOURCE_VERSION table stores all versions of source code system releases, both standard and LIT, whether or not they are formally versioned/released by the source.
RSFORM_STAGE	The RSFORM_STAGE table temporarily stores representations from source code systems during the ETL process.
ENTITY_RELATION_STAGE	The ENTITY_RELATION_STAGE table temporarily stores entity relationships from source code system during the ETL process.
ENTITY_ATTRIBUTE_STAGE	The ENTITY_ATTRIBUTE_STAGE table temporarily stores entity attributes from source code system during the ETL process.

Table 32: ENTITY_STAGE Table Detail. The ENTITY_STAGE table temporarily stores concepts from source vocabulary versions/releases during the ETL process. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
STAGE_ID	integer(22)	PK	No	Identifies the record, and facilitates diffing and applications.
SOURCE_VERSION_ID	integer(22)	FK	No	References the source version/release that this staging is for.

Table 32: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
CID	varchar(4000)		Yes	Used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME. CIDs must be consistently derived from source data between versions/releases or the diffing process will not work correctly because it will not be able to tell if the source data already exists in TME.
SCHEMA_CID	varchar(4000)		Yes	
DEFINITION_XML	clob(0)		Yes	
COMMENT_XML	clob(0)		Yes	
ONTOLOGY_XML	clob(0)		Yes	
SUPERSEDED_BY_CID	varchar(4000)		Yes	
STATUS_CID	varchar(4000)		Yes	
SOURCE_CID	varchar(4000)		Yes	
ACTION	varchar(1)		Yes	Set by the diff process, the action tells the load process what it must do in TME (insert, update, delete, or nothing). Following the diff process, the user can review and change the automatically-determined actions, and must specify actions anywhere the system was unable to automatically determine what the action should be.
CASE_CHANGE	varchar(1)		Yes	Flags whether this CID exists in TME exactly as it is here, except in a different case (i.e., upper vs. lower, or different combination of mixed-case).
TME_CID	varchar(4000)		Yes	Used during the diff process, this identifies the CID in TME that matches this CID, if it exists in TME.
TME_CID_COUNT	integer(22)		Yes	Used during the diff process, this shows how many CIDs in TME exactly or case-insensitively match this CID.

Table 33: TMEXF.SOURCE_VERSION Table Detail. The TMEXF.SOURCE_VERSION table stores all versions of source code system releases, both standard and LIT, whether or not they are formally versioned/released by the source. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
SOURCE_VERSION_ID	integer(22)	PK	No	Identifies the version/release.
SOURCE_ID	integer(22)		No	References the source of the version/release (e.g., LOINC, RXNorm).
VERSION_NUM	varchar(100)		No	The version/release number provided by the source, if one is provided, or made up by the user, otherwise.
VERSION_DATE	date(0)		No	The version/release date provided by the source, if any, or the user, otherwise.
VERSION_TYPE	varchar(100)		Yes	Type of version/release (e.g., full, patch, etc.). This is provided by the source, derived from the version/release, provided by the user, or left null.
ACTIVE	varchar(4)		No	Used by applications.
DESCRIPTION	varchar(1000)		Yes	Description of the version/release. Provided by the source or by the user.
COMMENTS	varchar(4000)		Yes	Comments on the version/release. Provided by the source or by the user.
IMPORT_DATE	date(0)		Yes	Date/Time the version/release was imported (not loaded) and versioned.
STAGE_DATE	date(0)		Yes	Date/Time the version/release was transformed into the staging tables.
DIFF_DATE	date(0)		Yes	Date/Time the staged version/release was diffed against the source's namespace in TME.

Table 33: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
LOAD_DATE	date(0)		Yes	Date/Time the diffed version/release was loaded into the source's namespace in TME. The loading process is what creates revisions of core TME components (i.e., entities, entity relations, representations, entity links, representation links and attributes).
STATUS	varchar(100)		Yes	The current status of the started process if any (IMPORTING_STARTED, _COMPLETED, or IMPORTED_WITH_ERRORS; STAGING_STARTED, _COMPLETED, or _ERROR; DIFFING_STARTED, _COMPLETED, _ERROR, or _CONFLICTS; LOADING_STARTED, _COMPLETED, or _ERROR)
STAGING_STATUS_NOTES	clob(0)		Yes	Execution details of the ETL process.

Table 34: RSFORM_STAGE Table Detail. The RSFORM_STAGE table temporarily stores representations from source code systems during the ETL process. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
STAGE_ID	integer(22)	PK	No	Identifies the record, and facilitates diffing and applications.
SOURCE_VERSION_ID	integer(22)	FK	No	References the source version/release that this staging is for.
CID	varchar(4000)		Yes	Used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
REPRESENTATION	varchar(4000)		Yes	

Table 34: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
CONTEXT_CID	varchar(4000)		Yes	Used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
CASE_SENSITIVE	varchar(1)		Yes	
STATUS_CID	varchar(4000)		Yes	Used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
SOURCE_CID	varchar(4000)		Yes	Used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
ACTION	varchar(1)		Yes	Set by the diff process, the action tells the load process what it must do in TME (insert, update, delete, or nothing). Following the diff process, the user can review and change the automatically determined actions, and must specify actions anywhere the system was unable to automatically determine what the action should be.
TME_CID	varchar(4000)		Yes	Used during the diff process, this identifies the CID in TME that matches this CID, if it exists in TME.
TME_CONTEXT_CID	varchar(4000)		Yes	Used during the diff process, this identifies the context CID in TME that matches this context CID, if it exists in TME.
TME_REPRESENTATION	varchar(4000)		Yes	
TME_RSFORM_ID	integer(22)		Yes	Used during the ETL processes. If a match is found in TME, then the ID in TME is stored here; otherwise, a sequenced unique number is generated to serve as the ID of the row inserted into TME.

