HL7_LM_TERM_KB_R1_I1_2021MAY

# HL7 Logical Model: Standardized Terminology Knowledgebase, Release 1

May 2021

**HL7 Informative Ballot**

**Sponsored by:**

**Vocabulary Work Group**

| **Terminology** | **Owner/Contact** |
|---|---|
| Current Procedures Terminology (CPT) code set | American Medical Association https://www.ama-assn.org/practice-management/cpt-licensing |
| SNOMED CT® | SNOMED International http://www.snomed.org/snomed-ct/get-snomed-ct or info@ihtsdo.org |
| Logical Observation Identifiers Names & Codes (LOINC®) | Regenstrief Institute |
| International Classification of Diseases (ICD) codes | World Health Organization (WHO) |
| NUCC Health Care Provider Taxonomy code set | American Medical Association. Please see www.nucc.org. AMA licensing contact: 312-464-5022 (AMA IP services) |

# Table of Contents

# List of Figures

# Purpose

This specification describes the requirements needed to manage terminology produced by a variety of organizations across the healthcare enterprise. The healthcare enterprise needs an extensive terminology foundation and the ability to implement extensions to support diverse informatics systems.

This specification is intended to support standard terminology modules, value sets, and coding systems as well as local terms and equivalence mappings.

A standard-based Terminology Knowledge Architecture (Tinkar) specification is necessary to support the operation of a variety of systems intended to deliver knowledge management for terminology to vendors, providers, and standards-development organizations, like Health Level Seven International (HL7).

# Acknowledgements

The project team would like to acknowledge the leadership and guidance from:

- **Keith E. Campbell, MD, PhD** - Director of Informatics Architecture, Knowledge Based Systems, Veterans Health Administration
- **Carol Macumber, MS, FAMIA** - Health Level Seven International (HL7) Vocabulary Work Group co-chair and HL7 Terminology Authority vice chair
- **Penni Hernandez, RN, ND, APN** - Nurse, Associate Director, Clinical Data Representation, Knowledge Based Systems, Veterans Health Administration
- **Jonathan Nebeker, MD, MS** - Deputy Chief Medical Informatics Officer, Veterans Health Administration
- **Steven Brown, MD, MS** - Director of Knowledge Based Systems, Veterans Health Administration

The Tinkar ballot received contributions and assistance from the following individuals:

| | |
|---|---|
| Keith E. Campbell [1] | Jessica Bota [2] |
| Raja A. Cholan [3] | Richard Esmond [4] |
| H. Alpha Garrett Jr. [5] | Jay Lyle [5] |
| Galen Mulrooney [5] | Akila Namasivayam [3] |
| Andrew Sills [3] | Ioana Singureanu [1] |
| Walter Sujansky[6] | Karen Thullner [4] |
| Tim Williams [3] | Kurt W. Allen [4] |

[1]Veterans Health Administration; [2]HL7 Vocabulary Work Group; [3]Deloitte Consulting LLP; [4]PenRad Technologies Inc;[5]J P Systems, Inc; [6]Sujansky & Associates, LLC

This work was primarily funded by the Veterans Health Administration Clinical Informatics and Data Management Office. This work would not have been possible without the support of Jonathan Nebeker and Steven Brown.

# Glossary

| | |
|---|---|
| Terminology | A set of concept identifiers, descriptors, and relationships used to capture domain information in an information system. |
| Standard terminology | A terminology developed and published for use across organizational boundaries. |
| Enterprise terminology | The set of terminology assets that are specific to and maintained by a healthcare organization, including, locations and services and their relationships to standard terminologies. |
| Tinkar | Terminology Knowledge Architecture (Tinkar) is intended to provide an architecture that delivers integrated terminology to the enterprise and its information systems. Tinkar addresses the differences in management and structure across reference terminology, and with local concepts and code lists/value sets. |
| Logical Model | A model expressed independently of a particular implementation technology. |
| Tinkar Reference Model | A logical model that describes the standardized model for terminology and change management. |
| Coordinate | A dimension of context in which a user wishes to view terminology assets. |
| STAMP – (status, time, author, module, path) | The precise version coordinates that govern granular change control for Tinkar assets. |
| Path | A set of assets under version control that can be managed distinctly from other assets. Paths “branch” from other paths when established, and can be “merged” with other paths as well. |
| Branch | "Branch" as commonly used in version control, implies a tree structure, where a version configuration is allowed to only have one origin. In many instances, supported version configuration has more than one origin. For example, a mapping configuration may have an origin of T1 on System A, and T2 on System B (based on the release schedules of Systems A and B). A Path, which is more general, allows multiple origins of a version configuration. |
| Publish | To move a set of assets to a “published” branch and support its export to an interoperable payload technology. Publication implies that the publisher has exercised some editorial judgment and warrants the validity and consistency of the publication. |
| Export | To serialize a set of content to an interoperable payload technology. Export does not carry the same quality expectations as publication. |
| Import | To replicate an exported set of content into a repository. This activity implies the implementation of rules and constraints asserted by both native and imported content. |
| Configuration requirements | Functionality required to make a Tinkar repository operational. |
| UML (Unified Modeling Language) | The Object Management Group (OMG) Unified Modeling Language (UML) specification defines a graphical language for specifying and documenting the artifacts of systems. |

| | |
|---|---|
| Minimally Required Content | The set of minimal content a Tinkar repository must implement to support operations. A Tinkar implementation must be furnished with the following content: (a) One root concept; (b) One module dependency graph; (c) Infrastructure concepts a.k.a Tinkar Model Concepts to support the core patterns listed above; (d) Import rules to support import of standard terminologies. |
| Tinkar Model Concepts (a.k.a Infrastructure concepts) | Minimally required content that are concepts. |
| Separation of Concerns | A design principle that allows a complete system to be subdivided into distinct sections or components with well-defined functionality and dependencies. This approach allows individual sections to be reused, as well as worked on and updated independently to address new requirements and use cases. |
| Foundation Layer of Knowledge Architecture | The Foundation layer provides the common elements of interoperability such as object identity, versioning, modularity, and knowledge representation. It includes the foundation and building blocks of the common model and how the modules of the architecture are tightly version controlled over time. |
| Terminology Knowledge Layer of Knowledge Architecture | The Terminology Knowledge Layer provides structured sets of medical terms and codes that define concepts of interest. This includes descriptions, dialects, language, and semantic hierarchy; SNOMED CT®, LOINC®, and RxNorm are part of this layer. |
| Solor | A project sponsored by the Department of Veterans Affairs (VA) and Logica Health that represents and brings together terminology standards in a single model that can encompass any customized content. Solor allows informaticists and developers to convert user-supplied terminologies into a single model using open-source software to produce Solor content. |
| Komet (Knowledge Management Environment) | Komet is an example implementation of the Tinkar model, and shows how terminologies like SNOMED CT®, LOINC®, RxNorm shall be represented. Using the Komet browser users can view and search concepts, view concept details, and navigate hierarchies, among many other functions. |
| UUID (Universally Unique Identifier) | A 128-bit number used to identify information in computer systems. |
| FHIR (Fast Healthcare Interoperability Resources) | A standard describing data formats and elements and an application programming interface for exchanging electronic health records. |
| Capability Maturity Model | A development model that can be viewed as a set of structured levels that describe how well the behaviors, practices, and processes of an organization can reliably and sustainably produce required outcomes. There are five levels defined along the continuum of the capability maturity model and the model provides a theoretical continuum along which process maturity can be developed incrementally from one level to the next. |

# 1. Tinkar Overview

Over the past decades, biomedical terminologies have increasingly been recognized as key resources for knowledge management, data integration, and decision support [1]. Acceleration and development of Electronic Health Record (EHR) systems has precipitated the emergence of "standard terminologies" and their widespread adoption in the clinical community. These include Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT®), the Logical Observation Identifiers, Names, and Codes (LOINC®) and RxNorm. The availability of these clinical terminologies through the terminology services of FHIR is facilitating their usage in support of interoperability in healthcare.

Interoperability requires standardized semantics based on reference terminology provided by standards development organizations, professional organizations, or government agencies. These organizations publish their content with the intention of licensing it to health IT vendors, providers, and research organizations. In the U.S., the core clinical reference terminology is based on SNOMED CT®, LOINC®, and RxNorm. Healthcare organizations must adopt and integrate subsets or modules of various reference terminology and manage references, dependencies, versions, and releases. It is important for the integrity of medical records that the change history of concepts and value sets can be managed and tracked to allow the exchange of either current or retrospective health records. Therefore, enterprise terminology requires integrated terminology using a common representation and management.

Despite the need to use standard terminologies in a highly integrated way, there is no standard representation across SNOMED CT®, LOINC®, RxNorm, etc. Some partnerships have been created among development teams to facilitate interoperability and minimize duplication of effort. Further integration has been proposed but will require additional resources to bring these terminologies closer together. However, while this evolution leads to greater compatibility and interoperability, integration of SNOMED CT®, LOINC®, and RxNorm is non-trivial as these terminologies use different formalisms and tools for their representation. Various terminologies have different semantics, models, release cycles, and versioning mechanisms [1]. While there is recognition that terminologies are not standardized at the exchange level, there is no consensus about harmonized next steps to solve the challenges.

This document focuses on the need for – and logical specification of – a Terminology Knowledge Architecture (Tinkar). The Tinkar Reference Model is a logical model that describes the standardized model for terminology and change management. Tinkar provides an architecture that delivers integrated terminology to the enterprise and its information systems. In doing so, it addresses the differences in management and structure across reference terminology, local concepts, and code lists/value sets.

The capability maturity model is a development model and can be viewed as a set of structured levels that describe how well the behaviors, practices, and processes of an organization can reliably and sustainably produce required outcomes. There are five levels defined along the continuum of the capability maturity model (see below). The model provides a theoretical continuum along which process maturity can be developed incrementally from one level to the next. An implementation of the Tinkar Specification can provide a single representation for all terminologies required in the U.S. and other countries, while also providing a better foundation for maturing change management models as described by the capability maturity model [2]:

1. **Initial** – Tinkar aims to reduce and eliminate challenges with management of changes to terminology being unpredictable, poorly controlled, and reactive.

2. **Repeatable** – Tinkar provides the foundation for robust configuration management and quality assurance for terminologies.

3. **Defined** – Tinkar allows terminologies to have standardized update and extension processes.

4. **Managed** – Tinkar represents updates and changes to terminologies so that the changes can be measured and controlled.

5. **Optimizing** – Tinkar aims to allow multiple stakeholders to apply and retrieve changes to shared terminology content with equivalent and harmonized results.

Tinkar aims to adhere to the following statement from a publication about developments in clinical terminologies in the 2018 Yearbook of Medical Informatics [1]: “The benefits of the integrated terminologies in terms of homogenous semantics and inherent interoperability should, however, outweigh the complexity added to the system.” This specification provides the foundation of a knowledge architecture that is intended to integrate reference terminology from distributors (e.g., SNOMED CT®, LOINC®, RxNorm) with local concepts to support interoperable information semantics across the enterprise.

## 1.1. Motivation: Why Tinkar?

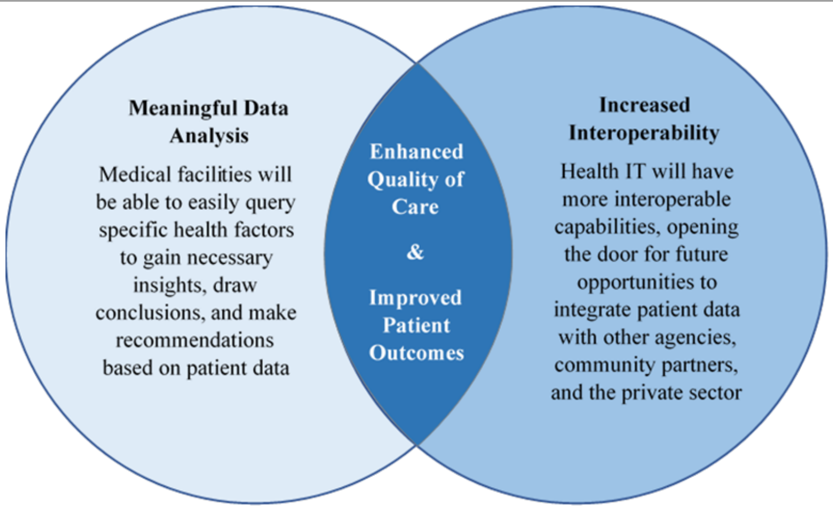
Information systems that are used across the healthcare enterprise record and manage clinical data using clinical statements and clinical terminologies in non-standardized ways. Interoperability specifications aim to require terminology bindings to concepts, code systems, and reusable value sets. Currently, there is variation in clinical data exchange across the enterprise, as current payloads and clinical statements use inconsistent and highly variable enterprise terminologies. The management of the concepts, code systems, and value sets is non-trivial because developers, implementers, and end users are forced to manage “unnecessary complexity.” For example:

- Representation of medications: RxNorm codes overlap with CVX codes. Investigational vaccines from the FDA are not represented in either RxNorm or CVX or SNOMED CT®.

- Representation of COVID-19 result codes are inconsistent and are not equivalent (e.g., detected, undetected, positive, negative, etc.).

As a result of these complexities, there are many ways to say the same thing using standard terminologies and standard formats. The Institute of Medicine report, *Health IT and Patient Safety: Building Safer Systems for Better Care*, highlighted the unintended consequences of health IT-induced harm that can result in serious injury and death due to dosing errors, failing to detect serious illnesses, and delaying treatment due to poor human-computer interactions, confusing clinical terminology, or unreliable data quality [3]. Despite the widespread understanding of the importance of the quality of clinical data, there is currently a lack of integration and management of clinical terminologies across the healthcare enterprise.

Tinkar intends to support integration of clinical terminology and local concepts to support increased data quality for interoperable clinical information. Quality clinical data enables healthcare systems across the enterprise to conduct robust and meaningful data analysis and increase overall interoperability, which ultimately enhances quality of care across all medical facilities.


Meaningful Data Analysis
Medical facilities will be able to easily query specific health factors to gain necessary insights, draw conclusions, and make recommendations based on patient data
Enhanced Quality of Care
&
Improved Patient Outcomes
Increased Interoperability
Health IT will have more interoperable capabilities, opening the door for future opportunities to integrate patient data with other agencies, community partners, and the private sector

## 1.2. The Problem Tinkar Addresses

The following four high level potential deficiencies related to poorly integrated terminology and inefficient change management describe preventable harm that Tinkar addresses:

1. **Inability to recognize equivalence.**
    - Difficulty with determining that codes/terms using standard terminologies from disparate health IT systems represent a common clinical idea/concept (e.g., “COVID-19 positive” and "COVID-19 detected").
2. **Inability to represent a pertinent phenomenon.**
    - A new set of local terminology may be managed with value sets and concept gaps addressed in quick iterations (e.g., "COVID-19 negative test result" was needed in practical use before official SDO releases, or gaps like "mild anemia" which was proposed but not accepted by both the international and U.S. SNOMED CT® release).
3. **Flawed information.**
    - Incorrect usage or representation of clinical ideas/concepts from standard terminologies due to a lack of harmonization and multiple representations that currently exist (e.g., LOINC®, and SNOMED CT® have overlapping concepts).
4. **Inability to reliably and safely evolve over time [4].**
    - There is a lack of clear, detailed descriptions of changes to terminologies over time so that changes can be understood by implementers. Terminologies often change in ways that are convenient for the creators, but complex for the users (e.g., redundancy, major name changes, code reuse, and changed codes).

**Consider the following examples of implementations that have gone wrong:** [5][6][7][8]

- **Computer error may have led to incorrect prescribing of statins to thousands of patients.**
    - Thousands of patients in England may have been incorrectly prescribed or taken off statins because of a major IT glitch.
    - Underlying cause: (1) code mapping errors, (2) brittle means for determining equivalence.
- **Alert for monitoring thyroid function when taking Amiodarone stopped working.**
    - Amiodarone is associated with several side effects, including thyroid dysfunction, which is due to amiodarone's high iodine content and its direct toxic effect on the thyroid.
    - Underlying cause: (1) the identifier for the drug amiodarone was changed in another system, (2) uncoordinated means for determining equivalence.
- **62 percent of clinical decision support (CDS) malfunctions were attributable to changes in underlying codes or data fields.**
    - Change is a constant feature of providing healthcare.
    - Underlying cause: (1) poorly managed change.

Tinkar addresses challenges and problems from the above implementation examples:

| **Challenge** | **Tinkar Solution** |
|---|---|
| Uncoordinated, or brittle, terminology integration frequently breaks across systems | Standardize (and facilitate sharing) of terminology representation across systems |
| Management of change over time | Consistent representation and configuration management |
| SNOMED CT's® proprietary aspects prevents use as a common format for LOINC® and similar | Build on existing SNOMED CT® foundation, rather than reinvent, using an open-source initiative approved permissive licenses (i.e., Apache 2) |

## 1.3. About Tinkar

Tinkar provides the foundation of a knowledge architecture that is intended to integrate reference terminology from distributors (e.g., SNOMED CT®, LOINC®, RxNorm) with local concepts to support interoperable information semantics across the enterprise.

This specification introduces an agile approach to terminology design and formatting that promotes the use of self-describing data. It is a shift from hard-coded models that have been favored due to their prescriptive nature but have shown limited flexibility and extensibility. Like FHIR, this specification places the focus on a self-describing, extensible approach to representing terminology. Therefore, Tinkar aims to be both self-describing and completely machine processed:

1. Self-describing machine-readable representation of terminology, such that if an application can process the metadata, it should be able to import the content/concepts and make it available to enterprise applications.

2. The machine-readable terminology could generate human-readable documentation so that business analysts and developers can understand and apply it correctly.

## 1.4. Tinkar Objectives and Purpose – Manage Terminology and Change

This specification describes the requirements and characteristics of systems required to manage terminology produced by a variety of organizations across a healthcare enterprise. This foundation must allow enterprise to extend terminology standards and implement extensions in a timely fashion.

This specification is intended to support healthcare organizations' standard terminology modules, value sets, and coding systems as well as local terms and equivalence mappings.

A standard-based Terminology Knowledge Architecture (Tinkar) specification is necessary to support the operation of a variety of systems intended to deliver knowledge management for terminology to vendors, providers, and even standards-development organizations, like HL7.

## 1.5. Related Efforts

Previous efforts have attempted to create a common set of terminology capabilities and services by specifying a single pre-defined structure for managing terminology. Unfortunately, a hardwired structure that works for one standard may not work for another. The inability to integrate content across terminology standards is a barrier to implementing services and modules that can deliver integrated concepts, code lists, and value sets required by enterprise systems for treatment, research, business process automation, quality measures, and outcome analysis.

- Clinical applications require integrated terminology to create interoperable clinical statements that are organized into messages, documents, or resources.

- Data analysis and research require integrated terminology to analyze aggregated information. Interoperability, clinical decision support, or other types of automation require common semantics based on a set of integrated models across reference standards (e.g. SNOMED CT®, LOINC®, RxNorm, etc.).

The Unified Medical Language System (UMLS) Metathesaurus is a compilation of multiple sources organized into ‘concepts’ that contain terms from many sources. The terms within a concept are declared synonyms by UMLS editors. However, its use in terminology systems has limited utility for several reasons. First, UMLS concepts are created on lexically-based rules and use very little of the additional information (relationships between concepts) that may be available from the source terminology. It does not permit classification to identify cases of possible missed synonymy. Second, issues of currency occur because of dyssynchrony of release dates between source terminologies and the UMLS itself. Third, though there are editing guidelines in place, there may be substantial variation in editing practices between the individual editors. Fourth, the UMLS does not support a contribution model. That is, it is a static file that cannot be amended to support additional terms that may be required to fill gaps in existing terminologies subsumed by the Metathesaurus; it does not support extensions. Lastly, there is no efficient format for sharing integrated Metathesaurus content (though there is RRF). The UMLS is not an architecture for terminology management. It may only serve as a reference, noting the aforementioned limitations. An implementation of Tinkar may help address these limitations.

CTS2 is an architecture for terminology management that supports history retrieval, though it does not support an arbitrarily granular change set model for versioning. The Tinkar Specification, in contrast, provides that every new assertion, whether a new component or a change to an existing component, must have a precise version coordinate that govern granular change control. CTS2 asserts a specific terminology model and does not support unanticipated properties with a self-describing model.