Table 35: ENTITY_RELATION_STAGE Table Detail. The ENTITY_RELATION_STAGE table temporarily stores entity relationships from source code system during the ETL process. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
STAGE_ID	integer(22)	PK	No	Identifies the record, and facilitates diffing and applications.
SOURCE_VERSION_ID	integer(22)	FK	No	References the source version/release that this staging is for.
CID_1	varchar(4000)		Yes	Used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
CID_2	varchar(4000)		Yes	Used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
RELATIONSHIP_CID	varchar(4000)		Yes	Used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
STATUS_CID	varchar(4000)		Yes	
CID_1_SOURCE_CID	varchar(4000)		Yes	
CID_2_SOURCE_CID	varchar(4000)		Yes	
SOURCE_CID	varchar(4000)		Yes	
ACTION	varchar(1)		Yes	Set by the diff process, the action tells the load process what it must do in TME (insert, update, delete, or nothing). Following the diff process, the user can review and change the automatically-determined actions, and must specify actions anywhere the system was unable to automatically determine what the action should be.
TME_CID_1	varchar(4000)		Yes	Used during the diff process, this identifies the CID in TME that matches this CID, if it exists in TME.

Table 35: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
TME_CID_2	varchar(4000)		Yes	Used during the diff process, this identifies the CID in TME that matches this CID, if it exists in TME.
TME_RELATIONSHIP_CID	varchar(4000)		Yes	Used during the diff process, this identifies the CID in TME that matches this CID, if it exists in TME.
TME_ENTITY_RELATION_ ID	integer(22)		Yes	Used during the diff and load processes. If a match is found in TME, then the ID in TME is stored here; otherwise, a sequenced unique number is generated to serve as the ID of the row inserted into TME.

Table 36: ENTITY_ATTRIBUTE_STAGE Table Detail. The ENTITY_ATTRIBUTE_STAGE table temporarily stores entity attributes from source code system during the ETL process. CIDs are used ETL process for mapping when a stable identifier is not available in the source code system. CIDs must be consistently derived from source data between versions/releases or the ETL process will not work correctly because it will not be able to tell if the source data already exists in TME. *Primary Key (PK)/Foreign Key (FK).

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
STAGE_ID	Integer (22)	PK	No	Identifies the record, and facilitates diffing and applications.
SOURCE_VERSION_ID	Integer (22)	FK	No	References the source version/release that this staging is for.
E_CID	Varchar (4000)		Yes	ENTITY.CID used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.

Table 36: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
R_RSFORM_CID	Varchar (4000)		Yes	RSFORM Rsform CID. Used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
R_REPRESENTATION	Varchar (4000)		Yes	RSFORM.REPRESENTATION.
R_CONTEXT_CID	Varchar (4000)		Yes	RSFORM.CONTEXT_TCID (CID). CID used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
ER_CID_1	Varchar (4000)		Yes	ENTITY_RELATION.TCID_1 (CID). CID used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
ER_CID_2	Varchar (4000)		Yes	ENTITY_RELATION.TCID_2 (CID). CID used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
ER_RELATIONSHIP_CID	Varchar (4000)		Yes	ENTITY_RELATION.RELATIONSHIP_TCID (CID). CID used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
EL_CID_1	Varchar (4000)		Yes	ENTITY_LINK.TCID_1 (CID). CID used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.

Table 36: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
EL_CID_2	Varchar (4000)		Yes	ENTITY_LINK.TCID_2 (CID). CID used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
EL_LINK_CID	varchar(4000)		Yes	ENTITY_LINK.LINK_TCID (CID). CID used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
RL_RSFORM_CID_1	varchar(4000)		Yes	Rsform link CID 1. Used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
RL_RSFORM_CID_2	varchar(4000)		Yes	Rsform link CID 2. Used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
RL_LINK_CID	varchar(4000)		Yes	Rsform link CID. Used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
PARENT_ATTRIBUTE_STAGE_ID	integer(22)		Yes	STAGE_ID of the parent attribute if an attribute of another attribute (created during the staging process); null otherwise.
SET_ID	integer(22)		Yes	Used during the diff and load processes. If a match is found in TME, then the ID in TME is stored here; otherwise, a sequenced unique number is generated to serve as the ID of the row inserted into TME.

Table 36: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
ATTRIBUTE_CID	varchar(4000)		Yes	Used instead of TCID because the source knows nothing about TCIDs, so there are no TCIDs in the version/release to be used to compare the version/release to TME.
VALUE_TYPE_CID	varchar(4000)		Yes	
CODED_VALUE_CID	varchar(4000)		Yes	
NUMERIC_VALUE	integer(22)		Yes	
ALPHA_VALUE	varchar(4000)		Yes	
XML_VALUE	clob(0)		Yes	
VALUE_UNIT_CID	varchar(4000)		Yes	
STATUS_CID	varchar(4000)		Yes	
SOURCE_CID	varchar(4000)		Yes	
ACTION	varchar(1)		Yes	Set by DIFFing - what action is necessary to perform during LOADing (Update, Insert, Delete, or nothing).
TME_ENTITY_CID	varchar(4000)		Yes	Used only during the DIFFing process of a entity attribute: the TCID of TME concept matched against the parent concept (if the parent concept still not existing in TME then TCID of the future parent concept which will be INSERTed).
TME_RSFORM_ID	integer(22)		Yes	Used during the ETL processes. If a match is found in TME, then the ID in TME is stored here; otherwise, a sequenced unique number is generated to serve as the ID of the row inserted into TME.
TME_ENTITY_RELATION_ID	integer(22)		Yes	Used during the ETL processes. If a match is found in TME, then the ID in TME is stored here; otherwise, a sequenced unique number is generated to serve as the ID of the row inserted into TME.

Table 36: Continued

FIELD NAME	DATA TYPE	PK/FK*	REQUIRED	FIELD DESCRIPTION
TME_ENTITY_LINK_ID	integer(22)		Yes	Used during ETL processes. If a match is found in TME, then the ID in TME is stored here; otherwise, a sequenced unique number is generated to serve as the ID of the row inserted into TME.
TME_RSFORM_LINK_ID	integer(22)		Yes	Used during the ETL processes. If a match is found in TME, then the ID in TME is stored here; otherwise, a sequenced unique number is generated to serve as the ID of the row inserted into TME.
TME_PARENT_ATTRIBUTE_ID	integer(22)		Yes	Used during ETL processes. If a match is found in TME, then the ID in TME is stored here; otherwise, a sequenced unique number is generated to serve as the ID of the row inserted into TME.
TME_SET_ID	integer(22)		Yes	Used during the ETL processes. If a match is found in TME, then the ID in TME is stored here; otherwise, a sequenced unique number is generated to serve as the ID of the row inserted into TME.
TME_ATTRIBUTE_ID	integer(22)		Yes	Used during the ETL processes. If a match is found in TME, then the ID in TME is stored here; otherwise, a sequenced unique number is generated to serve as the ID of the row inserted into TME.

APPENDIX C

TME SUPPORT FOR CTS FUNCTIONS

Table 37: TME Support for HL7 CTS Message Layer Runtime Functions. The following table is copied from the HL7 CTS Specification. It describe CTS methods related to message layer runtime functions.[65] *TME support for each method is indicated in an appended column.

Function	Inputs	Outputs	Description	TME Support Comment*
getServiceName		Service name	Return the name that was assigned to this service by the service provider.	Fully Supported: TFT Content Query. "Service" is a TFT entity and entity attributes and properties are used to store name, version, description, HL7 Release Version, CTS Version, and supported match algorithms information pertaining to the service.
getServiceVersion		Version identifier	Return the current version of the service software.	Fully Supported: TFT Content Query; Note service version is instantiated as an attribute value for "Service" TFT entity.
getServiceDescription		Service description	Return a description of the service function, authors, copyrights, etc.	Fully Supported: TFT Content; TFT "Service" entity metadata.

Table 37: Continued

Function	Inputs	Outputs	Description	TME Support Comment*
getHL7ReleaseVersion		Version identifier	Return the HL7 release version that is currently supported by this service.	Fully Supported: TFT Content; TFT "Service" entity metadata.
getCTSVersion		Major and minor version number	Return the CTS version that this service implements.	Fully Supported: TFT Content; TFT "Service" entity attribute.
getSupportedMatchAlgorithms		List of match algorithms	Return a list of string match algorithms implemented by this service.	Fully Supported: TFT Content; TFT "Service" entity attributes.
getSupportedVocabularyDomains	Match text and algorithm, time limit and size limit	List of vocabulary domain names	Return a list of the vocabulary domains matching the supplied match text that are recognized by this service.	Fully Supported: TME Design and TFT Content; TFT entity "All Domains." Children of "All Domains" in ENTITY_RELATION table supplies the list of supported vocabulary domains in the TFT.

Table 37: Continued

Function	Inputs	Outputs	Description	TME Support Comment*
validateCode	Name of the vocabulary domain, code to be validated, application context(realm), flag indicating whether to validate active concepts only and flag indicating whether to check both errors and warnings or just errors	List of errors and warnings.	Validate the coded attribute for the supplied vocabulary domain and context.	Fully Supported: TME Design and TFT Content. ENTITY table referenced to determine status, ENTITY_RELATION table referenced to validate participation in domain.

Table 37: Continued

Function	Inputs	Outputs	Description	TME Support Comment*
validateTranslation	Name of the vocabulary domain, coded attribute containing translation(s) to be validated, application context(realm), flag indicating whether to validate active concepts only and flag indicating whether to check both errors and warnings or just errors	List of errors and warnings.	Validate the CD translations, if any, for the supplied vocabulary domain and context.	Fully Supported: TME Design and TFT Content. Two options for validating translation. Runtime option references TFT and checks only the RSFORM table for concept equivalent translations. Option 2 involves querying the ENTITY_LINK table and specifying a link type.

Table 37: Continued

Function	Inputs	Outputs	Description	TME Support Comment*
translateCode	Name of the vocabulary domain, coded attribute to be translated, target code system, and target application context(realm)	Translation of the coded attribute.	Translate the supplied coded attribute into a form that uses the target code system or uses whatever code system is appropriate for the supplied context	Fully Supported: TME Design and TFT Content. Two options for translating a code. Runtime option references TFT and checks only the RSFORM table for concept equivalent translations. Option 2 involves querying the ENTITY_LINK table and specifying a link type.

Table 37: Continued

Function	Inputs	Outputs	Description	TME Support Comment*
fillInDetails	Coded attribute and target language code	Coded attribute value with details supplied.	Fill in the optional parts of the coded attribute such as the concept display name, the code system name and code system version.	Fully Supported: TME Design and TFT Content. TME supports additional more granular metadata that is not specifically called out in other CTS functions but can be provided through this function. This includes information such as USAGE_SCORE, entity link type, superseding entity, gender specific entities, representation links, representation status, case sensitive representations, etc.

Table 37: Continued

Function	Inputs	Outputs	Description	TME Support Comment*
subsumes	Parent coded attribute, child coded attribute	True / False	Determine whether the parent coded attribute subsumes (or implies) the child.	Fully Supported: TME Design and TFT Content. Query ENTITY_RELATION and validate that RELATION_TCID (relationship type) between two entities is a child of TFT "Subsumption Relationship Type."
areEquivalent	First coded attribute, second coded attribute	True / False	Determine whether the two coded attributes are 'equivalent'.	Fully Supported: TME Design and TFT Content. Query ENTITY_RELATION determine if two entities have TFT "Equivalent" relationship type.