The USCDI is an amalgamation of various encoding standards. The standards being identified for specific data elements do not themselves provide consistency for how encoding is represented, how those encoding standards change over time, and how those encoding standards are distributed. As demonstrated by COVID-19 data needs, coordinated extension of content, timely distribution of updates, and consistency of representation are required to effectively respond to needs of public health and syndromic surveillance. Tinkar could help make it easier to standardize the representation, distribution, version and configuration management, and ability to share extensions to the USCDI as well as the underlying terminology systems themselves.

## 1.6. Benefits of Self-Describing Architecture

Tinkar is self-describing and completely machine processed. Key advantages of a self-describing architecture (or metadata driven architecture) [9] include the following details:

**Changes can be reviewed immediately** – Every action or change by end users can be immediately previewed or tested, without needing any compilation or deployment process. The review can also be done before saving or publishing the changes, which makes it an interactive development environment for designers to create functionality in an iterative manner.

**Version Control with easy rollback** - Every time any changes are made to published terminology artifacts, the historic versions of the metadata files are maintained. This enables easy version control and rollback when necessary. Every time a change is made to any artifact, the prior version that existed is archived. When a developer needs to roll back to the prior version, it can be achieved easily.

**Any data source can be added** – A self-describing architecture facilitates the ability for multiple types of terminology data sources to be connected to the system.

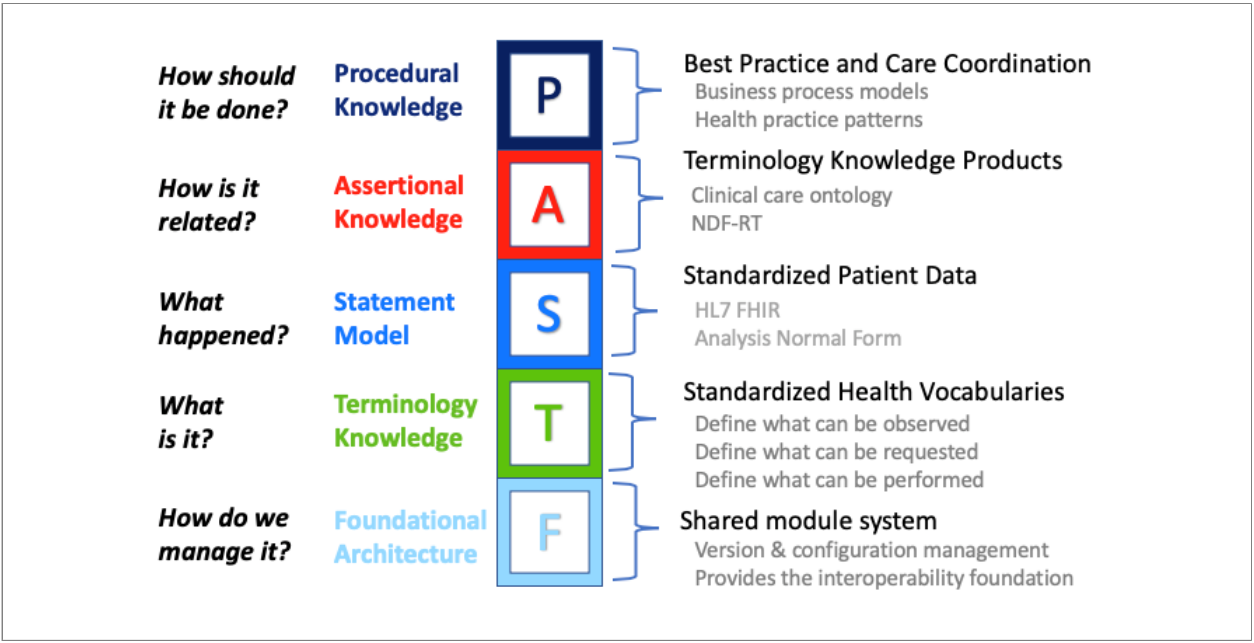
**Define granular coordinates and configuration management** – The functionality for defining granular user defined settings and controls for granular elements of clinical terminology management is supported. This includes create, read, and append settings, as well as management of individual elements, like fields or other controls.

**Faster extensions –** A benefit of a self-describing architecture is that it can abstract a lot of the deep internal complexities that makes development of standard terminologies complicated. This approach can improve processes around extensions to terminology.

## 1.7. Approach - Architectural Separation of Concerns

Increased reliance on computerized health records, including Electronic Health Records Systems, requires standardized medical terminology to encode health information consistently across systems and enterprises. Clinicians require not only objective quantitative measurements (e.g., 90 beats per minute for a patient's pulse), but also procedural context (e.g., pulse oximetry, manual) about past observations or requests for future interventions. While two quantitative measurements may be the same, the procedural information could indicate meaningful semantic differences and lead to different clinical interpretation and treatment. As information is exchanged across systems, the solution requires a common understanding of data, a method to support knowledge-representation, and clinical decision rules based on common terminology and statements. Each component must address an aspect, and together need to address the requirements of clinicians. Current HL7 standard implementations rely on profiles and templates to disambiguate statement and terminology, and provide sufficient precision for transactions, documents, and standards-based APIs. Therefore, the architectural approach described here is applicable to standards organizations developing interoperability for enterprise and project-specific implementations in equal measure.

Functional decomposition—often referred to as a *Separation of Concerns* (SoC)—across components or sections with a specific purpose is a foundational design principle for complex system architecture. SoC allows a complete system to be subdivided into distinct sections or components with well-defined functionality and dependencies. If successful, this approach allows individual sections to be able to be *reused*, as well as designed, implemented, and updated *independently* to address emerging *requirements*. This is especially useful and important in a medical context given how many different health information and clinical terminology projects are ongoing at any given time. Efforts are often uncoordinated and led by disparate and unrelated standards development organizations. In these cases, SoC allows teams to work independently, in coordination with each other, and reuse the resulting artifacts.


How should it be done?
Procedural Knowledge
P
Best Practice and Care Coordination
Business process models
Health practice patterns
How is it related?
Assertional Knowledge
A
Terminology Knowledge Products
Clinical care ontology
NDF-RT
What happened?
Statement Model
S
Standardized Patient Data
HL7 FHIR
Analysis Normal Form
What is it?
Terminology Knowledge
T
Standardized Health Vocabularies
Define what can be observed
Define what can be requested
Define what can be performed
How do we manage it?
Foundational Architecture
F
Shared module system
Version & configuration management
Provides the interoperability foundation

*Separation of concerns is an architectural design principle, whereby a system is divided into distinct sections, such that each section can address separate concerns. In this case, each architectural layer may build upon artifacts from lower layers.*

**Figure 1.1. Separation of Concerns: Knowledge Architecture**

**Foundational Architecture** – The Foundational layer of the Knowledge Architecture provides the common elements of interoperability, such as: object identity, versioning, modularity, and knowledge representation. It includes (a) the foundation and building blocks of the common model; (b) how the repeatable transformation process of disparate standards into the common model promotes interoperability with other environments; and (c) how the modules of the architecture are tightly version controlled over time. The Tinkar Reference Model belongs in this layer.

**Terminology Knowledge** - The Terminology Knowledge layer is responsible for structured sets of medical terms and codes that *define* concepts of interest, including descriptions, dialects, language, and semantic hierarchy. SNOMED CT®, LOINC®, and RxNorm are part of this layer. It defines what valid codes or expressions may be used by higher level layers.

**Statement Model** – The Statement Model layer is responsible for defining how data elements are combined to create a statement. This layer reuses the artifacts defined in the Terminology Knowledge layer. The ANF Reference Model [10] belongs in this layer.

**Assertional Knowledge** – The Assertional Knowledge layer makes use of the Terminology Knowledge layer concepts to specify *non-defining* facts that may be used by procedural knowledge algorithms. An example fact might be that "thiazide diuretics treat hypertension." Assertional Knowledge may indicate what symptoms may be associated with a disorder.

**Procedural Knowledge** – Procedural knowledge, also known as *imperative knowledge*, is the knowledge exercised in the performance of some task. An example would be determining a hypertension treatment plan by analyzing a combination of a patient's clinical statements and the available assertional knowledge. The procedural knowledge is responsible for information about standard ways to carry out specific procedures, as well as other procedural guidelines, e.g., treatment protocols for diseases and order sets focused on certain patient situations. Procedural knowledge, together with assertional knowledge, enables clinical decision support, quality measurement, and patient safety. This layer relies on the architectural foundation and terminology layers, incorporates the statement model for information retrieval, and uses the assertional knowledge. Procedural knowledge artifacts may include clinical alert rules, reminders, etc. that trigger actions or recommend interventions.

Examining a clinical procedure for controlling hypertension illustrates each of the layers of the informatics architectural separation of concerns.

- At the Terminology Knowledge layer there may be various codes and terms from disparate source terminologies to define a concept (e.g., hypertension). Ideally, these overlapping codes and terms would be oriented to the same parent concept during the transformation and integration process at the Foundational Architecture layer.
- The Statement Model layer enables representation of blood pressure measurement values (e.g., systolic BP = 140 mmHg), or the categorical data (e.g., pregnancy induced hypertension vs. renal hypertension) within a standard data structure to facilitate information exchange or retrieval, such as within a standards-based clinical statement (i.e., CIMI, CDA, FHIR, ANF, etc.).
- The Assertional Knowledge layer represents non-procedural statements, or facts, such as "Stage 2 high blood pressure is over 140 systolic or 90 diastolic," or that beta-blockers and ACE inhibitors may be used to treat hypertension, or that beta-blockers are contraindicated in patients with a diagnosis of reactive airway disease.
- Finally, the Procedural Knowledge layer provides algorithms to analyze clinical statements about a patient, in combination with the Assertional Knowledge, to recommend a treatment protocol for different kinds of hypertension, including the considerations of, e.g., patient age, co-morbidities etc., which can be generated by an electronic clinical decision support system (Statement + Assertional layers). This layer adds support for workflow and conditional logic (i.e., if-then-else).

A clear separation of concerns enables the isosemantic transformation of standards-based clinical statements to normal form in the Statement Model layer by decoupling structure from semantics and workflow.

HL7 relies on implementation guides (for V2, CDA, and FHIR) to add sufficient terminology knowledge to standards-based clinical statements. Terminology constraints documented as profiles or templates are the mechanism to create interoperable implementation guides from health IT standards. Only after the Terminology Knowledge is fully defined, can the standards-based statements be used to support business and workflow decision points consistent with the Assertional and Procedural layers described above.

## 1.8. About this Document

This document describes how encoded clinical data can be improved with a terminology management model. This terminology can be unified for HL7 and non-HL7 systems. The Terminology Knowledge Architecture, known as Tinkar, treats terminology in a common way for managing enhanced patient care and improved record keeping. The unification of models such as SNOMED CT®, LOINC®, and RxNorm will allow more robust computable medical records. The following sections contain the Tinkar Reference Model, along with illustrative examples as to the complexity and necessity of type of structure.

Tinkar will take different language sources and cohesively manage terminology data. Section 2 lays out the specific business requirements necessary for this task. The model representation is outlined in Section 3. Section 4 shows how Tinkar brings together the biomedical terminologies by a common description format. An implementation of the Tinkar specification can be used to fill the gaps between the common HL7 Terminologies and other systems like SNOMED CT®, RxNorm, UCUM, etc. The result is the distribution and sharing of cohesive data across all platforms.

# 2. Business Requirements

This section details specific business requirements for a Tinkar logical model.

## 2.1. Clinical Requirements

The ultimate goal of this effort is to support the coordination of safe, effective medicine (Requirement 1). This goal requires quality information in the patient record (Requirement 2), wherever it comes from, and the increasingly distributed nature of care that requires commonly understood data standards (Requirement 3) to ensure mutual comprehension across the care team and over time. There are four outlined clinical use cases:

- **Record Patient Data**

A care provider, already authenticated and authorized to the system and using the appropriate context to ensure the system records the data for the correct patient, adds or modifies information in the patient record. This may include signs, symptoms, impressions, diagnoses, orders, notes, or other assets.

This operation may initiate workflow processes or automated processes such as Clinical Decision Support suggestions.

For structured data using standard terminologies, the terms available are appropriate (Requirement 4) for the clinical context, for the role (i.e., terms may differ for different kinds of users), and for the data context (e.g., data entry fields may not support inactive or deprecated terms that would be allowed in search or analytical contexts).

If the available terminology does not support the provider's needs, the provider may assert a need for a new term.

- **Propose Terminology Change**

If a provider attempts to enter a term that is not supported by the enterprise terminology, the effort will be captured as a proposed term (Requirement 5).

Systems may capture this information unobtrusively as text, or prompt further information from the clinician to assist the authoring process. The system will convey at least the text and the identity of the clinician to the terminologist.

- **Review Patient Data**

A provider, already authenticated and authorized to the system and using the appropriate context to ensure the system records the data for the correct patient, finds and reviews information in the patient record.

For structured data using standard terminologies, the terms available are appropriate for the clinical context, for the role (i.e., terms may differ for different kinds of users), and for the data context (i.e., data entry fields may not support inactive or deprecated terms that would be allowed in search or analytical contexts).

Changes to the terminology that could affect record interpretation will be indicated (Requirement 6), along with a way to identify the change and its effect.

- **Review Knowledge Base changes relevant to record**

If the system identifies a relevant change, the provider may request further information.

This will include the ability to see available values and CDS results for specific dates and contexts (Requirement 7), including those under which the data was recorded or specific decisions were made.

*Clinical Use Cases*

The key capability for a clinician should be to record and review data quickly and accurately (Requirement 8), taking advantage of up-to-date classifications and decision support rules. This should be accomplished by knowing when a change in the knowledge base might affect a record. The change management capability that supports these operations should be as unobtrusive as possible to patients and care providers, and always readily available.

These operations depend on the availability not only of currently accurate terminology assets, but also assets from prior points in time (Requirement 7). These may include assets as defined or refined by different stakeholders with different sets of assumptions. For instance, whether a disorder meets a criterion defined by a standard terminology, a payor, a professional society, or a locally chartered board of specialists.

To support these needs, the Enterprise Terminology that supports the clinical systems must manage change systematically (Requirement 9), and it must do so for both internally managed and externally sourced assets.

## 2.2. Asset Curation Requirements

Curation of these assets requires detailed change data. The evolution and maturation of knowledge happens at different times and places. Keeping standards and relationships to standards current is a complex undertaking. A health system may subscribe to dozens of standard and commercial terminologies, each of which may publish scheduled updates several times a year, and any of which may push out an emergency update at any time. All these assets have different designs, so ensuring continued cohesion is expensive and time-consuming, and the necessary transformations introduce risk. Systematic management of change requires granular representation of the assets and associated asset changes.

There are best-practice capabilities in knowledge asset maintenance. The following is proposed for clinical data standards:

- Unique object identification (Requirement 10): Every object under version control must have a unique identifier, and the identifier must remain unchanged as the object is modified and different versions of it are created and saved.
- Version history retention (Requirement 11): Each version of an object must be persisted as the object changes over time, along with meta-data indicating its version identifier, time of creation, creating author, and branch of the version control system on which it was created. Further, every version of each object must remain available for retrieval and inspection.
- Version comparison (Requirement 12): It must be easy to compare two versions of the same object and identify all differences between them. Among other things, this capability is important to determine whether updates to a sub-artifact have changed its semantics in a way that may affect the behavior of one or more of its parent artifacts. Ready comparison is also important when merging two or more concurrent development efforts involving the same knowledge artifacts.
- Branching capabilities (Requirement 13): It must be possible to create a virtual copy of the entire version-control configuration, or a defined subset, in a new “path,” such that changes made to objects in this branch do not appear in the original configuration. This capability allows individual knowledge engineers to make and test changes to knowledge artifacts without affecting the work of other knowledge engineers or the integrity of knowledge artifacts currently in production. This facility is critical to the orderly and safe management of a clinical decision support system.
- Merging capabilities (Requirement 14): It must be possible to incorporate all the changes made on one branch of the version-control repository into another branch, such that any conflicts between different versions of the same objects are detected and resolved. This capability is important to enable work done by multiple knowledge engineers concurrently to be combined and incorporated into the main branch of the repository. The merging capability is also important to allow knowledge engineers to update their local branches of the repository with changes that may have been made by others to the main branch, thus ensuring that changes will remain compatible with the latest version of the system.

These core properties support authoring and maintenance operations: at a high level, this means modifying the enterprise terminologies (Requirement 15), importing standard terminologies (Requirement 16), and publishing the enterprise terminologies (Requirement 17) to the client clinical systems. The standard terminology publisher has the same needs around modification and publishing as the enterprise, and some standards import other standards as well (e.g., MedRT, which publishes relationships among other standards).

We distinguish between the Enterprise Terminologist and the Standards Development Organization (SDO) Terminologist. The Enterprise Terminologist is responsible for ensuring that the terminology resources provided to clinical systems are current and accurate. This involves managing the consumption of external terminologies as well as maintenance of assets defined within the enterprise. The SDO Terminologist is responsible for ensuring that the terminology resources provided to other terminology systems are current and accurate. This may involve managing the consumption of external terminologies as well as maintenance of assets defined within the SDO.

- **Modify Enterprise Terminology**

A user adds, modifies, or deactivates content in the terminology assets of the enterprise, including assets provided to clinical systems as well as management data used only within the knowledge base.

- **Publish Enterprise Terminology**

A user manages the publication process that supports the automated provision of terminology content to clinical systems.

- **Import Standard Terminology**

A user incorporates a new standard terminology or new version of a standard terminology into the enterprise terminology. During this process, functionality supports the assessment and management of impacts on existing enterprise assets.

- **Publish Standard Terminology**

A user manages the publication process that supports the automated provision of terminology content to client terminology servers.

- **Modify Standard Terminology**

A user adds, modifies, or deactivates content in the terminology assets of the standard, including assets provided to client terminology systems as well as management data used only within the knowledge base (e.g., changing SNOMED CT® relationships, inserting a concept in between two existing concepts). *Note that a standard can only be modified by the standard owner*. A client enterprise may add to or modify the content in an "overlay," but those changes are part of the local enterprise assets. The client enterprise cannot actually modify the standard.

*Asset Curation Use Cases*

Today, clinical systems consume terminologies, but the interfaces are point-to-point. To assert or assess new information, the tools must already understand all relevant interface models. Since an external organization may modify that model at any time, the ability to consume external assets involves ongoing manual efforts to understand or confirm the model and the design of transformations to support consumption. This is potentially expensive and risky.

We propose a “data-driven” architecture to support self-describing terminology assets. All changes can be programmatically managed with a globally consistent design. Management may involve human review, but it can leverage pattern-based recognition of specific change types for automated handling, leaving a smaller number of cases that require human judgment. This information design will support a common representation of all terminologies. There are two key requirements for this design:

1. A complete record of all changes, including relevant change context information (Requirement 11)

2. A single syntax to support the representation of all terminology assets, known and future (Requirement 18)

The context information of the first requirement includes the following:

1. The Status of the asset: whether it should be considered active or inactive in the context of these other attributes (Requirement 19). For systems that do not support status, the default will be "active."

2. The Time of the change, specified with a time zone and at an appropriate precision (Requirement 20). For systems that do not provide a time, the default will be the release time.

3. The Author of the creation or change, unambiguously identified (Requirement 21). For systems that do not provide an author, a default author will be created for the system.

4. The domain or organizational name of the larger asset within which the component is meaningful, such as code system or edition (a.k.a., Module) (Requirement 22). For systems that do not provide a module, a default module will be created for the system.

5. The production branch of that organization, e.g., for distributed development, testing, staging, or production (a.k.a., Path) (Requirement 23). For systems that do not provide a path, a default path will be created for the system.

These elements together are referred to by the acronym "**STAMP**." Every new assertion, whether a new asset or a change to an existing asset, must have a STAMP to determine when it is to be used. The STAMP properties support the ability to apply terminology assets for specific purposes. For example,

- "Path" can be used to test provisional content without physically swapping out systems.

- "Modules" are used to organize content for maintenance and publication purposes. Modules are the domain or organizational name of the larger asset within which the component is meaningful, such as code system or edition.

- "Time" supports the ability to apply CDS rules as they would have looked in the past.

A further requirement is that not only must the architecture support these properties, but that it must require the properties for all assets under curation. Without consistent application of this rule, the foundational capability of detailed version management is more difficult.

Additionally, for an asset to support a record of changes, each asset must itself be identifiable (Requirement 10).

The "single syntax" requirement is harder to satisfy. One approach would be to define a syntax that addresses the data elements of all known terminologies. This would be a heavy specification, that would be difficult to maintain, and could fail to capture new elements as terminologies are added in the future.