Table 37: Continued

Function	Inputs	Outputs	Description	TME Support Comment*
lookupValueSetExpansion	Name of the vocabulary domain, application context(realm), language for expansion text, flag indicating whether to do a complete expansion of just one level, time limit and size limit	Hierarchical expansion of the value set associated with the domain in the supplied context	Return a hierarchical list of selectable concepts for the supplied vocabulary domain and context.	Fully Supported: TME Design and TFT Content. Query ENTITY_RELATION table specifying the domain entity and relationship type.

Table 37: Continued

Function	Inputs	Outputs	Description	TME Support Comment*
expandValueSetExpansionContext	Opaque expansion context returned from previous lookupValueSetExpansion or expandValueSetExpansionContext call	Further hierarchical expansion of the value set associated with the domain in the supplied context	Return further expansion on nested value set contents.	Fully Supported: TME Design and TFT Content. Query ENTITY_RELATION table specifying the domain entity and relationship type. TFT content includes a specific relationship type for transitive closure. So it is possible to query for immediate "parent" only or all ultimate parents of an entity.

Table 38: TME Support for HL7 CTS Vocabulary Layer Runtime Functions. The following table is copied from the HL7 CTS Specification. It describe CTS methods related to vocabulary layer runtime functions.[65] *TME support for each method is indicated in an appended column.

Function	Input	Output	Description	TME Support Comment*
getServiceName		Service name	Return the name that was assigned to this service by the service provider.	Fully Supported: TFT Content Query. "Service" is a TFT entity and entity attributes and properties are used to store name, version, description, HL7 Release Version, CTS Version, and supported match algorithms information pertaining to the service.
getServiceVersion		Version identifier	Return the current version of the service software.	Fully Supported: TFT Content Query; Note service version is instantiated as an attribute value for TFT "Service" entity.
getServiceDescription		Service description	Return a description of the service function, authors, copyrights, etc.	Fully Supported: TFT Content; TFT "Service" entity metadata.

Table 38: Continued

Function	Input	Output	Description	TME Support Comment*
getCTSVersion		Major and minor version number	Return the CTS version that this service implements.	Fully Supported: TFT Content; TFT "Service" entity attribute.
getSupportedCodeSystems	Time limit and size limit	List of code systems and versions supported by the service implementation.	Return the identifier, name and release versions of all code systems that are supported by the service.	Fully Supported: TFT Content; TFT "Supported Code Systems" entity is a domain that includes all supported code system root entities.
lookupCodeSystemInfo	Code system name or identifier	Description of the code system including name, id, description, version, supported languages, supported relations, supported properties, etc.	Return detailed information about a specific code system.	Fully Supported: TFT Content; Code system metadata are associated with code system root entities through ENTITY_ATTRIBUTE and PROPERTY tables.

Table 38: Continued

Function	Input	Output	Description	TME Support Comment*
isConceptIdValid	Code system identifier, concept code and flag indicating whether inactive concepts are considered valid	True / False	Determine whether concept code is currently valid in the specified code system	Fully Supported: TME Design. Validate that supplied code is a valid representation (RSFORM) on an active concept (ENTITY.STATUS_TCID) in the specified code system (ENTITY_RELATION).
lookupDesignation	Code system identifier, concept code and target language code	Designation text	Return the preferred designation for the concept code in the supplied language	Fully Supported: TME Design. Determined through RSFORM LANGUAGE_TCID, PREFERRED_REP, and CONTEXT_TCID.

Table 38: Continued

Function	Input	Output	Description	TME Support Comment*
areCodesRelated	Code system identifier, source concept code, target concept code, relationship code, relationship qualifiers, and flag indicating whether to use only directly related codes or the transitive closure of the relationship	True/False	Determine whether the named relationship exists between the source and target codes.	Fully Supported: TME Design. Determined through ENTITY_RELATION RELATION_TCID, TCID_1, and TCID_2.

Table 39: TME Support for HL7 CTS Code Mapping Functions. The following table is copied from the HL7 CTS Specification. It describe CTS methods related to code mapping functions.[65] *TME support for each method is indicated in an appended column.

Function	Input	Output	Description	TME Support Comment*
getServiceName		Service name	Return the name that was assigned to this service by the service provider.	Fully Supported: TFT Content Query. "Service" is a TFT entity and entity attributes and properties are used to store name, version, description, HL7 Release Version, CTS Version, and supported match algorithms information pertaining to the service.
getServiceVersion		Version identifier	Return the current version of the service software.	Fully Supported: TFT Content Query; Note service version is instantiated as an attribute value for "Service" TFT entity.
getServiceDescription		Service description	Return a description of the service function, authors, copyrights, etc.	Fully Supported: TFT Content; TFT "Service" entity metadata.
getCTSVersion		Major and minor version number	Return the CTS version that this service implements.	Fully Supported: TFT Content; TFT "Service" entity attribute.

Table 39: Continued

Function	Input	Output	Description	TME Support Comment*
getSupportedMaps		List of named sets consisting of from code system id, name and version, to code system id, name, and version and a mapping description	Return a list of mappings that are supported by this service.	Partially Supported: The TME links concepts in disparate code systems through the ENTITY_LINK table. Those links are "flattened" in the TFT for runtime deployment of content. Comprehensive point-to-point mapping is not the specific objective of the TME. Mappings are more purpose driven so it is more difficult to name and enumerate the mappings supported in the TME.
mapConceptCode	Source code system identifier and concept code, target code system identifier and name of mapping resource	Corresponding concept code in target system and quality indicator	Return the mapping of the supplied concept code from the source code system to the target code system using the named mapping resource.	Fully Supported: Information specific to the mapping resource and type of mapping is stored in the ENTITY_LINK table, the source and target codes are representations that are assigned contexts in the TFT. Either mapping links between source and target namespaces in the TME or designations in separate contexts in the TFT can be used to support this function.

Table 40: TME Support for HL7 CTS Message Layer Browsing Functions. The following table is copied from the HL7 CTS Specification. It describe CTS methods related to message layer browsing functions.[65] *TME support for each method is indicated in an appended column.

Function	Input	Output	Description	TME Support Comment*
getServiceName		Service name	Return the name that was assigned to this service by the service provider.	Fully Supported: TFT Content Query. "Service" is a TFT entity and entity attributes and properties are used to store name, version, description, HL7 Release Version, CTS Version, and supported match algorithms information pertaining to the service.
getServiceVersion		Version identifier	Return the current version of the service software.	Fully Supported: TFT Content Query; Note service version is instantiated as an attribute value for "Service" TFT entity.
getServiceDescription		Service description	Return a description of the service function, authors, copyrights, etc.	Fully Supported: TFT Content; TFT "Service" entity metadata.

Table 40: Continued

Function	Input	Output	Description	TME Support Comment*
getHL7ReleaseVersion		Version identifier	Return the HL7 release version that is currently supported by this service.	Fully Supported: TFT Content; TFT "Service" entity metadata.
getCTSVersion		Major and minor version number	Return the CTS version that this service implements.	Fully Supported: TFT Content; TFT "Service" entity attribute.
getSupportedMatchAlgorithms		List of string match algorithms implemented by the browser service		Fully Supported: TFT Content; TFT "Service" entity attributes.
getSupportedAttributes	Match text and algorithm, time limit and size limit	List of RIM attributes known to the browser	Returns a list of RIM attributes whose name matches the supplied match text that are known to the browser.	Partially Supported: TME can support this function from a design perspective, but supporting content that maps RIM attributes has not been instantiated in the TFT.

Table 40: Continued

Function	Input	Output	Description	TME Support Comment*
getSupportedVocabularyDomains	Match text and algorithm, time limit and size limit	List of vocabulary domains known to the browser	Returns a list of vocabulary domains whose name matches the supplied match text that are known to the browser.	Fully Supported: TME Design and TFT Content; TFT entity "All Domains." Children of "All Domains" in ENTITY_RELATION table supplies the list of supported vocabulary domains in the TFT.
getSupportedValueSets	Match text and algorithm, time limit and size limit	List of value sets known to the browser	Returns a list of value sets whose name matches the supplied match text that are known to the browser.	Fully Supported: TME Design and TFT Content; TFT entity "All Value Sets." Children of "All Value Sets" in ENTITY_RELATION table supplies the list of supported value sets in the TFT.
getSupportedCodeSystems	Match text and algorithm, time limit and size limit	List of code systems known to the browser	Returns a list of code systems whose name matches the supplied match text that are known to the browser.	Fully Supported: TFT Content; TFT "Supported Code Systems" entity is a domain that includes all supported code system root entities.

Table 40: Continued

Function	Input	Output	Description	TME Support Comment*
lookupVocabularyDomain	Name of vocabulary domain	Domain name, description, domains restricted by this domain, list of RIM attributes that use this domain, and list of value sets that represent this domain	Look up all of the information known about the supplied vocabulary domain	Partially Supported: TME can support this function from a design perspective, but supporting content that maps RIM attributes dependencies has not been instantiated in the TFT.
lookupValueSet	Value set name or identifier	Detailed value set description, including name, identifier, description, list of value sets used to construct the set, value sets that this set helps define, list of concept codes the value set references, etc.	Look up detailed information on a value set (including vocabulary domains, constructors, etc).	Fully Supported: Attributes, properties and relationships of the value set entity.

Table 40: Continued

Function	Input	Output	Description	TME Support Comment*
lookupCodeSystem	Code system name or identifier	Name, id, copyright, release and registration information	Look up details on a code system	Fully Supported: Attributes, properties and relationships of the code system root concept entity.
lookupValueSetForDomain	Name of vocabulary domain and application context(realm)	Name and id of the value set used for this vocabulary domain	Return the identifier of the value set that would be used for the vocabulary in the supplied context (if any).	Fully Supported: Representations and properties of the value set entity.
isCodeInValueSet	Value set name or identifier, code system identifier and concept code, and indicator whether to include the "head code" as part of the value set	True/False	Determine whether the supplied concept code is a valid value in the supplied value set	Fully Supported: Entity relationship link between entity with specified concept code and value set entity.

Table 41: TME Support for HL7 CTS Vocabulary Layer Browsing Functions. The following table is copied from the HL7 CTS Specification. It describe CTS methods related to vocabulary layer browsing functions.[65] *TME support for each method is indicated in an appended column.

Function	Input	Output	Description	TME Support Comment*
getServiceName		Service name	Return the name that was assigned to this service by the service provider.	Fully Supported: TFT Content Query. "Service" is a TFT entity and entity attributes and properties are used to store name, version, description, HL7 Release Version, CTS Version, and supported match algorithms information pertaining to the service.
getServiceVersion		Version identifier	Return the current version of the service software.	Fully Supported: TFT Content Query; Note service version is instantiated as an attribute value for "Service" TFT entity.

Table 41: Continued

Function	Input	Output	Description	TME Support Comment*
getServiceDescription		Service description	Return a description of the service function, authors, copyrights, etc.	Fully Supported: TFT Content; TFT "Service" entity metadata.
getCTSVersion		Major and minor version number	Return the CTS version that this service implements.	Fully Supported: TFT Content; TFT "Service" entity attribute.
getSupportedMatchAlgorithms		List of string match algorithms implemented by the browser service		Fully Supported: TFT Content; TFT "Service" entity attributes.
getSupportedCodeSystems	Time limit and size limit	List of supported code systems and their descriptions		Fully Supported: TFT Content; TFT "Supported Code Systems" entity is a domain that includes all supported code system root entities.