The other approach is to use a "self-describing" or "meta-modeling" approach, where the syntax defines not only the content but also what the content means. "Rigid" or "brittle" specifications determine in advance where information belongs: a database may use column names to suggest what belongs in a column, but there is no way to determine whether the name is a good one, or whether an instance value meets the criterion implied by the name. But flexible specifications support data definition. XML (a subset of SGML) provides a way to specify types of data and structural (not semantic) relationships. RDF goes one step further by making the relationship between an element and its containing class an explicit part of every triplet. If this relationship is specified in a controlled terminology, then assertions can be tested for validity. For example, if an RDF Schema specification (RDFS) asserts that the finding site of a lesion must be an anatomical feature, then assertions about actual lesions can be tested for valid finding sites. Furthermore, this logic specifies a "range" in the same syntactic structure as the instance assertion: changes to the knowledge base do not affect the syntactical representation of the knowledge. Systems that adopt this approach will require effort to take advantage of new features of terminologies, without having to rebuild their infrastructure when changes are made.

Having change data in discrete tagged change sets will allow the software to hide most of the complexity of version management from the human managers, allowing them to focus on significant decisions.

## 2.3. Configuration Requirements

A granular self-describing model will support any statement that can be made using concepts in a subject-predicate-object structure, and its compositional aspect permits compound predicates. It is difficult to imagine a proposition that cannot be supported. This power, however, means that there are multiple ways to support any specific kind of statement that a terminology knowledge base must support. This section addresses best practices for these cases.

### 2.3.1. Operations

Import: A user may identify content from another system and write it into the Terminology Repository. When this happens, the new content will be recorded in the common, self-describing format. When a set of content is imported, rules asserted by the source steward or the Terminology Repository steward may be used to assert structural equivalence in the repository – i.e., different source concepts may be represented as alternate representations of the same root concept. During importation of subsequent versions of a system, changes to assets on which other enterprise assets depend must be identified and managed as directed by documented policies. The import operation will usually identify sets of such changes which require prioritization to prevent redundant processing.

Search (Requirement 24): A user may use lexical or concept-based parameters to search for a set of matching assets.

View (Requirement 25): A user may view an asset, the view consisting of related information associated in visually appropriate ways. This view may omit information not appropriate to the user's context.

Compare (Requirement 12): A user may view related assets, including versions of component, in a form designed to support analytical comparison, e.g., side-by-side display.

Authoring/Maintenance: A user may modify existing content or add new content. To preserve prior states, all modifications are recorded as new versions of content: prior versions will remain unchanged. Any time a change is made, the system will identify dependent assets and rules for handling these changes.

- An addition (Requirement 26) is a new version with a new asset UUID (universally unique identifier). Patterns may assert constraints for additions, which may be specific to context (Modules, Paths, Languages, etc.).
- An inactivation (Requirement 27) is a new version of an existing asset with status set to "inactive." Patterns may assert rules for deletions, which may be specific to context.
- A change (Requirement 28) is a new version of an existing asset with the new value(s), distinguishable by STAMP value. A change may involve only a STAMP value. For example, deactivation, or import of a concept to a new module or path.

Classify (Requirement 29): A user may select a logical profile and classifier and use classification logic to test equivalence and subsumption of identified assets, or to generate a set of inferred relationships from a set of stated relationships. An inferred set may be persisted.

Publish (Requirement 30): A user may promote content into a "publication" path and produce a transmissible payload of content that can be consumed by other repositories. This promotion is a change and may require resolution of constraints on membership in that path.

### 2.3.2. Patterns for Representing Various Assets

The data architecture must support patterns for the representation of many kinds of assets. A minimal list includes the following.

1. A **term** must have:
    a. A string representation

   b. A language, possibly including refinements
   c. An indicator of case sensitivity
   d. A type
2. A **concept** must have:
   a. At least one term
   b. At least one parent, except for root concepts
3. A **logical definition** must have:
   a. A definitional status
4. **STAMP values must include**:
   a. “Active” and “inactive” status concepts
   b. At least one “default” author
   c. At least one “root” module
   d. Paths supporting “development” and “publication”
5. An **inferred classification** must indicate:
   a. The classifier used for its generation
   b. The logic profile used for its generation
   c. The stated asset(s)
6. The **module dependency** graph:
   a. Identifies the root module
   b. Lists all other modules, indicating dependency
   c. Must be acyclical

Many other patterns may be present. Implementations are expected to support:

7. **Any assembly of relationships** associating one concept with another must have:
   a. At least one default rule (constraint) for handling changes (e.g., whether assets dependent on changed assets can be automatically handled or require intervention)
8. **Any assembly of relationships** may include components that are themselves semantics
9. **Value sets** may include:
   a. Rule-based member inclusions
   b. Enumerated members
10. System-specific **import rules**:

a. System equivalences for Tinkar attribute and other infrastructure concepts

b. Specified exclusions of logical assertions to support equivalence-on-import inferences irrespective of administrative metadata

11.**Maps**:

a. Relationships for equivalence assertions

b. Relationships for subsumption assertions

c. Relationships for other functions, e.g., CDC Reportable Condition Mapping Table

12.**Constraints** on asset patterns, including:

a. Logical composition constraints on concepts, e.g., the SNOMED CT® concept model

b. Syntactic compositional constraints on strings, e.g., MIME types, ISO languages, or UCUM units

c. Pattern constraints, e.g., presence of exactly one name classified as "fully specified," or names in specified languages

d. Rules that may govern modifications to other assets, e.g., incremental addition of effort estimates based on known problematic terms.

One other feature is the set of concepts that the application will use to determine how to present the data to the user. A key dimension is the STAMP information defined above. In addition, three other "coordinates" are required for managing the presentation:

13.Language: A user may assert a required or preferred language, or a set of ranked language priorities.

14.Logic: A user may select the parameters for logical classification.

15.Navigation: A user may select the parameters for presentation of the logical classification

Like other concepts, these can be represented by the core data architecture. The application implementing the Tinkar specification must be able to identify those concepts appropriate for these uses.

## 2.3.3. Constraints

Constraints are required to:

1. Ensure that the appropriate level of detail for standard terminologies are represented within Tinkar

2. Create terminology extensions that conform to the requirements of the standard(s) the extension is based on

3. Perform general quality assurance

For example, constraints would be used to represent standard terminology artifacts, like the SNOMED CT® Machine Readable Concept Model. Additionally, constraints could be used to ensure that the terminologies represented within a Tinkar implementation are completely and consistently queried and displayed.

These same constraints can be used to create new content within a Tinkar implementation to specify the minimally viable data that would be required. For example:

4. All concepts must have at least one Fully Qualified Name within at least one Language or Dialect.

5. All concepts must have at least one Name specified as Preferred within at least one Language or Dialect.

6. All concepts must have at least one parent, unless it is a root concept.

Constraints can be applied (or not applied) based on various criteria to perform Quality Assurance on content that is represented within a Tinkar implementation. For example:

7. SNOMED CT® FullySpecifiedName hierarchy tags are applied based on where a concept exists in a hierarchy

8. Relationships between concepts have domain (based on hierarchy) and range (the hierarchy(s) of values that a relationship takes)

9. Modeling templates can be specified to ensure that new content that is created under a certain node in a hierarchy uses similar wording and relationships.

Since some Quality Assurance Constraints do not always indicate an error, an Allow List could also be represented as a Semantic to record concepts that are allowed to not conform to a constraint. Constraints would be represented using semantics as they are self describing and can support multiple different representations for constraints (SNOMED CT® Expression Constraint Language, Drools, etc.). Representing Constraints as a Semantic also ensures STAMP. STAMP is versioned over time, capturing author information and allowing for tests and progress over different modules and paths.

Implementing Constraints would depend upon how the Constraints are written and formatted. For example, Implementers could utilize a Rete algorithm through something like Drools to implement Constraints.

### 2.3.4. Minimally Required Content

A Tinkar implementation must be furnished with the following content:

1. One root concept

2. One module dependency graph

3. Infrastructure concepts a.k.a Tinkar Model Concepts to support the core patterns listed above

4. Import rules to support import of standard terminologies, including:

    a. Equivalences to support semantic integration of terminologies (e.g., that a LOINC® “system” instantiates the same relationship concept as the SNOMED CT® “inheres in” attribute)

    b. Exclusions to support removal of non-semantic properties from classification (e.g., RxNorm Translated CDs)

Definitions for Tinkar Model Concepts are located in Appendix C.

## 2.4. List of Requirements

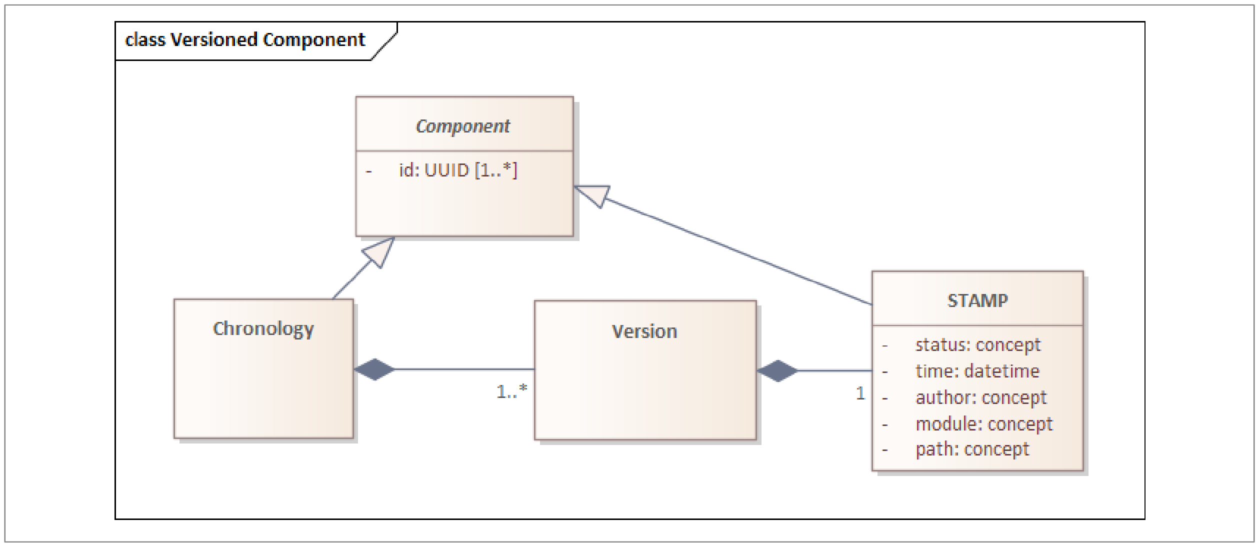
| ID | Requirement | Level | Clinical System | Terminology Management | Information Design |
|---|---|---|---|---|---|
| 1 | Support the practice and coordination of safe, effective medicine | Need | x | | |
| 2 | Provide quality information in the patient record | Need | x | | |
| 3 | Represent information in commonly understood data standards | Feature | x | x | x |
| 4 | Provide terms appropriate to the context | Feature | x | x | x |
| 5 | Capture terminology suggestions from point of care | Function | x | | |
| 6 | Indicate data for which changes to the terminology could affect record interpretation | Function | x | x | x |
| 7 | View available terms and decision support recommendations for specified dates and contexts | Function | x | x | x |
| 8 | Support rapid and accurate recording and review of record data | Need | x | | |
| 9 | Manage change systematically | Feature | | x | x |
| 10 | Identify assets uniquely | Function | | x | x |
| 11 | Retain all prior versions | Function | | x | x |
| 12 | Support comparison of versions | Function | | x | x |
| 13 | Support branching of sets of assets for independent development | Function | | x | x |
| 14 | Support controlled merging of branches by identifying and addressing conflicts with defined rules | Function | | x | x |
| 15 | Modify enterprise terminology by creating, modifying, or deactivating assets and relationships | Function | | x | x |
| 16 | Import standard terminologies, including merging capability for assets referring to prior versions of the standard | Function | | x | x |
| 17 | Publish enterprise terminologies, including application and resolution of constraints specific to the publication path | Function | | x | x |
| 18 | A self-describing method for representing terminology assets from diverse and mutable models | Feature | | | x |
| 19 | Represent the status of the asset in a context | Function | | x | x |
| 20 | Represent the time at which a change is recorded | Function | | x | x |

| ID | Requirement | Level | Clinical System | Terminology Management | Information Design |
|---|---|---|---|---|---|
| 21 | Represent the author of a change | Function | | x | x |
| 22 | Represent the system or sub-system of an asset | Function | | x | x |
| 23 | Represent the path or branch of an asset version | Function | | x | x |
| 24 | Support search using lexical or logical criteria | Function | | x | |
| 25 | Support detailed view of assets and diverse properties, filtering content not relevant to the chosen context | Function | | x | |
| 26 | Add terminology assets, including concepts, terms, relationships, definitions, value sets, maps, and others | Function | | x | x |
| 27 | Deactivate assets, preserving their original form | Function | | x | x |
| 28 | Modify assets, preserving their original form | Function | | x | x |
| 29 | Classify assets using identified tools and logical profiles in chosen contexts, with the option to persist the inferred assets | Function | | x | |
| 30 | Process a set of content for publication, including identification and resolution of unresolved constraints | Function | | x | |

# 3. Tinkar Reference Model

The Tinkar Reference Model is a logical model described herein using the *Object Management Group (OMG) Unified Modeling Language (UML) 2.0* notation to describe the structure of integrated data representation and change management for biomedical terminologies. Tinkar provides an architecture that delivers integrated terminology to the enterprise and its information systems. In doing so, it addresses the differences in management and structure across reference terminology, local concepts, and code lists/value sets. This section describes classes of objects that support a common foundational framework for terminology and knowledge base systems (e.g. SNOMED CT®, LOINC®, RxNorm, HL7, etc.). An implemenation of Tinkar can provide a single representation for all terminologies required in the U.S. and other countries, while also providing a better foundation for managing change. Tinkar could support the operation of a variety of systems intended to deliver knowledge management for terminology to vendors providers, and standards-development organizations like HL7.

## 3.1. Standard Class Model



*Versioned Component*

**Figure 3.1. Versioned Component**

The Tinkar Reference Model fulfills the requirement of capturing a complete record of all changes, including relevant context information. This is captured via the STAMP class using the following fields:

1. **Status:** A status is identified by a concept, which may be annotated with other identifying information. For example: active or inactive (Requirement 19)

2. **Time:** Timestamps must employ a common standard, which must support precision and time zone. (Requirement 20)

3. **Author:** An author is identified by a concept, which may be annotated with other identifying information as required. (Requirement 21)

4. **Module:** Assignment to the appropriate terminology (e.g., LOINC) or terminology component (e.g., SNOMED CT®, US Extension). A module is identified by a concept, which may be annotated with other identifying information. (Requirement 22)

5. **Path:** Specification of an object under version control within a terminology development lifecycle, e.g., for distributed development, testing, staging, or production. A path is a common synomyn for "branch" as used in current software version control best practices/literature. A path is identified by a concept, which may be annotated with other identifying information. A core set of paths is necessary to support publication to external organizations. (Requirement 23)

These elements together are referred to by the acronym “**STAMP**,” as described previously. Every new assertion, whether a new component or a change to an existing component, must have a STAMP to determine when it is to be used. The STAMP properties support the ability to apply terminology components for specific purposes. For example,

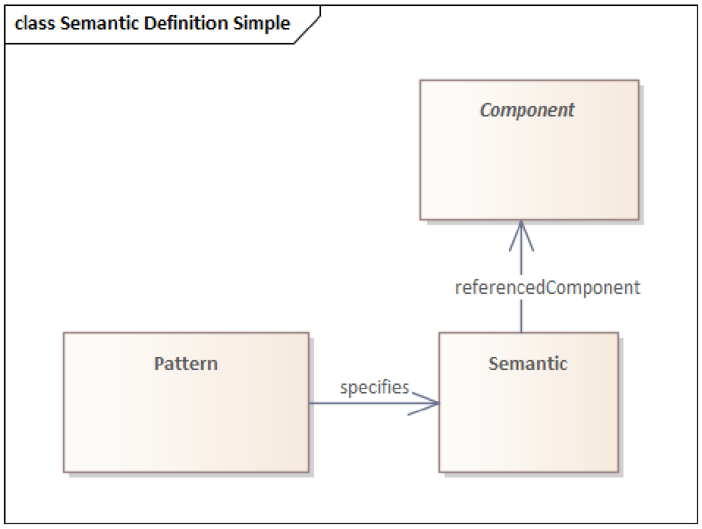
- “Path” can be used to test provisional content without physically swapping out systems.
- “Module” can be used to filter out work that has not been authorized by the enterprise.
- “Time” supports the ability to apply CDS rules as they would have looked in the past.

The Tinkar Reference Model does not merely support the ability to “STAMP” components; it asserts a requirement that all changes have a STAMP. STAMP assertions are unversioned IdentifiedComponents that are referenced by the components they scope. Since STAMP uses versioned concepts (that have a STAMP), having the STAMP as a versioned component would lead to an infinite regress.

All IdentifedComponents in the knowledge base will consist of a series of change records, called ComponentVersions, (beginning with the “Create” version), all associated to an underlying ComponentChronology.

A Components Chronology only has properties attributed to it by its versions. Looking at the IdentifiedComponent through different sets of changes (published version, geographically defined set of modules, historical timestamp) may reveal substantially different IdentifedComponents.

## 3.2. Component Types



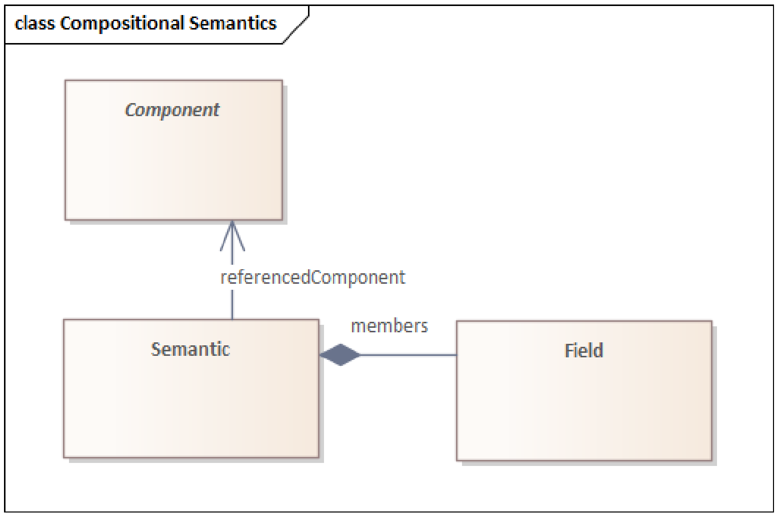
*Component Types*

**Figure 3.2. Component Types**

All Components in Tinkar are uniquely identified using UUIDs. A Component will be represented by an array of UUIDs with at least one UUID, but can be represented by more than one UUID in the case of a concept being derived from multiple sources. For example, the concept Acetaminophen (which exists in SNOMED CT®, LOINC, and RxNorm) could have a UUID from each terminology and be represented as an array of UUIDs for this single concept within a Tinkar implementation.

A Concept is identified using UUIDs and contains no information. To assemble groups of assertions and to provide information about Concepts, Tinkar uses a construct called a **Semantic**. A Semantic is a class containing a set of predicates and objects about a subject. A semantic adds meaning to the components it references, through the fields it contains. A Semantic supports the specification of value sets, compositional definitions, and other components requiring internal structure, and it specifies the nature of the compositional relationship explicitly.

The Semantic class uses a Concept to define the relationship between the value(s) and the Concept; the value itself may be either a concept or some other kind of data type, such as a string. This creates the ability to assemble assertions into more complex structures.



*Compositional Semantics*

**Figure 3.3. Compositional Semantics**

As discussed earlier, if an author makes a change to an IdentifiedComponent, the prior Version is unchanged, but a new version – with the appropriate STAMP information – is recorded. Users viewing the Concept and associated Semantics in the prior context (i.e., as of the prior time, if no other STAMP element has changed) will see the old values; users viewing the Concept and associated Semantics in the new context will see the new values.

Since it is versioned, a Semantic is manifested as a **SemanticChronology**, containing a set of **SemanticVersions**. SemanticVersion is a single instance of a Semantic with a STAMP, and a SemanticChronology is the set of versions having a STAMP for a Semantic. Concepts, too, are manifest as collections: a **ConceptChronology** consisting of a set of **ConceptVersions**. ConceptVersion is a single instance of an identifier for a concept with a STAMP and the ConceptChronology is the set of versions having a STAMP for a concept. A concept identifier specifies a ConceptChronology; specifying a ConceptVersion requires a rule or parameter for selecting among STAMP values.

If other IdentifiedComponents depend on the changed concept, these IdentifiedComponents can be identified by relationships in the Semantics. The Semantics can assert rules for how to manage these changes. A Semantic defining a value set for data entry might automatically accept any deactivations from the source system authority, while a Semantic defining a value set for research might automatically decline to adopt deactivations, or do so based on whether there are extant operational values. Escalating such decisions for human adjudication or review at multiple levels is also always an option. Systems might adopt any number of methods for dealing with identified changes: the important thing is to ensure the changes can be identified consistently.

## 3.3. Field Data Types

Tinkar supports the following field data types for use with Semantics.

1. String - a sequence of characters, either as a literal constant or as a variable. Strings could be used to represent terms from code systems or URLs, textual definitions, etc.

2. Integer - data type that represents some range of mathematical integers.

3. Float - represents values as high-precision fractional values.

4. Boolean - represents the values true and false.

5. Byte Array - an array of 8-bit signed two's complement integers.

6. Directed Graph or Digraph - a graph whose edges are ordered pairs of vertices. Each edge can be followed from one vertex to another vertex.

7. Instant - models a single instantaneous point on a timeline.

8. Planar Point - position in a two-dimensional space (a plane).

9. Spatial Point - position in a three-dimensional space.

10. Component ID List - an ordered list of Component IDs.

11. Component ID Set - an unordered list of Component IDs.

12. UUID (Universally unique identifier) - A 128-bit number used to identify information in computer systems.

13. Directed Tree or Ditree - a graph obtained from an undirected tree by replacing each undirected edge by two directed edges with opposite directions.

14. DiGraph: A graph in which a set of objects are connected where all the edges are directed from one vertex to another.

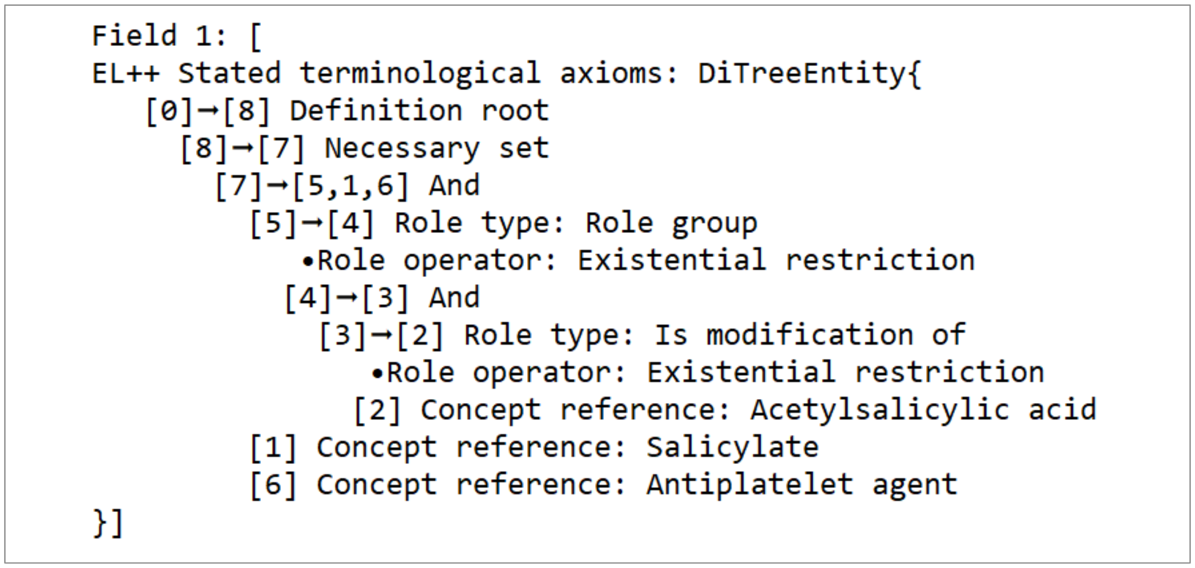
15. Vertex - The fundamental unit of data that makes up a graph or tree.

    a. In Tinkar, property graphs are used as a general-purpose data pattern to represent an abstract syntax tree (AST), such as OWL EL++. This allows for data types without requiring custom nodes.

        i. An AST may be used "during semantic analysis, where the compiler checks for correct usage of the elements of the program and the language. The compiler also generates symbol tables based on the AST during semantic analysis. A complete traversal of the tree allows verification of the correctness of the program. After verifying correctness, the AST serves as the base for code generation. The AST is often used to generate an intermediate representation, sometimes called an intermediate language, for the code generation." [11]

        ii. An AST is made up of nodes and branches. In Tinkar, every tree will always have roots, but they are specific: "An OWL EL root" vs. a "BPMN root", etc. Each node must have 0 or more children.

    b. Here is an example of Tinkar output of semantics that reference multiple concepts.



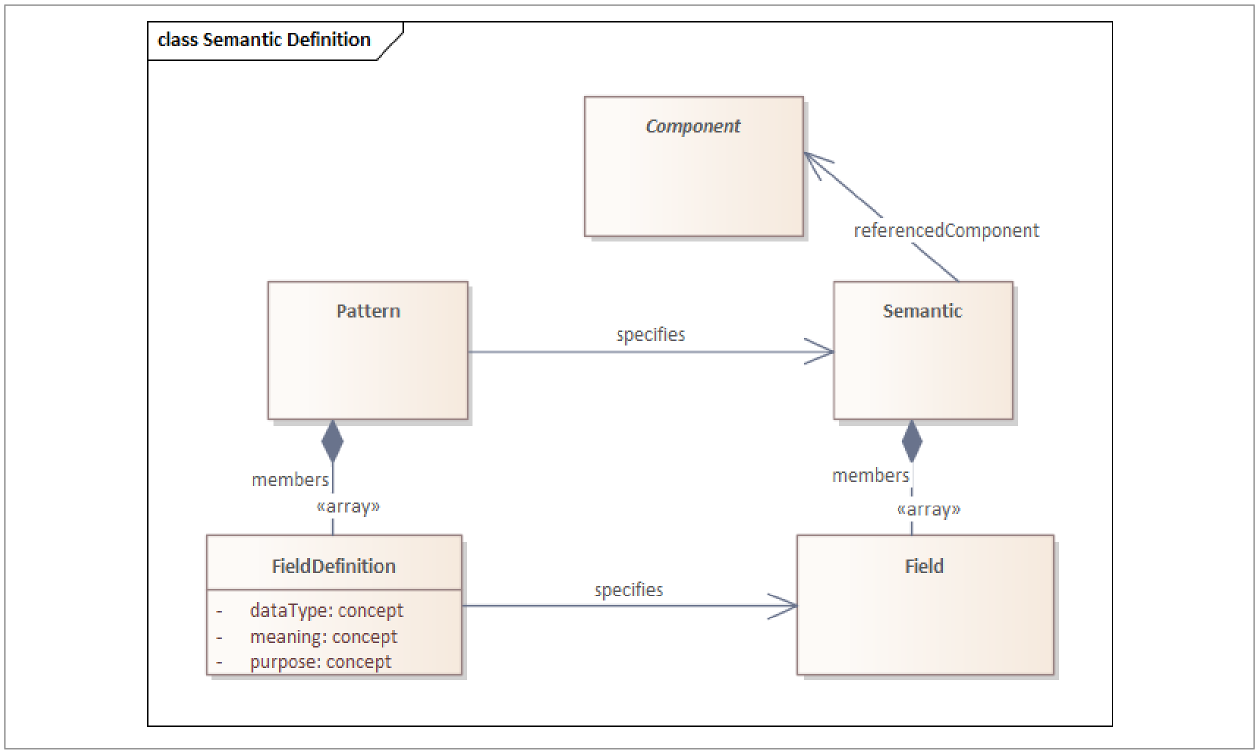
```
Field 1: [
EL++ Stated terminological axioms: DiTreeEntity{
   [0]→[8] Definition root
     [8]→[7] Necessary set
       [7]→[5,1,6] And
         [5]→[4] Role type: Role group
            •Role operator: Existential restriction
           [4]→[3] And
             [3]→[2] Role type: Is modification of
                •Role operator: Existential restriction
               [2] Concept reference: Acetylsalicylic acid
         [1] Concept reference: Salicylate
         [6] Concept reference: Antiplatelet agent
}]
```

c. In this output, one can see a sufficient set and necessary set. Bulleted items are properties in the node. The output is printed as a "depth first search." Each depth adds 3 characters of padding and shows how OWL EL ++ definitions, using only terminology and a standard property graph data structure, are represented. The 1st one is node index 0 which has a child of node index 8. Node index 0 is the OWL EL++ definition root. Node 8 points to Node 7, and the meaning of Node 8 is that it is a necessary set. Node 7 is 'And' and points to Node 5,1,6. Node 5's meaning is 'Role Type', Value is 'Role group', and its other property is 'Role Operator.' Node 5 points to Node 4. Node 4 is 'And.' Node 3 is 'Role Type.' Node 2 is Concept Reference. 7 also points to 1 and 6 (Concept References).

d. The property graph model demonstrates that each vertex has a meaning. Tinkar can use concepts to represent anything end users might need at nodes. This allows for data types without requiring custom nodes. With no changes to the underlying data structures, Tinkar can represent more than OWL EL++. With updates to terminology, Tinkar can represent any parsable standard, such as BPMN and DMN, using this property graph model and a proper set of terminology concepts and semantics represented using Tinkar.

## 3.4. Pattern (For Semantics)

The Tinkar Reference Model defines a first-class feature of the model, the **Pattern** (**PatternVersion** and **PatternChronology**). A Pattern is a class defining a set of predicates and object types that can be asserted about a class of subjects. All Semantics follow Patterns. A PatternVersion is a single instance of a pattern with a STAMP and a PatternChronology is the set of versions having a STAMP for a pattern. This feature asserts patterns that Semantic components can follow, like an XML or RDF Schema.

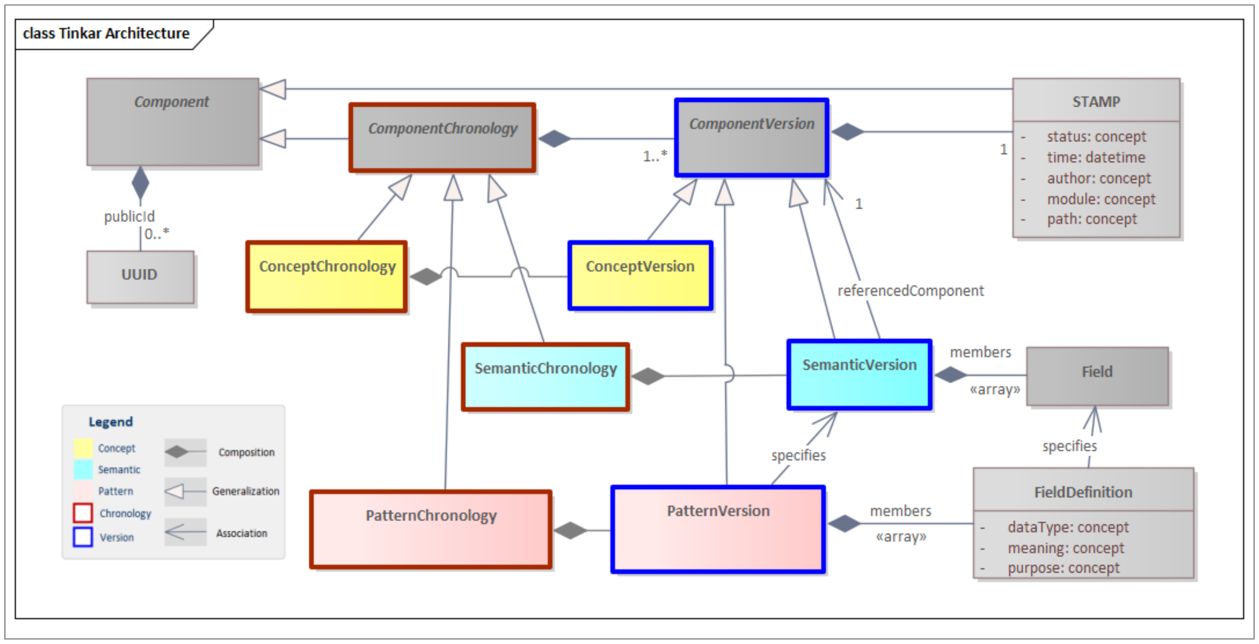


*Pattern*

**Figure 3.4. Pattern**

Using the Pattern, Semantics with varying fields and data types can be specified to represent any structure needed to provide meaning to a concept. For example, if a field within a semantic is used to describe an SDO's website, the Meaning would be "URL", DataType of "String," and Purpose of "Website." The Pattern would then contain an array of these FieldDefinitions.

## 3.5. Overall Tinkar Architecture



*Overall Tinkar Architecture*

**Figure 3.5. Overall Tinkar Architecture**

| **Class Definitions** |
|---|
| **Concept** - An identifier for a concept or instance. The identifier contains no information; all information specifying or describing the concept is asserted with Semantics.<br><br>**ConceptVersion** – A single instance of an identifier for a concept with a STAMP.<br><br>**ConceptChronology** - The set of versions having a STAMP for a concept. A concept identifier specifies a ConceptChronology; specifying a ConceptVersion requires a rule or parameter for selecting among STAMP values.<br><br>**Semantic** - A class containing a set of predicates and objects about a subject. Semantics perform the descriptive work in Tinkar.<br><br>**SemanticVersion** - A single instance of a Semantic with a STAMP.<br><br>**SemanticChronology** - The set of versions having a STAMP for a Semantic.<br><br>**Pattern** - A class defining a set of predicates and object types that can be asserted about a class of subjects. All Semantics follow Patterns.<br><br>**PatternVersion** - A single instance of a pattern with a STAMP.<br><br>**PatternChronology** - The set of versions having a STAMP for a pattern. |

## 3.6. Coordinate

The Tinkar Reference Model supports and encourages the storage of time series data that utilizes multiple coordinates, for example, STAMP, Language, Dialect, clinical domains, etc. The ability to efficiently search, display, and navigate concepts and semantics requires the ability to calculate combinations of content based on one or more of these different coordinates.

In order to facilitate the computability of various, complex coordinates, including time series data, a graph structure is commonly used in software versioned control systems. A particular type of graph structure that is commonly used is a "version graph," such as a directed acyclic graph. A version graph would enable a Tinkar implementation to recover the state of the graph at a particular point in time. Most graph databases do not support versioning as a first-class concept. It is possible, however, to create a versioning scheme inside the graph model whereby nodes and relationships are timestamped and archived whenever they are modified. The downside of such versioning schemes is that they leak into any queries written against the graph, adding a layer of complexity to even the simplest query.

Types of Coordinates:

1. STAMP coordinates are the most basic type of coordinate on which all content should be filtered. Examples of STAMP coordinates are:
    a. Most recent version
    b. Set of data from several versions
    c. All active components only
2. Language coordinates are used to control the terms that should be displayed. Examples of Language coordinates are:
    a. Displaying terms based on a language and/or dialect
    b. Prioritized list of synonyms based on a particular clinical domain
3. Logic coordinates are used to identify the various results from Description Logic Classifiers as well as the different versions of the output over time:
    a. Result from various Description Logic Classifiers
4. Navigation coordinates are used to assist in viewing and searching for a particular concept. Examples of these would be:
    a. Stated vs inferred relationships from SNOMED CT®
    b. Concepts inclusion/exclusion for a particular domain

### 3.6.1. Calculating Coordinates

The ComponentChronology contains all the versions of a component from the date it was instantiated until the most recent version. Components only get a new version whenever something about the component changes. To calculate the latest version requires the ability to find the most recent version of each component. Utilizing the STAMP Coordinates supports calculating all other coordinates:

1. Identify the Module(s) the user would like to view/search/modify.
2. Identify the Path the user would like to view/search/modify.
3. Identify the Status or Statuses the user would like to view/search/modify.
4. If relevant, identify the Author(s) the user would like to view/search/modify.
5. The last piece of the STAMP coordinate (time) is the most difficult to calculate. In most cases the user will need to find the most recent version of the component as of the current time to calculate this point of the coordinate.

However, since Tinkar supports and encourages the representation of historical, the user may need to calculate the most recent version as of a different point in time.

After the STAMP Coordinates have been calculated, additional coordinates can then be applied as well. For example, applying a language and dialect coordinate will be important not only for viewing and searching, but also to determine the appropriate preferred name for displaying a hierarchy.

### 3.6.2. Future Iterations

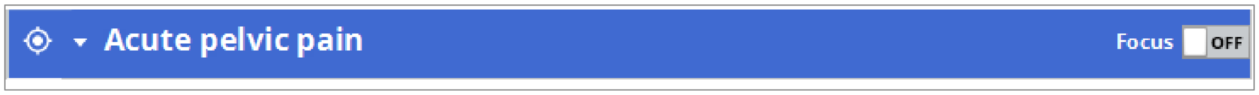
As the Tinkar specification evolves towards a DSTU and Connectathons, more coordinates and detailing will be provided.

# 4. Example Implementation of Tinkar

This section includes user documentation for an example implementation of Tinkar, referred to as Komet (https://github.com/logicahealth/komet), and how various terminologies, such as SNOMED CT®, LOINC®, RxNorm, and HL7 terminologies shall be reproducibly represented using Tinkar. The Komet browser allows users to view the search Concepts, view the Concept details (Semantics), and navigate hierarchies, as well as many other functions that are outside the scope of this ballot.

Java Reference Classes for the Tinkar model can be accessed here:(https://github.com/HL7/tinkar-java). C# Reference Classes can be accessed here:(https://github.com/HL7/tinkar-cs).

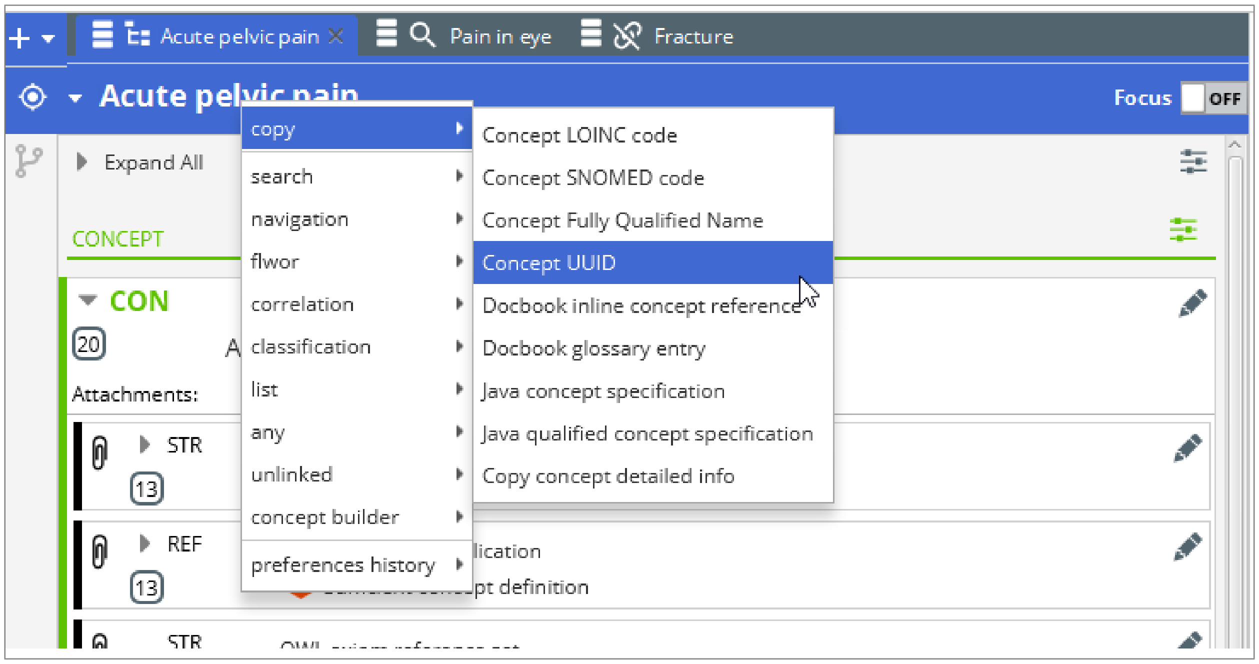
## 4.1. Concept



*Concept*

**Figure 4.1. Concept**

The image above represents the Concept "Acute pelvic pain" within the Komet Concept Detail panel. In this part of the Concept Detail Panel, no identifiers are displayed. Access to identifiers is gained by right clicking and copying them.