Table 41: Continued

Function	Input	Output	Description	TME Support Comment*
lookupConceptCodesByDesignation	Code system identifier, match text and algorithm, target language code, flag indicating whether nonactive concepts should be retrieved, time limit and size limit	List of code system identifiers and concept codes.	Return a list of concept codes that have designations that match the supplied match string in the supplied language, if any.	Fully Supported: Return code representations on all entities with the supplied designation in the RSFORM table.
lookupConceptCodesByProperty	Code system identifier, match text and algorithm, target language code, flag indicating whether nonactive concepts whould be retrieved, optional list of property mime types, time limit and size limit	List of code system id / concept codes.	Return a list of concept codes that have properties that meet the supplied criteria	Fully Supported: Return code representations on all entities with the supplied properties in the ENTITY_ATTRIBUTE or PROPERTY tables.

Table 41: Continued

Function	Input	Output	Description	TME Support Comment*
lookupCompleteCodedConcept	Code system identifier and concept code	Everything that is known about the concept (designations, properties, relationships, etc.)	Return a complete description of the supplied concept code	Fully Supported: Return entity relationships, links, attributes, properties, and representations.
lookupDesignations	Code system id and concept code, match text and algorithm, target language	List of designations	Return all designations for the supplied concept code that match the supplied criteria.	Fully Supported: Return all of representations associated with an entity in the RSFORM table.
lookupProperties	Code system id and concept code, match text and algorithm, list of property codes to search, list of mime types to match and target language code	List of concept properties (property code, value, language, mime type)	Return the properties of the given code system id / concept code that match the supplied criteria.	Fully Supported: Return properties/attributes for the specified entity from the ENTITY_ATTRIBUTE and PROPERTY tables.

Table 41: Continued

Function	Input	Output	Description	TME Support Comment*
lookupCodeExpansion	Code system id and concept code, relationship code, relationship direction indicator, target languag code, size limit and time limit	Hierarchical code expansion list	Recursively list the concept codes that are related to the supplied concept, including the preferred designation for the codes.	Fully Supported: Returns relationships in the ENTITY_RELATION table.

REFERENCES

1. Millenson ML. *Demanding medical excellence : doctors and accountability in the information age.* Chicago, Ill.: University of Chicago Press; 1997.

2. Agency for Healthcare Research & Quality (AHRQ). *National Healthcare Quality Report.* 2008. AHRQ Publication No. 09-0001.

3. Thomas EJ, Studdert DM, Burstin HR, et al. Incidence and types of adverse events and negligent care in Utah and Colorado. *Med Care.* Mar 2000;38(3):261-271.

4. Brennan TA, Leape LL, Laird NM, et al. Incidence of adverse events and negligence in hospitalized patients. Results of the Harvard Medical Practice Study I. *N Engl J Med.* Feb 7 1991;324(6):370-376.

5. American Hospital Association, Health Forum (Organization). Hospital statistics. Chicago, Ill.: Healthcare InfoSource; 1998.

6. Martin AB, Lassman D, Washington B, Catlin A. Growth In US health spending remained slow in 2010; health share of gross domestic product was unchanged from 2009. *Health Affairs.* 2012;31(1):208-219.

7. Shojania KG, Duncan BW, McDonald KM, Wachter RM, Markowitz AJ. Making health care safer: a critical analysis of patient safety practices. *Evid Rep Technol Assess (Summ).* 2001(43):i-x, 1-668.

8. Shekelle PG, Morton SC, Keeler EB. Costs and benefits of health information technology. *Evid Rep Technol Assess (Full Rep).* Apr 2006(132):1-71.

9. Bakken S, Currie LM, Lee NJ, Roberts WD, Collins SA, Cimino JJ. Integrating evidence into clinical information systems for nursing decision support. *International journal of medical informatics.* Jun 2008;77(6):413-420.

10. Mookencherry S. Eight reasons payer interoperability and data sharing are essential in ACOs. Interoperability standards could be a prerequisite to measuring care. *Health Manag Technol.* Jan 2012;33(1):16-19.

11. Eddy DM. Clinical decision making: from theory to practice. Anatomy of a decision. *JAMA.* Jan 19 1990;263(3):441-443.

12. Anusuya MA, Katti SK. Superficial analogies and differences between the human brain and the computer. *IJCSNS.* July 2010 2010;10(7).

13. Shapiro AR. Taming variability in free text: application to health surveillance. *MMWR Morb Mortal Wkly Rep.* Sep 24 2004;53 Suppl:95-100.

14. IEEE Computer Society. Standards Coordinating Committee. *IEEE standard computer dictionary : a compilation of IEEE standard computer glossaries, 610.* New York, NY, USA: Institute of Electrical and Electronics Engineers; 1990.

15. International Organization for Standardization (ISO). Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model (ISO/IEC 7498-1). *International Organization for Standards.* 1996.

16. National Committee on Vital and Health Statistics (NCVHS). *Report to the Secretary of the U.S. Department of Health and Human Services on uniform data standards for patient medical record information.* July 6, 2000 2000.

17. Cimino JJ, Elhanan G, Zeng Q. Supporting infobuttons with terminological knowledge. *AMIA Annual Fall Symposium.* 1997:528-532.

18. Lau LM, Lam SH, Shakib SC. What true interoperability means. Paper presented at 12th World Congress on Health (Medical) Informatics; 2007.

19. Stewart BA, Fernandes S, Rodriguez-Huertas E, Landzberg M. A preliminary look at duplicate testing associated with lack of electronic health record interoperability for transferred patients. *JAMIA.* 2010;17(3):341-344.

20. Amato-Gauci A, Ammon A. The surveillance of communicable diseases in the European Union--a long-term strategy (2008-2013). *Euro surveillance : bulletin europeen sur les maladies transmissibles = European communicable disease bulletin.* Jun 26 2008;13(26).