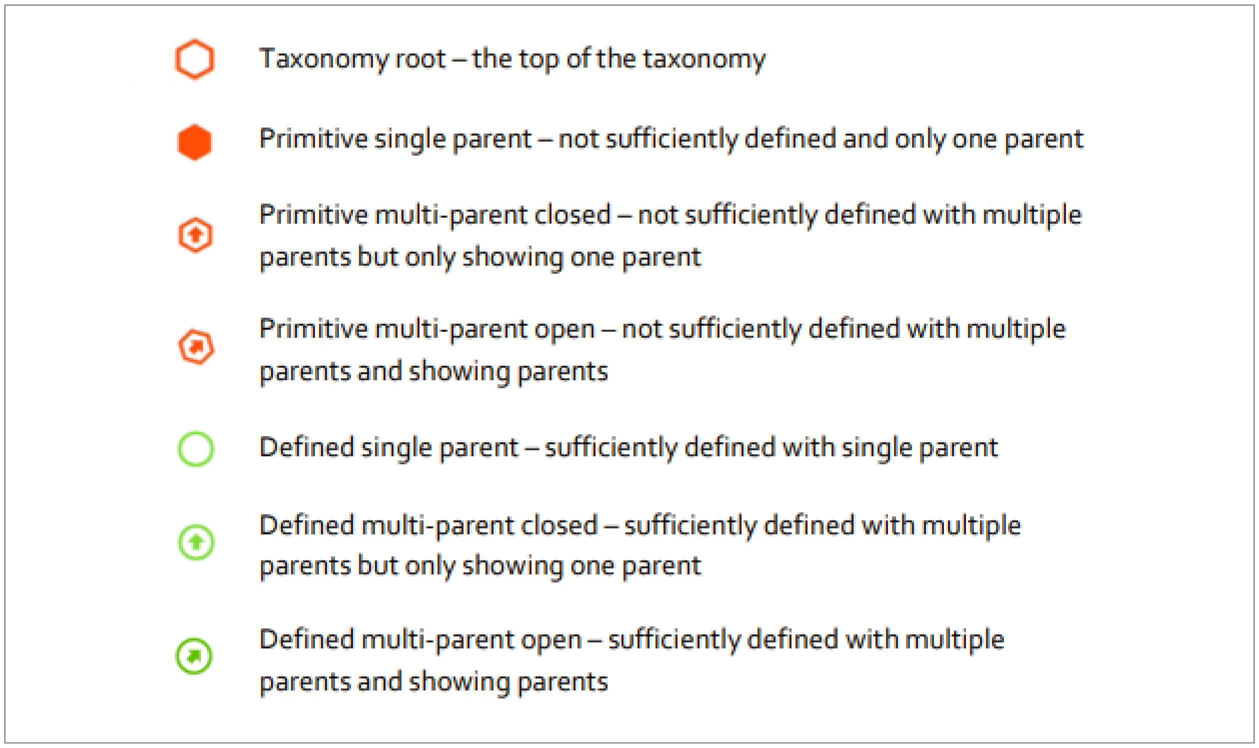


*Accessing Identifiers*

**Figure 4.2. Accessing Identifiers**

While there is a name displayed for human readable purposes, all information describing the Concept is asserted with Semantics (see below, section 4.2).

When viewing Concepts in Komet there are various icons that have different meanings based on certain Semantics that are associated with the Concept.



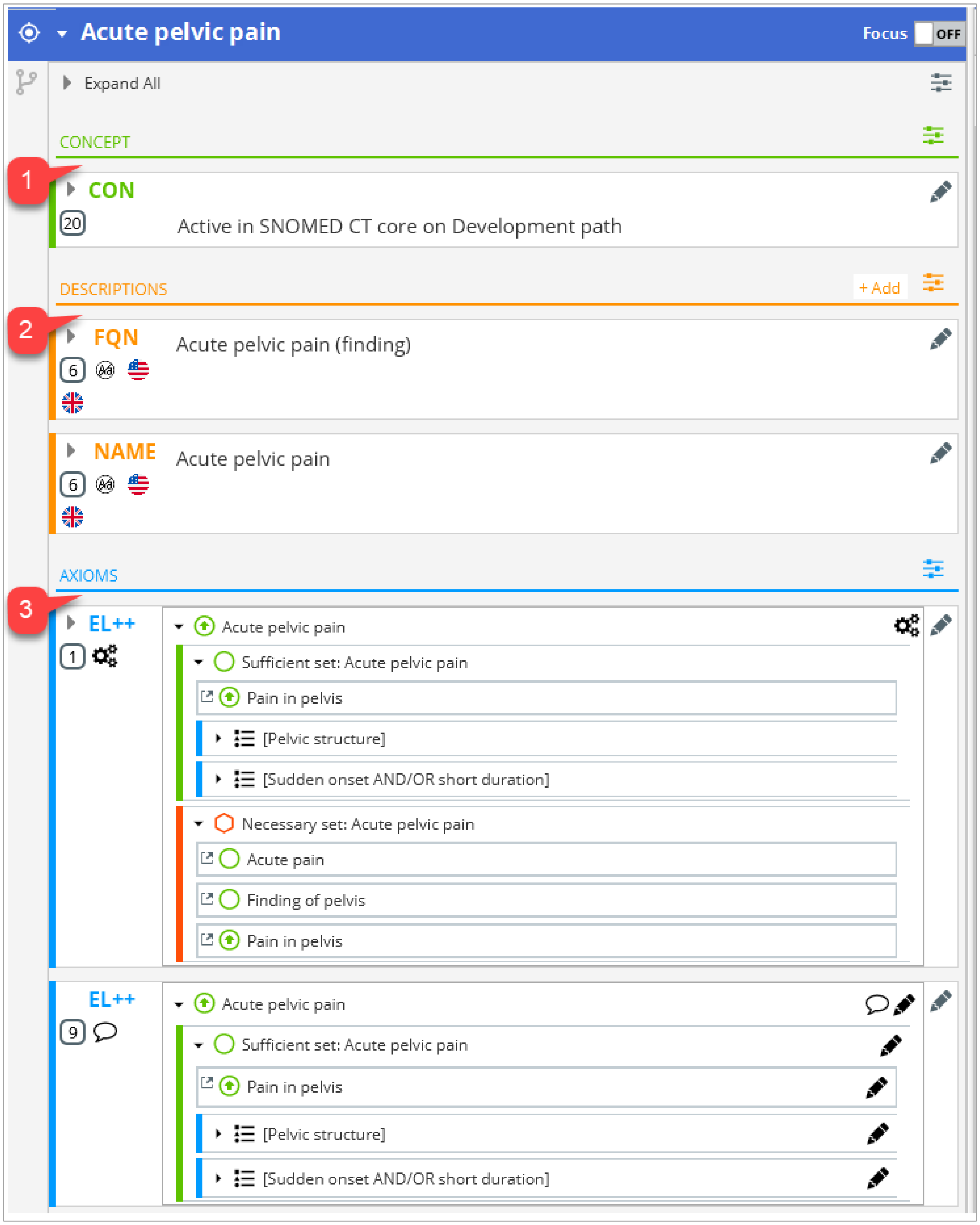
| | |
|---|---|
| ⬡ | Taxonomy root – the top of the taxonomy |
| ⬢ | Primitive single parent – not sufficiently defined and only one parent |
| ⬡ | Primitive multi-parent closed – not sufficiently defined with multiple parents but only showing one parent |
| ⬡ | Primitive multi-parent open – not sufficiently defined with multiple parents and showing parents |
| ○ | Defined single parent – sufficiently defined with single parent |
| ○ | Defined multi-parent closed – sufficiently defined with multiple parents but only showing one parent |
| ○ | Defined multi-parent open – sufficiently defined with multiple parents and showing parents |

*Concept Icons*

**Figure 4.3. Concept Icons**

## 4.2. Semantic

Semantics are the predicates and objects about a Concept. Semantics perform the descriptive work in Tinkar. In the figure below we see (1) concept semantics, (2) descriptions and (3) axioms attached to the "Acute pelvic pain" Concept which add information specifying the Concept. A single instance of one of these Semantics is a SemanticVersion (with a STAMP).


Acute pelvic pain

Focus OFF

Expand All

CONCEPT

1 CON 20 Active in SNOMED CT core on Development path

DESCRIPTIONS + Add

2 FQN 6 Acute pelvic pain (finding)

NAME 6 Acute pelvic pain

AXIOMS

3 EL++ 1
- Acute pelvic pain
  - Sufficient set: Acute pelvic pain
    - Pain in pelvis
    - [Pelvic structure]
    - [Sudden onset AND/OR short duration]
  - Necessary set: Acute pelvic pain
    - Acute pain
    - Finding of pelvis
    - Pain in pelvis

EL++ 9
- Acute pelvic pain
  - Sufficient set: Acute pelvic pain
    - Pain in pelvis
    - [Pelvic structure]
    - [Sudden onset AND/OR short duration]

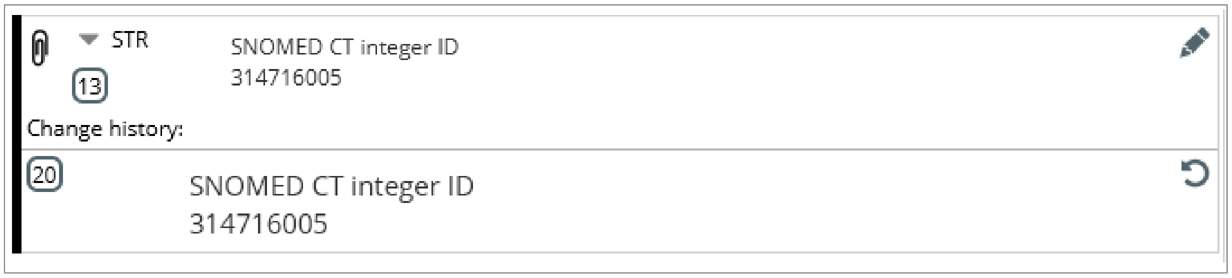
*Semantics*

**Figure 4.4. Semantics**

## 4.3. Pattern

Komet currently does not contain functionality to view and edit Patterns. Functionality is in progress that will potentially be available to demonstrate before the May 2021 HL7 meeting. Some examples of common patterns that are built into the tooling are:

- STR Pattern: This pattern is used to associate a String to a Component. The type of association to a Component is represented by a Tinkar Model Concept. In the example below, the type of String is SNOMED CT® integer ID.



STR

13

SNOMED CT integer ID
314716005

Change history:

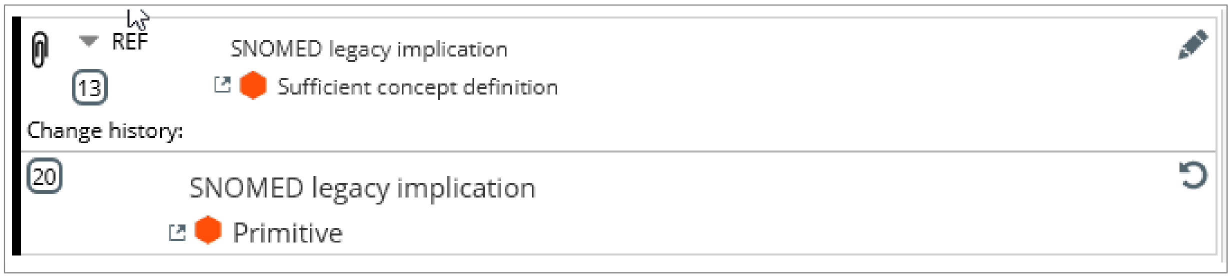
20

SNOMED CT integer ID
314716005

*STR Pattern*

**Figure 4.5. STR Pattern**

- REF Pattern: This pattern is used to associate a Concept to a Component. The type of association is also represented by a Tinkar Model Concept. In the example below, the type of association is "SNOMED legacy implication."



REF

13

SNOMED legacy implication
Sufficient concept definition

Change history:

20

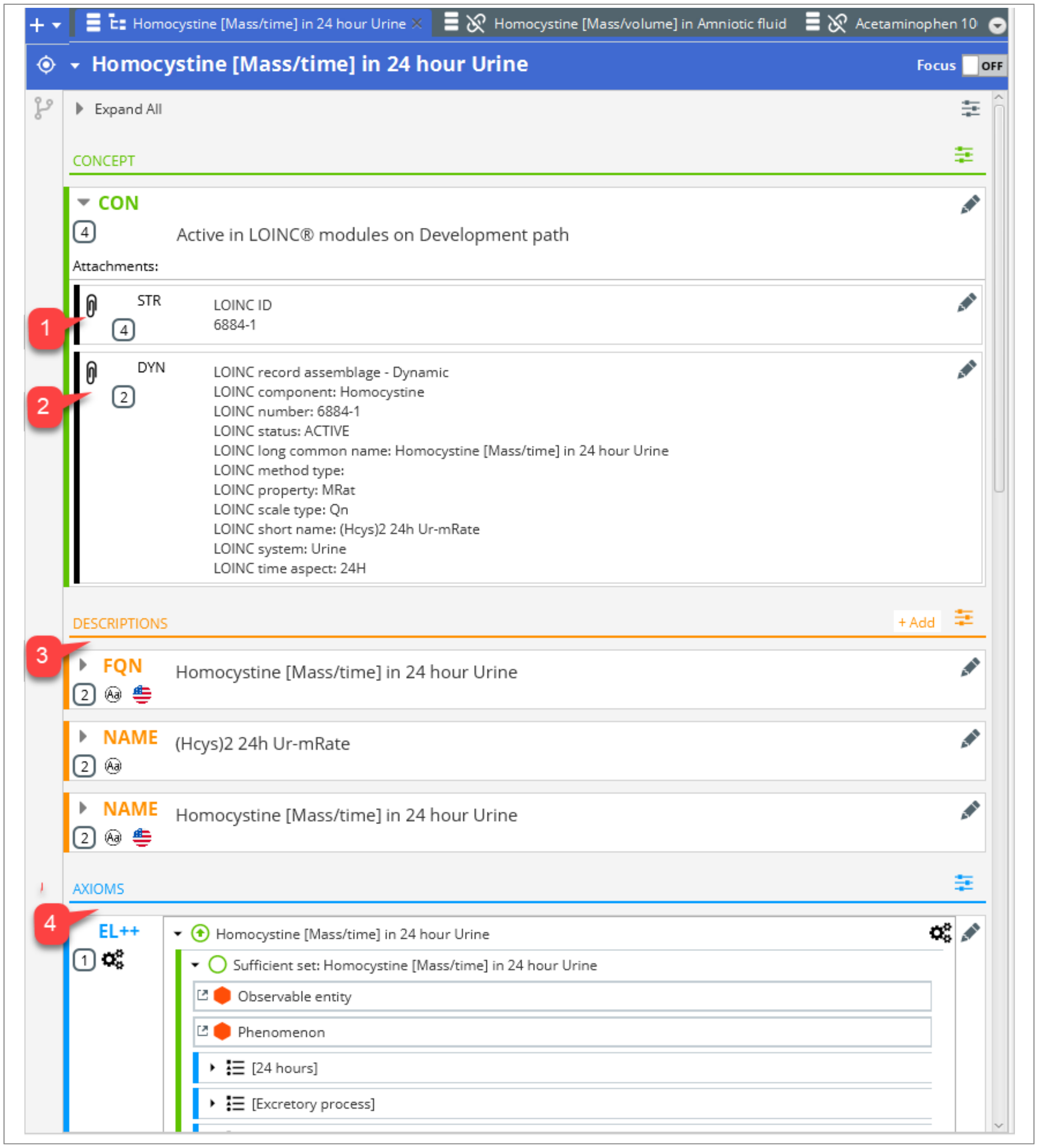
SNOMED legacy implication
Primitive

*REF Pattern*

**Figure 4.6. REF Pattern**

## 4.4. Concept Detail

Below, the details for the LOINC® Concept Homocystine [Mass/time] in 24 hour Urine is presented. Users can see the (1) LOINC® ID, (2) data from the LOINC® table, (3) Solor assigned Names, and (4) relationships from the LOINC®-SNOMED CT® Cooperation Project.


Homocystine [Mass/time] in 24 hour Urine
Homocystine [Mass/volume] in Amniotic fluid
Acetaminophen 10
Homocystine [Mass/time] in 24 hour Urine
Focus OFF
Expand All
CONCEPT
CON
4
Attachments:
1
2
3
4
DESCRIPTIONS
+ Add
AXIOMS

Active in LOINC® modules on Development path

STR — LOINC ID 6884-1

DYN — LOINC record assemblage - Dynamic
LOINC component: Homocystine
LOINC number: 6884-1
LOINC status: ACTIVE
LOINC long common name: Homocystine [Mass/time] in 24 hour Urine
LOINC method type:
LOINC property: MRat
LOINC scale type: Qn
LOINC short name: (Hcys)2 24h Ur-mRate
LOINC system: Urine
LOINC time aspect: 24H

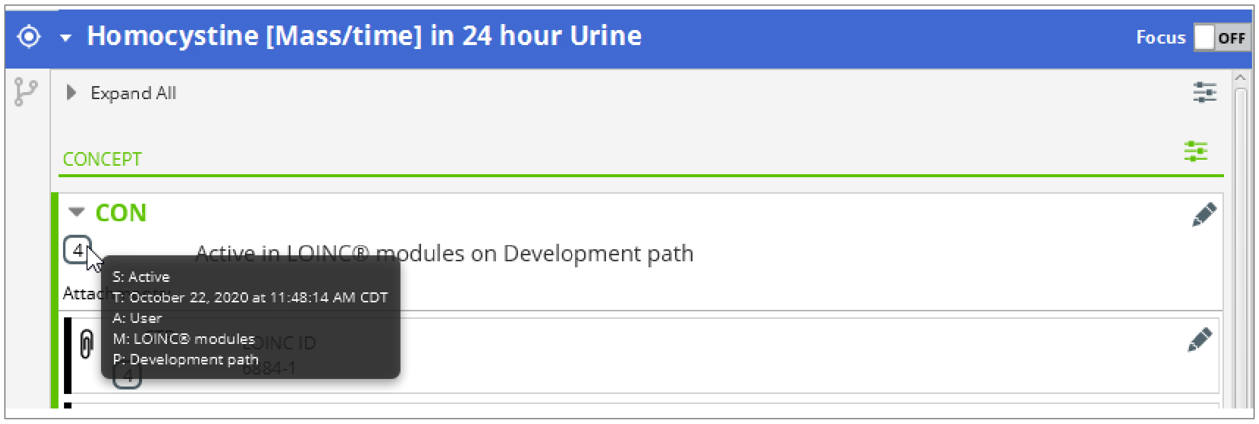
FQN — Homocystine [Mass/time] in 24 hour Urine
NAME — (Hcys)2 24h Ur-mRate
NAME — Homocystine [Mass/time] in 24 hour Urine

EL++ — Homocystine [Mass/time] in 24 hour Urine
Sufficient set: Homocystine [Mass/time] in 24 hour Urine
Observable entity
Phenomenon
[24 hours]
[Excretory process]

*Concept Details*

**Figure 4.7. Concept Details**

In the following image, the latest STAMP information for the LOINC® Concept Homocystine [Mass/time] in 24 hour Urine is presented.


Homocystine [Mass/time] in 24 hour Urine

Active in LOINC® modules on Development path

S: Active
T: October 22, 2020 at 11:48:14 AM CDT
A: User
M: LOINC® modules
P: Development path

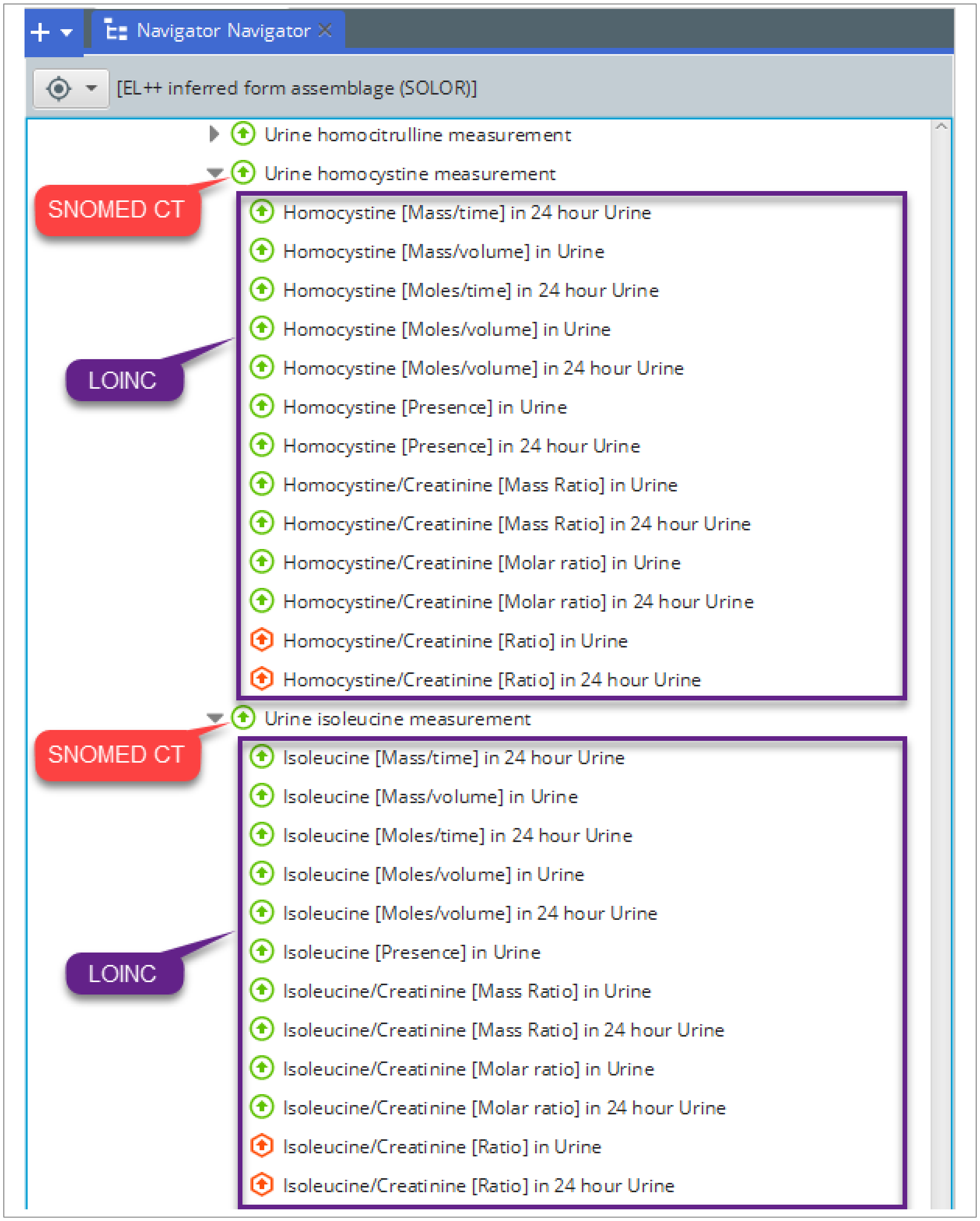
LOINC ID
6884-1

*STAMP*

Figure 4.8. STAMP

## 4.5. Navigation View

In this example, users can see the results of combining the LOINC® - SNOMED CT® Cooperation Project module with the latest SNOMED CT® and LOINC® releases. Users can then visualize and navigate the SNOMED CT® hierarchy and arrive at associated LOINC® codes that would be reached by the Urine homocystine measurement Concept.



[EL++ inferred form assemblage (SOLOR)]

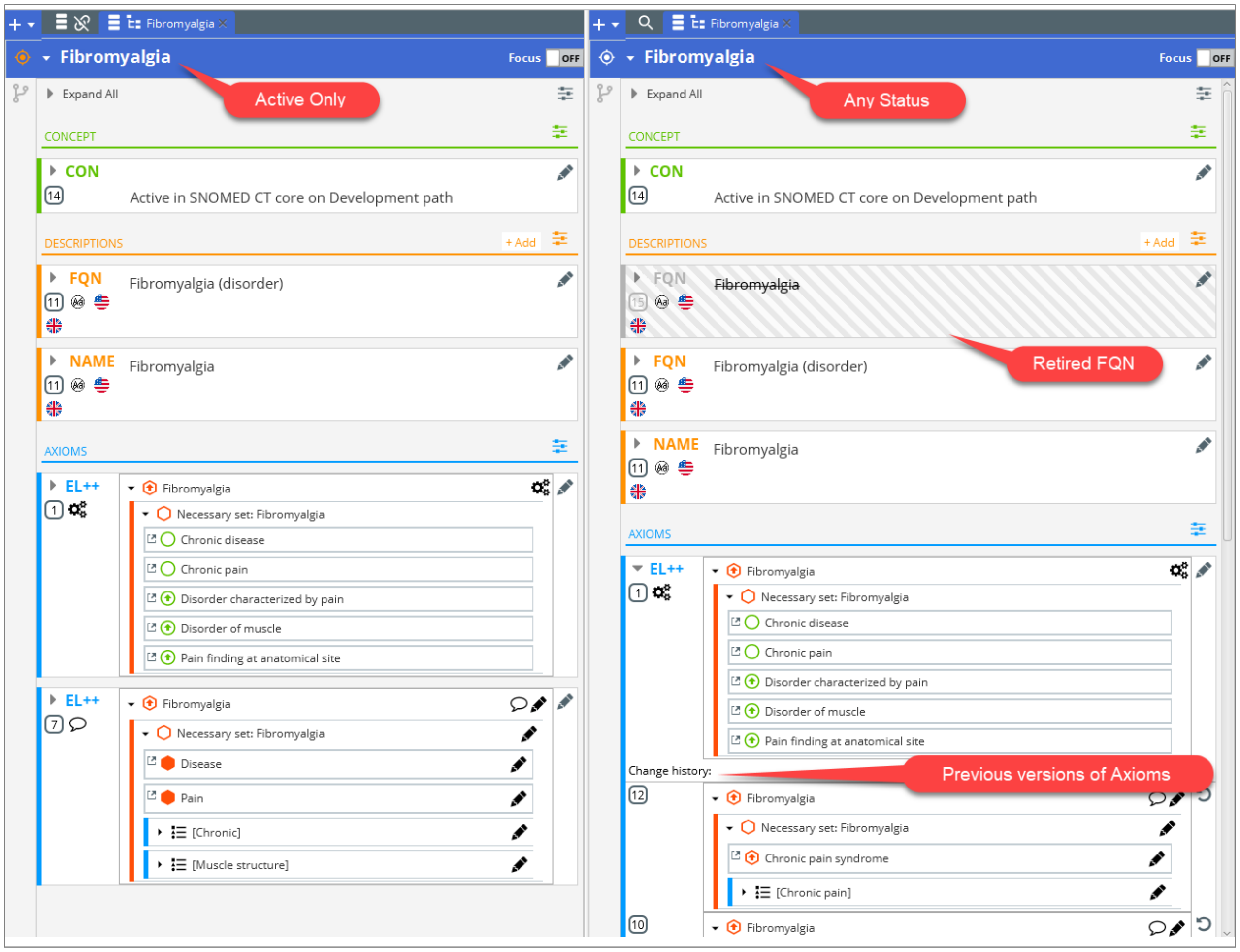
Urine homocitrulline measurement
Urine homocystine measurement
Homocystine [Mass/time] in 24 hour Urine
Homocystine [Mass/volume] in Urine
Homocystine [Moles/time] in 24 hour Urine
Homocystine [Moles/volume] in Urine
Homocystine [Moles/volume] in 24 hour Urine
Homocystine [Presence] in Urine
Homocystine [Presence] in 24 hour Urine
Homocystine/Creatinine [Mass Ratio] in Urine
Homocystine/Creatinine [Mass Ratio] in 24 hour Urine
Homocystine/Creatinine [Molar ratio] in Urine
Homocystine/Creatinine [Molar ratio] in 24 hour Urine
Homocystine/Creatinine [Ratio] in Urine
Homocystine/Creatinine [Ratio] in 24 hour Urine
Urine isoleucine measurement
Isoleucine [Mass/time] in 24 hour Urine
Isoleucine [Mass/volume] in Urine
Isoleucine [Moles/time] in 24 hour Urine
Isoleucine [Moles/volume] in Urine
Isoleucine [Moles/volume] in 24 hour Urine
Isoleucine [Presence] in Urine
Isoleucine/Creatinine [Mass Ratio] in Urine
Isoleucine/Creatinine [Mass Ratio] in 24 hour Urine
Isoleucine/Creatinine [Molar ratio] in Urine
Isoleucine/Creatinine [Molar ratio] in 24 hour Urine
Isoleucine/Creatinine [Ratio] in Urine
Isoleucine/Creatinine [Ratio] in 24 hour Urine

*Navigation View*

**Figure 4.9. Navigation View**

## 4.6. Versioning in Komet

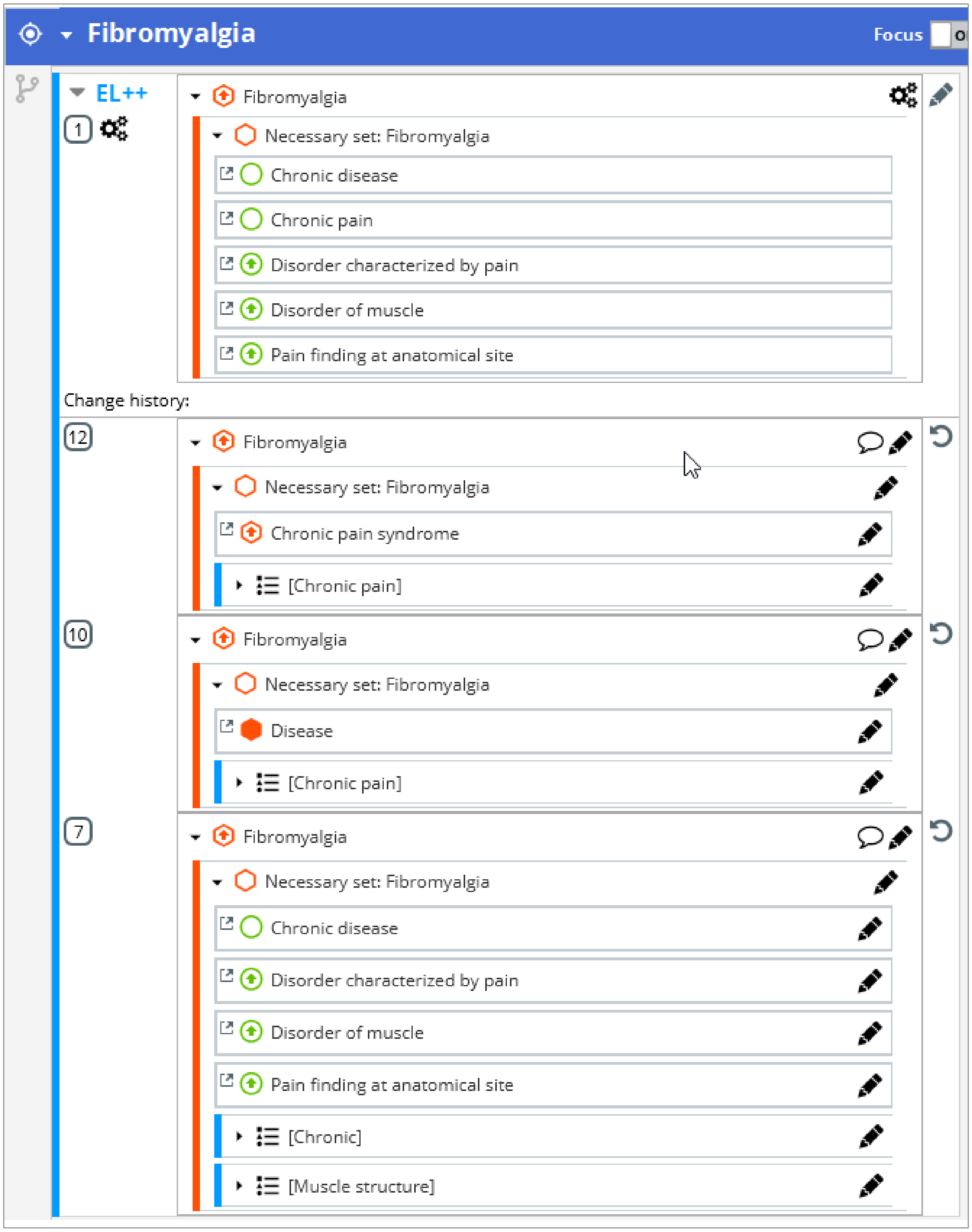
Representing all versions of components within Tinkar allows for the ability to view both Active and Inactive components and to compare the representations across time. The image below shows only the Active components for the Concept Fibromyalgia compared to all previous versions of Fibromyalgia over time.



*Latest View with Active Status versus Latest View with Any Status*

**Figure 4.10. Latest View with Active Status versus Latest View with Any Status**

The image below shows the inferred definition marked as "1" that represents the most recent and active version of the Concept Definition. Below the current definition is the change history where each of the previous versions of Concept Definitions are represented.



Fibromyalgia
Focus
EL++
1
Fibromyalgia
Necessary set: Fibromyalgia
Chronic disease
Chronic pain
Disorder characterized by pain
Disorder of muscle
Pain finding at anatomical site

Change history:

12
Fibromyalgia
Necessary set: Fibromyalgia
Chronic pain syndrome
[Chronic pain]

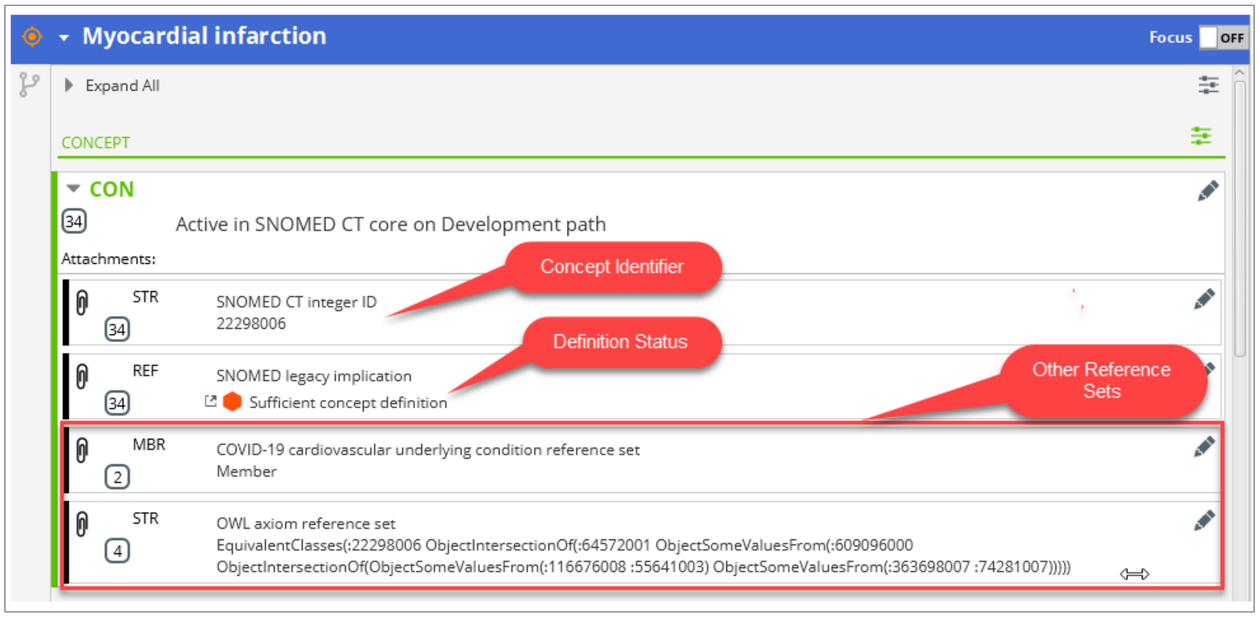
10
Fibromyalgia
Necessary set: Fibromyalgia
Disease
[Chronic pain]

7
Fibromyalgia
Necessary set: Fibromyalgia
Chronic disease
Disorder characterized by pain
Disorder of muscle
Pain finding at anatomical site
[Chronic]
[Muscle structure]

*Concept Definition History*

**Figure 4.11. Concept Definition History**

## 4.7. SNOMED CT® Example


Myocardial infarction

Focus OFF

Expand All

CONCEPT

CON

34 Active in SNOMED CT core on Development path

Attachments:

STR 34 SNOMED CT integer ID 22298006

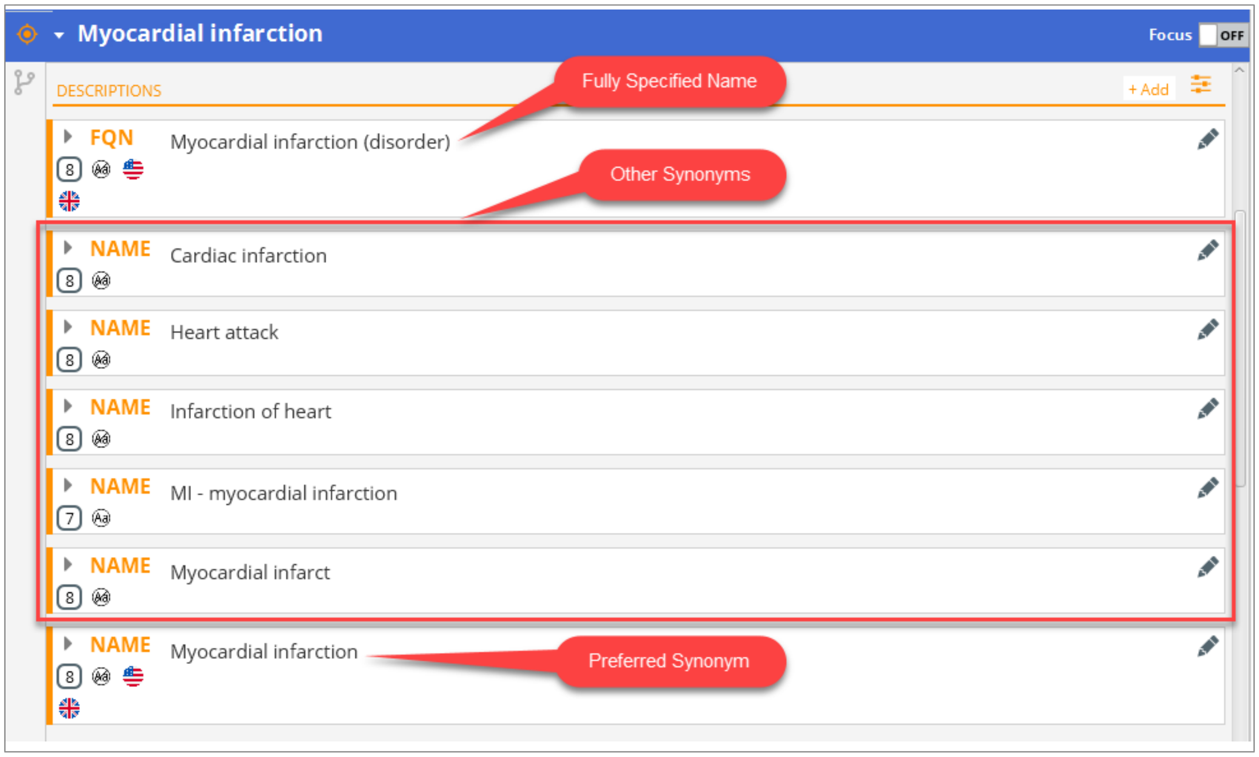
REF 34 SNOMED legacy implication Sufficient concept definition

MBR 2 COVID-19 cardiovascular underlying condition reference set Member

STR 4 OWL axiom reference set EquivalentClasses(:22298006 ObjectIntersectionOf(:64572001 ObjectSomeValuesFrom(:609096000 ObjectIntersectionOf(ObjectSomeValuesFrom(:116676008 :55641003) ObjectSomeValuesFrom(:363698007 :74281007)))))

*SNOMED CT® Concept*

**Figure 4.12. SNOMED CT® Concept**


Myocardial infarction

Focus OFF

DESCRIPTIONS + Add

FQN 8 Myocardial infarction (disorder)