21. Alfreds ST, Witter DM. *The impact of electronic Health Information Exchange (HIE) services in Maine: avoidable service and productivity savings estimates related to HealthInfoNet services.* The Maine HealthInfoNet Stakeholder Group Maine Quality Forum;2008.

22. Walker J, Pan E, Johnston D, Adler-Milstein J, Bates DW, Middleton B. The value of health care information exchange and interoperability. *Health Affairs.* Jan-Jun 2005;Suppl Web Exclusives:W5-10-W15-18.

23. H.R. 1--111th Congress: American Recovery and Reinvestment Act of 2009. 2009.

24. Office of the National Coordinator for Health Information Technology. About ONC. 2012; http://healthit.hhs.gov/portal/server.pt/community/healthit_hhs_gov__onc/1200. Accessed July 9, 2012.

25. Rocha RA, Huff SM, Haug PJ, Warner HR. Designing a controlled medical vocabulary server: the VOSER project. *Comput Biomed Res.* Dec 1994;27(6):472-507.

26. Cimino JJ. Desiderata for controlled medical vocabularies in the twenty-first century. *Methods Inf Med.* Nov 1998;37(4-5):394-403.

27. Apelon Inc. Apelon home page. 2012; http://www.apelon.com. Accessed July 9, 2012.

28. Apelon Inc. Apelon DTS. 2012; http://apelon-dts.sourceforge.net. Accessed July 9, 2012.

29. Lau LM, Shakib SC. Towards data interoperability: practical Issues in terminology implementation and mapping. Clinical Vocabulary Mapping Methods Institute, 77th AHIMA Convention and Exhibit; 2005.

30. Columbia-Presbyterian, New York-Cornell. Medical Entities Dictionary. 2013; http://med.dmi.columbia.edu/. Accessed April 17, 2013.

31. Cimino JJ. From data to knowledge through concept-oriented terminologies experience with the Medical Entities Dictionary. *JAMIA.* 2000;7(3):288-297.

32. Health Language Inc. HLI Language Engine. 2013; http://www.healthlanguage.com/products/language-engine.html. Accessed February 12, 2013.

33. Clinical Architecture. Clinical Architecture home page. 2013; http://www.clinicalarchitecture.com/solutions/symedical-server/. Accessed February 12, 2013.

34. National Cancer Institute. LexGrid. 2013; https://wiki.nci.nih.gov/display/LexEVS/LexGrid. Accessed April 17, 2013.

35. National Institutes of Health (U.S.). BiomedGT wiki. 2013; https://wiki.nci.nih.gov/display/EVS/BiomedGT+Wiki. Accessed April 17, 2013.

36. Health Level Seven International. HL7 home page. 2013; http://www.hl7.org/. Accessed April 17, 2013.

37. McDonald CJ, Overhage JM, Dexter P, Takesue B, Suico JG. What is done, what is needed and what is realistic to expect from medical informatics standards. *International journal of medical informatics.* Feb 1998;48(1-3):5-12.

38. Strang N, Cucherat M, Boissel JP. Which coding system for therapeutic information in evidence-based medicine. *Computer methods and programs in biomedicine.* 2002;68(1):73-85.

39. Smith B. From concepts to clinical reality: an essay on the benchmarking of biomedical terminologies. *J Biomed Inform.* Jun 2006;39(3):288-298.

40. Cimino JJ. In defense of the desiderata. *J Biomed Inform.* Jun 2006;39(3):299-306.

41. Canada Health Infoway. The Electronic Health Record Solution (EHRS) blueprint version 2. 2006.

42. Rosenbloom ST, Miller RA, Johnson KB, Elkin PL, Brown SH. A model for evaluating interface terminologies. *JAMIA.* 2007;15(1):65-76.

43. American Medical Association. *CPT 2003: Current Procedural Terminology.* Chicago2002.

44. Abraham M, Ahlman JT, Anderson C, Boudreau AJ, Connelly J. *CPT 2012 (CPT / Current Procedural Terminology (Professional Edition)).* Chicago: American Medical Association Press; 2011.

45. The International Health Terminology Standards Development Organisation (IHTSDO). *SNOMED Clinical Terms® technical reference guide.* 2008.

46. Regenstrief Institute for Health Care. LOINC and RELMA. Indianapolis, Ind.: Regenstrief Institute; 1995: http://www.regenstrief.org/loinc/#releasenotes.

47. Huff SM, Rocha RA, McDonald CJ, et al. Development of the Logical Observation Identifier Names and Codes (LOINC) Vocabulary. *JAMIA.* 1998;5:276–292.

48. National Library of Medicine (U.S.). RxNorm. Bethesda, MD: U.S. National Library of Medicine, National Institutes of Health, Health & Human Services; 2004: http://www.nlm.nih.gov/research/umls/rxnorm/. Accessed December 15, 2004.

49. National Library of Medicine (U.S.). UMLS reference manual. Bethesda, MD: U.S. National Library of Medicine, National Institutes of Health; 2010: http://www.ncbi.nlm.nih.gov/books/NBK9676. Accessed February 24, 2010.

50. Buck CJ. *2012 ICD-9-CM for hospitals, volumes 1, 2 and 3 professional edition.* Elsevier Health Sciences; 2011.

51. Leon-Chisen N. *ICD-10-CM and ICD-10-PCS Coding Handbook, With Answers, 2012 Revised Edition.* AHA Press.; 2011.

52. Nelson DA. Why does medicine need standards? *Medical Computing Today*. 1997. http://medicalcomputing.org/archives/0astandwhy.php. Accessed April 17, 2013.

53. Schulz S, Boeker M, Stenzhorn H, Niggemann J. Granularity issues in the alignment of upper ontologies. 20090313 DCOM- 20090814 2009(0026-1270 (Print)).