NAME 8 Cardiac infarction

NAME 8 Heart attack

NAME 8 Infarction of heart

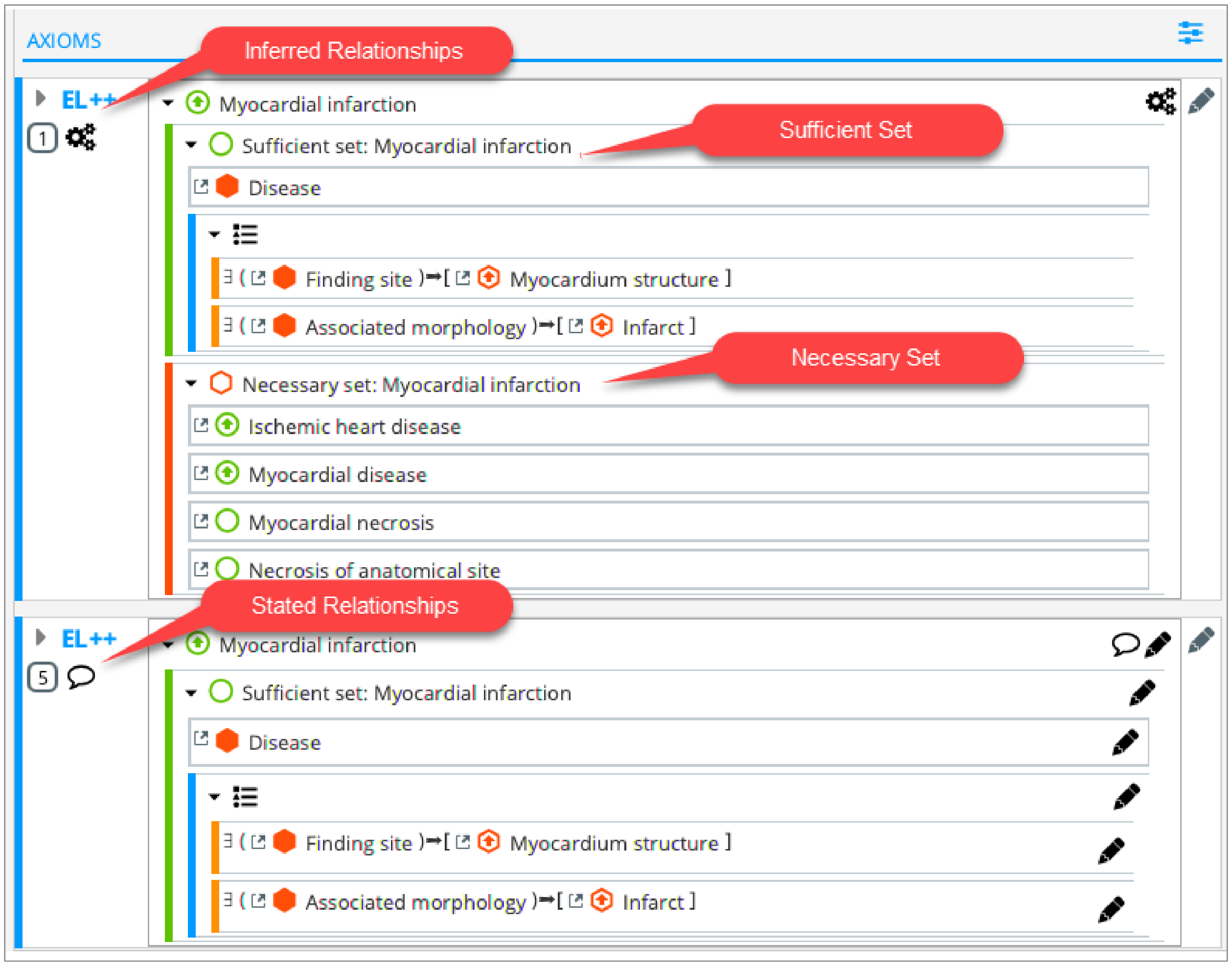
NAME 7 MI - myocardial infarction

NAME 8 Myocardial infarct

NAME 8 Myocardial infarction

*SNOMED CT® Descriptions*

**Figure 4.13. SNOMED CT® Descriptions**



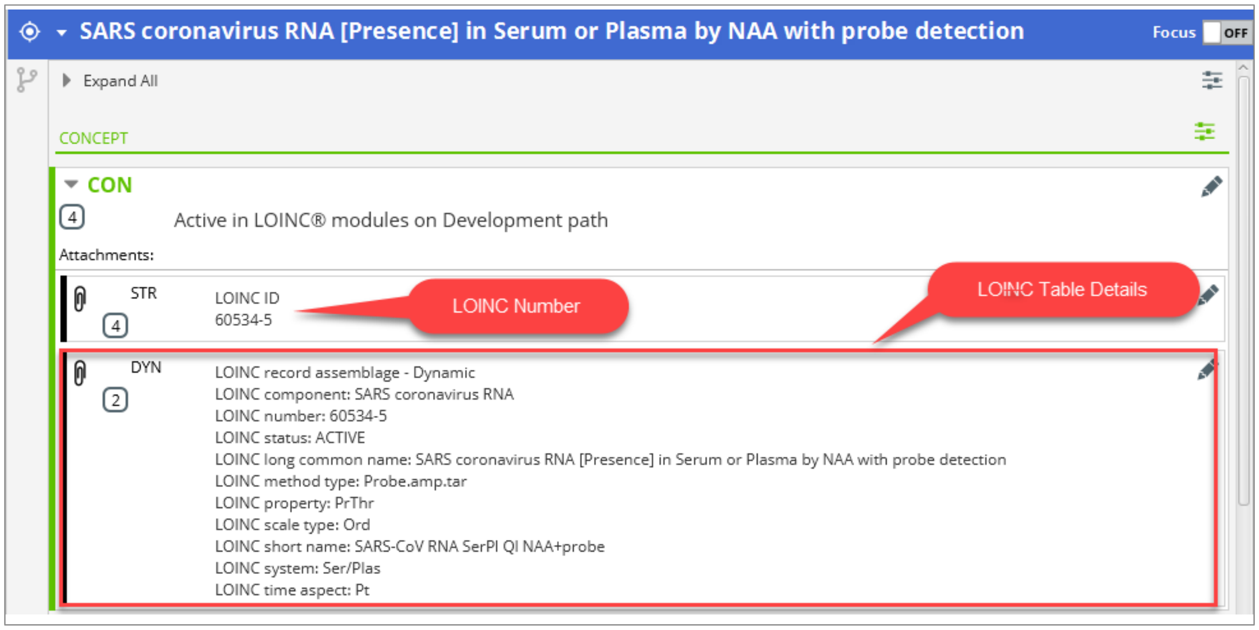
Myocardial infarction
Sufficient set: Myocardial infarction
Disease
∃ ( Finding site )➞[ Myocardium structure ]
∃ ( Associated morphology )➞[ Infarct ]
Necessary set: Myocardial infarction
Ischemic heart disease
Myocardial disease
Myocardial necrosis
Necrosis of anatomical site

Myocardial infarction
Sufficient set: Myocardial infarction
Disease
∃ ( Finding site )➞[ Myocardium structure ]
∃ ( Associated morphology )➞[ Infarct ]

*SNOMED CT® Relationships*

**Figure 4.14. SNOMED CT® Relationships**

## 4.8. LOINC® Example


SARS coronavirus RNA [Presence] in Serum or Plasma by NAA with probe detection

Focus OFF

Expand All

CONCEPT

CON

4 Active in LOINC® modules on Development path

Attachments:

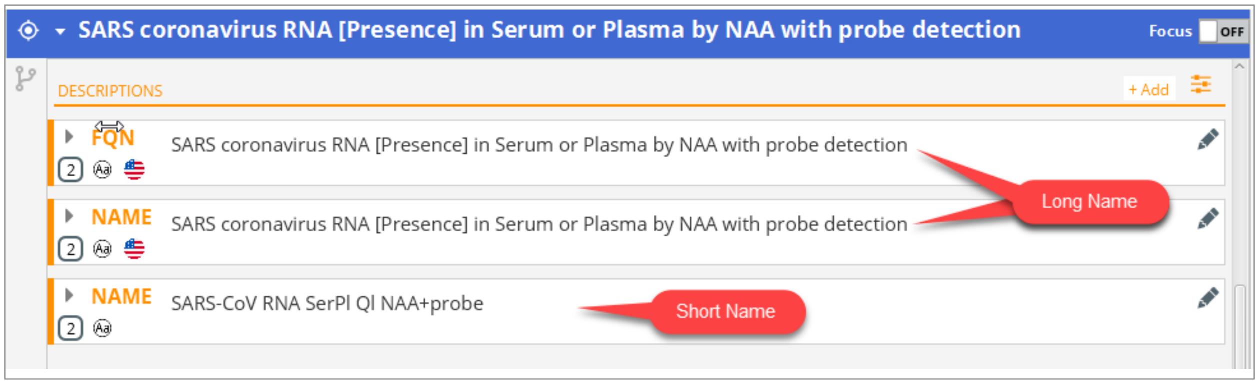
STR 4 LOINC ID 60534-5

DYN 2

LOINC record assemblage - Dynamic
LOINC component: SARS coronavirus RNA
LOINC number: 60534-5
LOINC status: ACTIVE
LOINC long common name: SARS coronavirus RNA [Presence] in Serum or Plasma by NAA with probe detection
LOINC method type: Probe.amp.tar
LOINC property: PrThr
LOINC scale type: Ord
LOINC short name: SARS-CoV RNA SerPl Ql NAA+probe
LOINC system: Ser/Plas
LOINC time aspect: Pt

*LOINC® Concept*

**Figure 4.15. LOINC® Concept**


SARS coronavirus RNA [Presence] in Serum or Plasma by NAA with probe detection

Focus OFF

DESCRIPTIONS + Add

FQN 2 SARS coronavirus RNA [Presence] in Serum or Plasma by NAA with probe detection

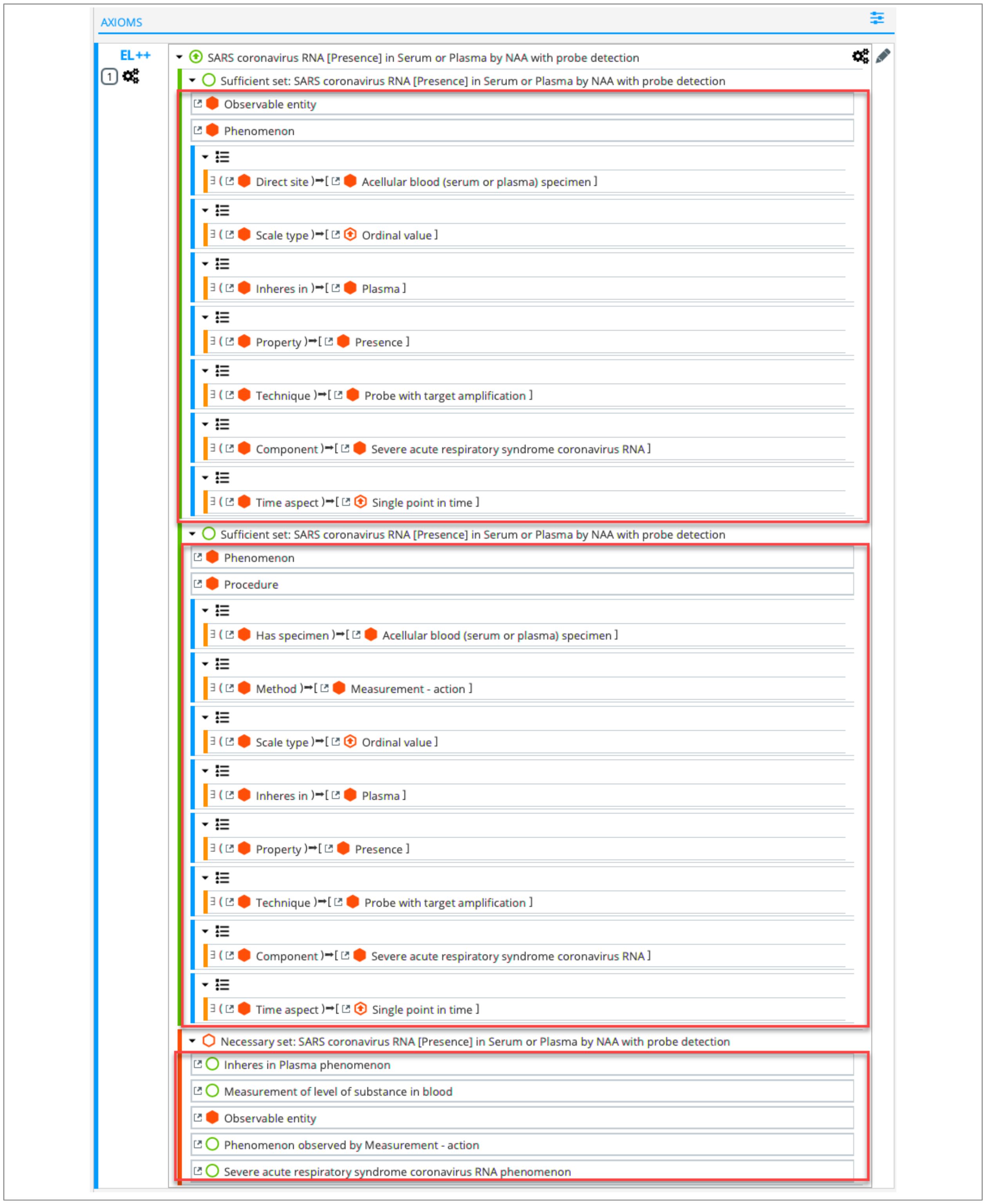
NAME 2 SARS coronavirus RNA [Presence] in Serum or Plasma by NAA with probe detection

NAME 2 SARS-CoV RNA SerPl Ql NAA+probe

*LOINC® Descriptions*

**Figure 4.16. LOINC® Descriptions**

The figure below shows the LOINC® inferred relationships that are derived from the latest version of the SNOMED CT LOINC Integration files. The parents and attributes represented in the Sufficient and Necessary sets are represented using SNOMED CT concepts based on mappings from the LOINC Part concepts to SNOMED CT concepts.



AXIOMS

EL++

SARS coronavirus RNA [Presence] in Serum or Plasma by NAA with probe detection

Sufficient set: SARS coronavirus RNA [Presence] in Serum or Plasma by NAA with probe detection

- Observable entity
- Phenomenon
- ∃ ( Direct site )➡[ Acellular blood (serum or plasma) specimen ]
- ∃ ( Scale type )➡[ Ordinal value ]
- ∃ ( Inheres in )➡[ Plasma ]
- ∃ ( Property )➡[ Presence ]
- ∃ ( Technique )➡[ Probe with target amplification ]
- ∃ ( Component )➡[ Severe acute respiratory syndrome coronavirus RNA ]
- ∃ ( Time aspect )➡[ Single point in time ]

Sufficient set: SARS coronavirus RNA [Presence] in Serum or Plasma by NAA with probe detection

- Phenomenon
- Procedure
- ∃ ( Has specimen )➡[ Acellular blood (serum or plasma) specimen ]
- ∃ ( Method )➡[ Measurement - action ]
- ∃ ( Scale type )➡[ Ordinal value ]
- ∃ ( Inheres in )➡[ Plasma ]
- ∃ ( Property )➡[ Presence ]
- ∃ ( Technique )➡[ Probe with target amplification ]
- ∃ ( Component )➡[ Severe acute respiratory syndrome coronavirus RNA ]
- ∃ ( Time aspect )➡[ Single point in time ]

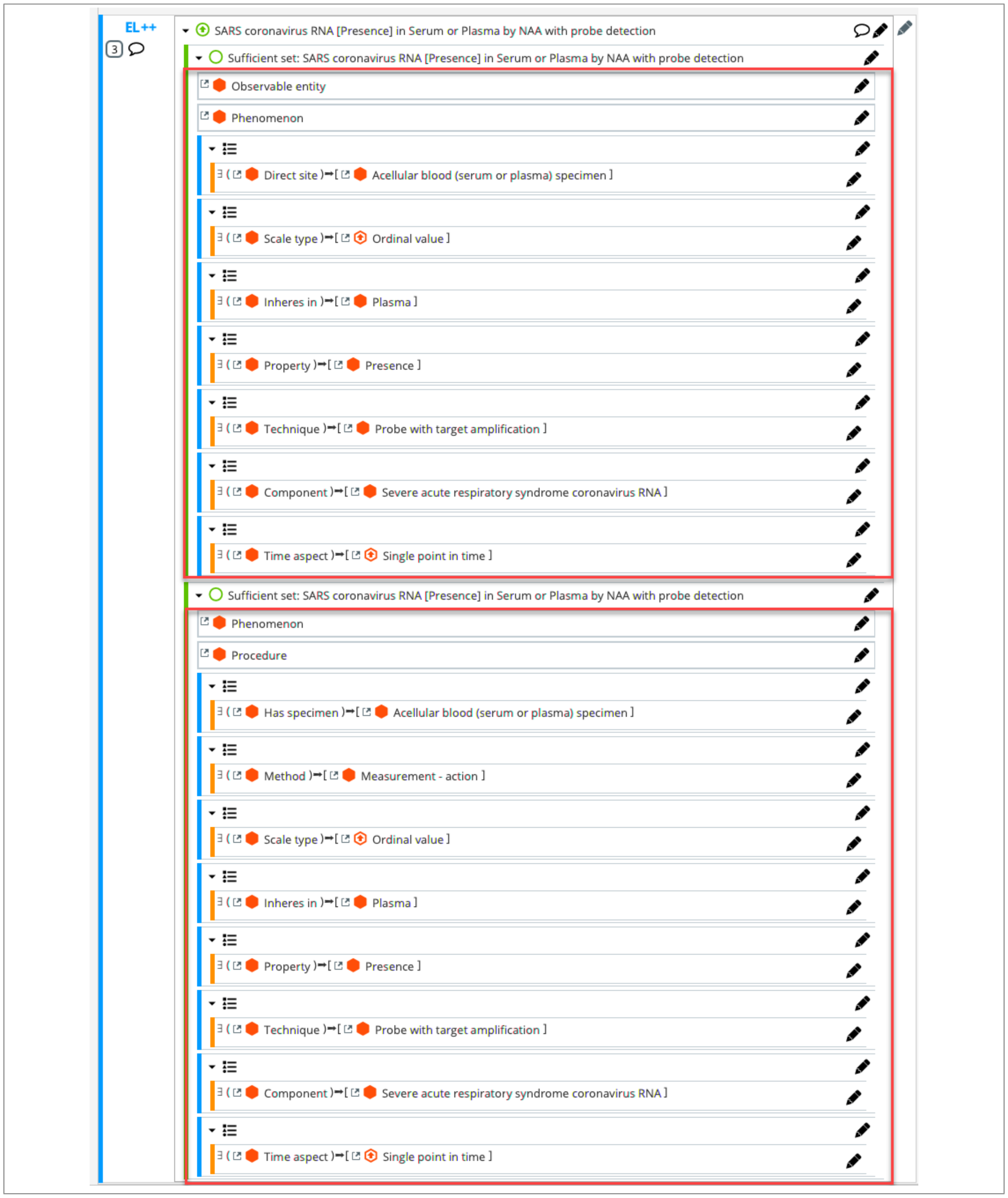
Necessary set: SARS coronavirus RNA [Presence] in Serum or Plasma by NAA with probe detection

- Inheres in Plasma phenomenon
- Measurement of level of substance in blood
- Observable entity
- Phenomenon observed by Measurement - action
- Severe acute respiratory syndrome coronavirus RNA phenomenon

*LOINC® Inferred Relationships*

**Figure 4.17. LOINC® Inferred Relationships**

The next figure shows the LOINC® stated relationships that are contained within the latest version of the SNOMED CT LOINC Integration files. The parents and attributes represented in the Sufficient sets are represented using SNOMED CT concepts based on mappings from the LOINC Part concepts to SNOMED CT concepts.