54. Rector AL, Nowlan WA. The GALEN project. 19950419 DCOM-19950419 1994(0169-2607 (Print)).

55. Shakib SC, Knight E, Lau LM. Challenges of mapping microbiology data to a common data dictionary. Poster presented at AMIA Annual Symposium; 2000.

56. National Drug Data File Plus. In: First DataBank, ed: National Libarary of Medicine; 2010.

57. Current Procedural Terminology (CPT). In: Association AM, ed: American Medical Association; 2003.

58. Oliver DE, Shahar Y, Shortliffe EH, Musen MA. Representation of change in controlled medical terminologies. *Artificial intelligence in medicine.* Jan 1999;15(1):53-76.

59. Hole WT, Carlsen BA, Tuttle MS, et al. Achieving "source transparency" in the UMLS Metathesaurus. *Studies in health technology and informatics.* 2004;107(Pt 1):371-375.

60. Lau LM, Banning PD, Monson K, Knight E, Wilson PS, Shakib SC. Mapping Department of Defense laboratory results to Logical Observation Identifiers Names and Codes (LOINC®). Paper presented at AMIA Annual Symposium; 2005.

61. Shakib SC, Lau LM. Making standard terminologies operational in the electronic health record. Paper presented at 12th World Congress on Health (Medical) Informatics; 2007.

62. Huff SM, Rocha RA, Solbrig HR, Barnes MW, Schrank SP, Smith M. Linking a medical vocabulary to a clinical data model using Abstract Syntax Notation 1. *Methods Inf Med.* 1998;37:440-452.

63. Shakib SC, Endo J, Lau LM, Karitis JW. Development of an integrated dentistry information model. Poster presented at AMIA Annual Symposium; 2001.

64. National Library of Medicine (U.S.). SNOMED Clinical Terms (SNOMED CT). Bethesda, MD: U.S. National Library of Medicine, National Institutes of Health, Health & Human Services; 2009: http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html. Accessed October 20, 2009.

65. Solbrig H. HL7 Common Terminology Services. HL7 Version 3 Standard. *Health Level Seven, Inc.* 2003.

66. Object Management Group Inc. Lexicon Query Service specification. Object Management Group, Inc.; 2000.

67. National Library of Medicine (U.S.). UMLS knowledge sources. Bethesda, MD: U.S. National Library of Medicine, National Institutes of Health, Health & Human Services; 1999: http://www.nlm.nih.gov/research/umls/knowledge_sources/index.html. Accessed April 2, 2008.

68. McDonald C, Huff S, Vreeman DJ, Mercer K, Hernandez JA. Logical Observation Identifiers Names and Codes (LOINC®). 1995.

69. Friedman C. Towards a comprehensive medical language processing system: methods and issues. AMIA Annual Symposium; 1997.

70. Chen PP-S. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS).* 1976;1(1):9-36.

71. Schadow G, McDonald CJ. The Unified Code for Units of Measure (UCUM). *Regenstrief Institute and Indiana University School of Informatics.* 2005.

72. Halevy AY, Ashish N, Bitton D, et al. Enterprise information integration: successes, challenges and controversies. ACM SIGMOD International Conference on Management of Data; 2005.

73. Lau LM, Johnson K, Banning P, et al., Inventors; Google Patents, assignee. Managing relationships between unique concepts in a database. 2001.

74. Rocha RA, Rocha BH, Huff SM. Automated translation between medical vocabularies using a frame-based interlingua. Paper presented at Annual Symposium on Computer Application in Medical Care; 1993.

75. Nachimuthu SK, Lau LM. Applying hybrid algorithms for text matching to automated biomedical vocabulary mapping. *AMIA Annual Symposium.* 2005;2005:555.

76. Browne AC, Divita G, Aronson AR, McCray AT. UMLS language and vocabulary tools: AMIA 2003 open source expo. AMIA Annual Symposium; 2003.

77. Zollo KA, Huff SM. Automated mapping of observation codes using extensional definitions. *JAMIA.* 2000;7(6):586-592.

78. Shakib SC, Che C, Lau LM. Using knowledge rules for pharmacy mapping. Poster presented at AMIA Annual Symposium; 2006.

79. Che C, Monson K, Poon KB, Shakib SC, Lau LM. Managing vocabulary mapping services. AMIA Annual Symposium; 2005.

80. Fensel D. *Ontologies.* Springer Berlin Heidelberg; 2001.

81. Gennari JH, Musen MA, Fergerson RW, et al. The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies.* 2003;58(1):89-123.

82. Mayo Clinic. Common Terminology Services 2. 2013; http://informatics.mayo.edu/cts2/index.php/Main_Page. Accessed April 7, 2013.

83. Lee DH, Lau FY, Quan H. A method for encoding clinical datasets with SNOMED CT. *BMC Medical Informatics and Decision Making.* 2010;10(1):53.

84. Poon KB, Che C, Monson K, Shakib SC, Lau LM. The evolution of tools and processes for data mapping. ACMI Senior Member Presentations presented at AMIA Annual Symposium; 2005.

85. Shakib SC, Poon KB, Lau LM. Tools and processes to improve data mapping accuracy and reliability. Poster presented at 11th World Congress on Medical Informatics; 2004.

86. Poon KB, Shakib SC, Lau LM. A quality assurance browser tool for tracking vocabulary mapping. Poster presented at 11th World Congress on Medical Informatics; 2004.

87. Shakib SC, Knight E, Endo J, Lau LM. An application to integrate the logistical and technical aspects of data dictionary support to multiple healthcare systems. Poster presented at AMIA Annual Symposium; 2002.

88. Brickley D, Guha RV. Resource Description Framework (RDF) schema specification 1.0: W3C candidate recommendation 27 March 2000. 2000.