EL++

3

SARS coronavirus RNA [Presence] in Serum or Plasma by NAA with probe detection

Sufficient set: SARS coronavirus RNA [Presence] in Serum or Plasma by NAA with probe detection

Observable entity

Phenomenon

∃ ( Direct site )➡[ Acellular blood (serum or plasma) specimen ]

∃ ( Scale type )➡[ Ordinal value ]

∃ ( Inheres in )➡[ Plasma ]

∃ ( Property )➡[ Presence ]

∃ ( Technique )➡[ Probe with target amplification ]

∃ ( Component )➡[ Severe acute respiratory syndrome coronavirus RNA ]

∃ ( Time aspect )➡[ Single point in time ]

Sufficient set: SARS coronavirus RNA [Presence] in Serum or Plasma by NAA with probe detection

Phenomenon

Procedure

∃ ( Has specimen )➡[ Acellular blood (serum or plasma) specimen ]

∃ ( Method )➡[ Measurement - action ]

∃ ( Scale type )➡[ Ordinal value ]

∃ ( Inheres in )➡[ Plasma ]

∃ ( Property )➡[ Presence ]

∃ ( Technique )➡[ Probe with target amplification ]

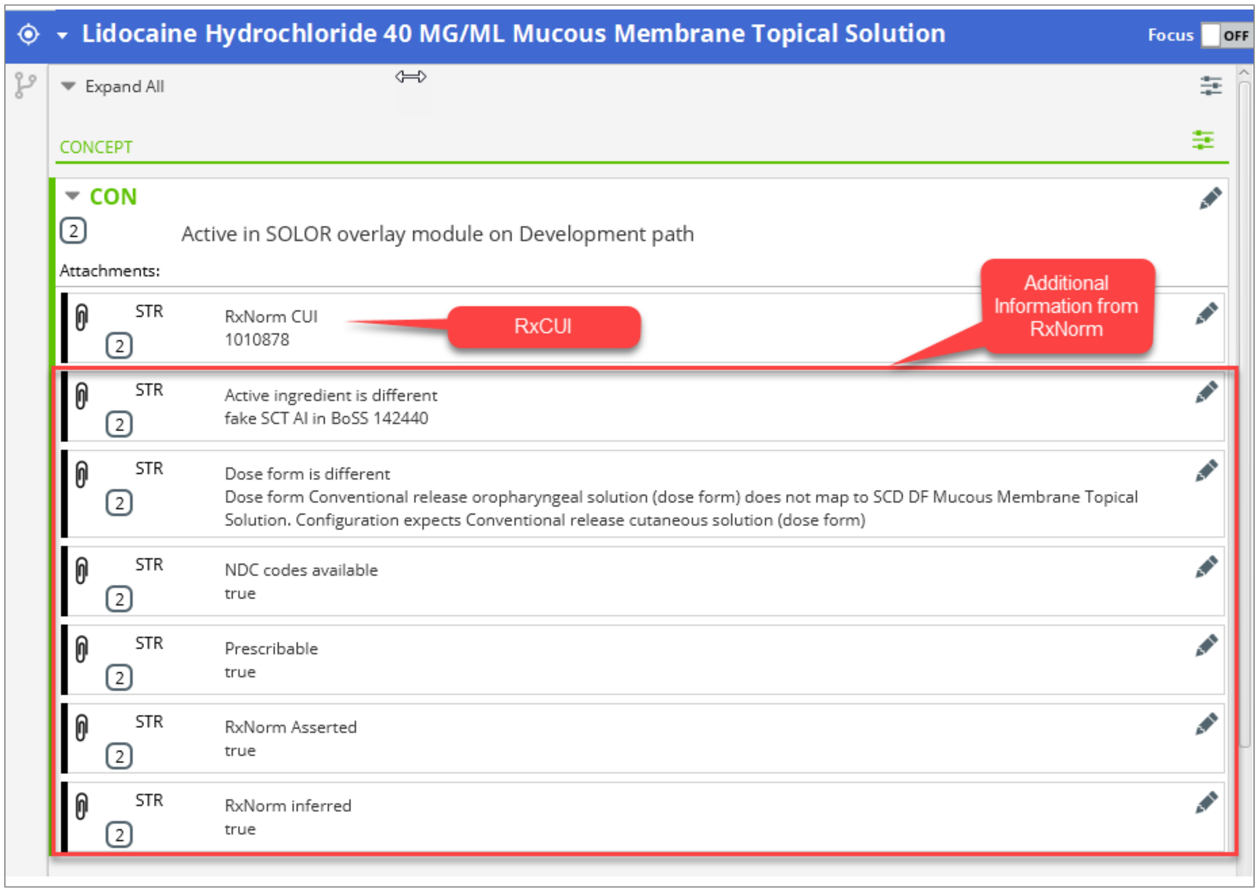
∃ ( Component )➡[ Severe acute respiratory syndrome coronavirus RNA ]

∃ ( Time aspect )➡[ Single point in time ]

*LOINC® Stated Relationships*

**Figure 4.18. LOINC® Stated Relationships**

## 4.9. RxNorm Example



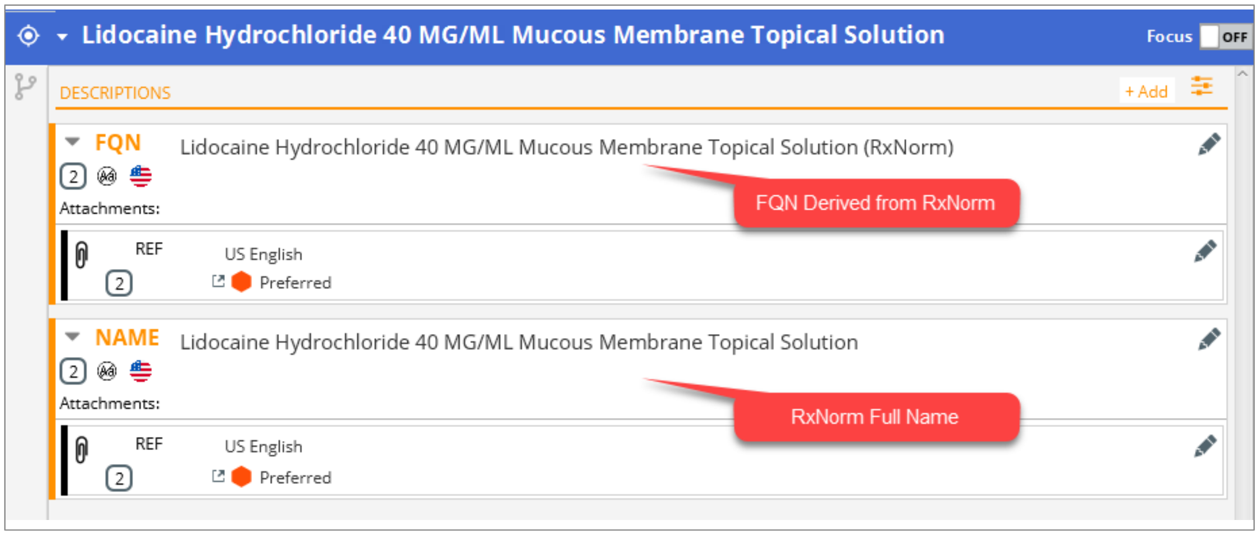
Lidocaine Hydrochloride 40 MG/ML Mucous Membrane Topical Solution

Active in SOLOR overlay module on Development path

| | | |
|---|---|---|
| STR | 2 | RxNorm CUI<br>1010878 |
| STR | 2 | Active ingredient is different<br>fake SCT AI in BoSS 142440 |
| STR | 2 | Dose form is different<br>Dose form Conventional release oropharyngeal solution (dose form) does not map to SCD DF Mucous Membrane Topical Solution. Configuration expects Conventional release cutaneous solution (dose form) |
| STR | 2 | NDC codes available<br>true |
| STR | 2 | Prescribable<br>true |
| STR | 2 | RxNorm Asserted<br>true |
| STR | 2 | RxNorm inferred<br>true |

*RxNorm Concept*

**Figure 4.19. RxNorm Concept**


Lidocaine Hydrochloride 40 MG/ML Mucous Membrane Topical Solution

Focus OFF

DESCRIPTIONS + Add

FQN Lidocaine Hydrochloride 40 MG/ML Mucous Membrane Topical Solution (RxNorm)

2

Attachments:

REF US English

2 Preferred

NAME Lidocaine Hydrochloride 40 MG/ML Mucous Membrane Topical Solution

2

Attachments:

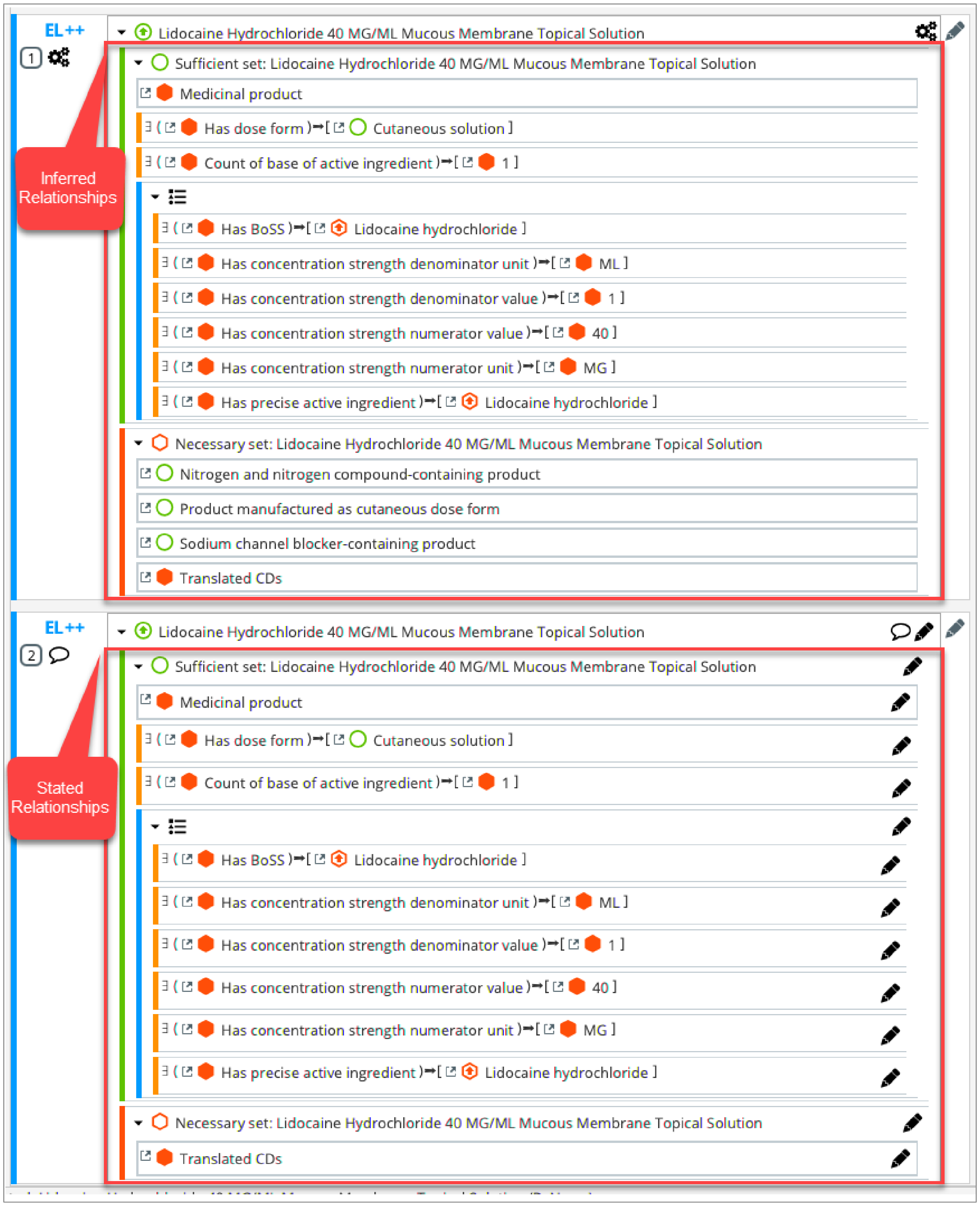
REF US English

2 Preferred

*RxNorm Descriptions*

**Figure 4.20. RxNorm Descriptions**

The figure below shows the Inferred and Stated Relationships based on a test version of an OWL file of RxNorm Stated Relationships that are based on the new SNOMED CT drug model. The relationships are represented using SNOMED CT concepts.



EL++
1
Lidocaine Hydrochloride 40 MG/ML Mucous Membrane Topical Solution
Sufficient set: Lidocaine Hydrochloride 40 MG/ML Mucous Membrane Topical Solution
Medicinal product
∃ ( Has dose form )→[ Cutaneous solution ]
∃ ( Count of base of active ingredient )→[ 1 ]
∃ ( Has BoSS )→[ Lidocaine hydrochloride ]
∃ ( Has concentration strength denominator unit )→[ ML ]
∃ ( Has concentration strength denominator value )→[ 1 ]
∃ ( Has concentration strength numerator value )→[ 40 ]
∃ ( Has concentration strength numerator unit )→[ MG ]
∃ ( Has precise active ingredient )→[ Lidocaine hydrochloride ]
Necessary set: Lidocaine Hydrochloride 40 MG/ML Mucous Membrane Topical Solution
Nitrogen and nitrogen compound-containing product
Product manufactured as cutaneous dose form
Sodium channel blocker-containing product
Translated CDs

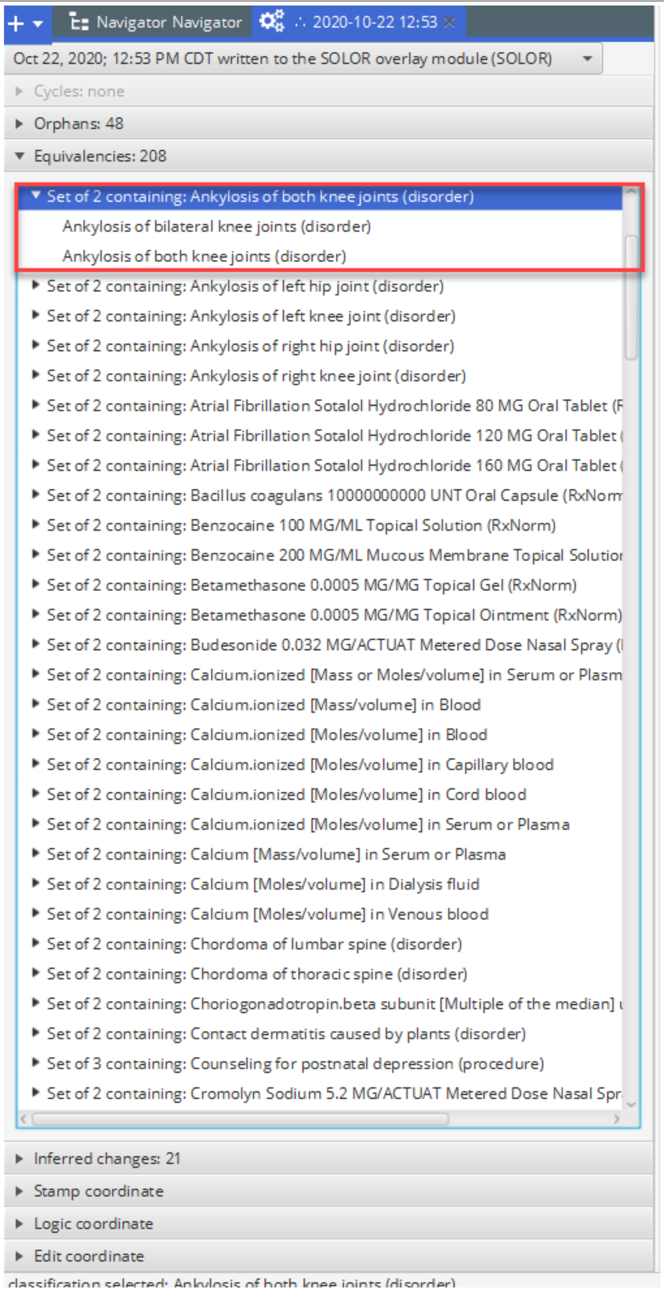
EL++
2
Lidocaine Hydrochloride 40 MG/ML Mucous Membrane Topical Solution
Sufficient set: Lidocaine Hydrochloride 40 MG/ML Mucous Membrane Topical Solution
Medicinal product
∃ ( Has dose form )→[ Cutaneous solution ]
∃ ( Count of base of active ingredient )→[ 1 ]
∃ ( Has BoSS )→[ Lidocaine hydrochloride ]
∃ ( Has concentration strength denominator unit )→[ ML ]
∃ ( Has concentration strength denominator value )→[ 1 ]
∃ ( Has concentration strength numerator value )→[ 40 ]
∃ ( Has concentration strength numerator unit )→[ MG ]
∃ ( Has precise active ingredient )→[ Lidocaine hydrochloride ]
Necessary set: Lidocaine Hydrochloride 40 MG/ML Mucous Membrane Topical Solution
Translated CDs

*RxNorm Relationships*

**Figure 4.21. RxNorm Relationships**

## 4.10. Equivalences between Modules

Komet includes the capability to use a description logic classifier to create inferred views and detect equivalences and other errors. The following image shows the results of the classifier after merging the January 2021 version of the SNOMED CT® International release with the September 2020 version of the SNOMED CT® U.S. Edition and a Solor Extension to SNOMED CT. The classifier has detected that there is an equivalence between the Concepts: "Ankylosis of bilateral knee joints (disorder)" and "Ankylosis of both knee joints (disorder)."



Oct 22, 2020; 12:53 PM CDT written to the SOLOR overlay module (SOLOR)

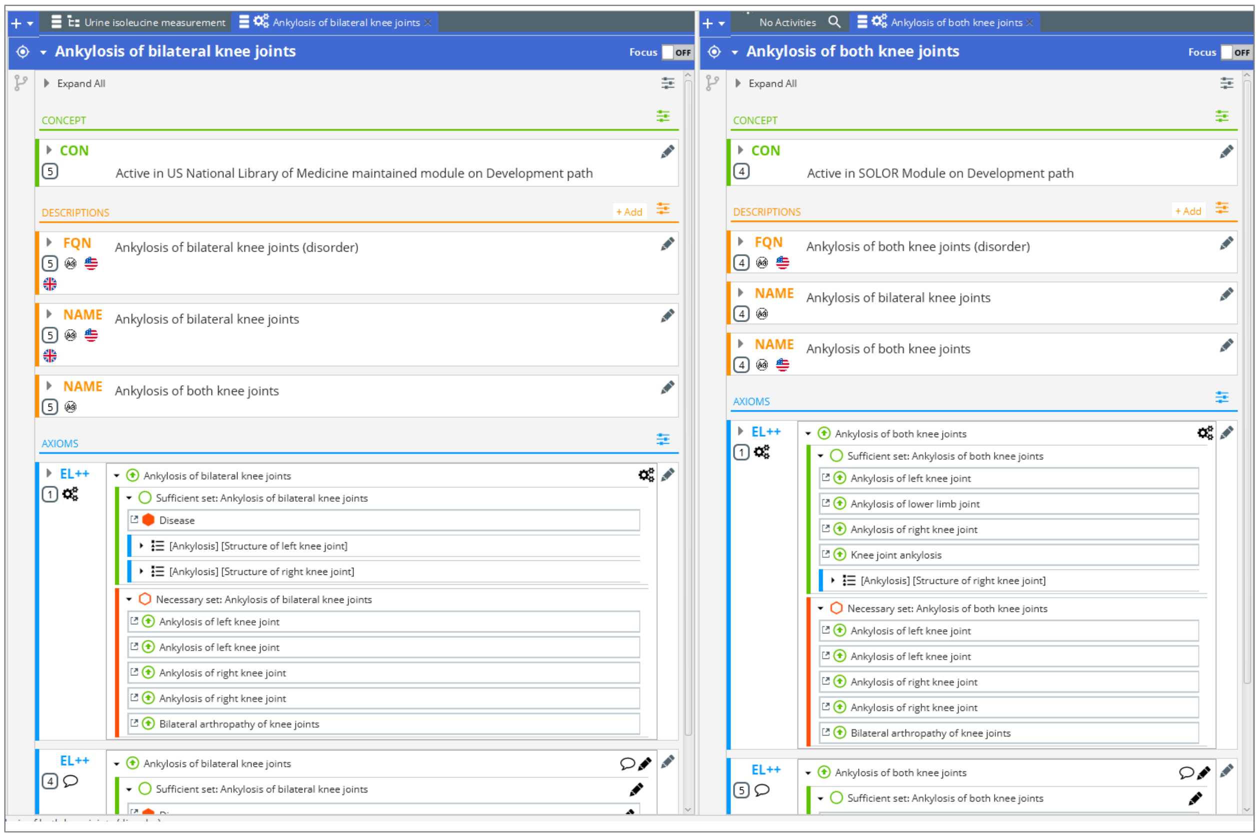
- Cycles: none
- Orphans: 48
- Equivalencies: 208
  - Set of 2 containing: Ankylosis of both knee joints (disorder)
    - Ankylosis of bilateral knee joints (disorder)
    - Ankylosis of both knee joints (disorder)
  - Set of 2 containing: Ankylosis of left hip joint (disorder)
  - Set of 2 containing: Ankylosis of left knee joint (disorder)
  - Set of 2 containing: Ankylosis of right hip joint (disorder)
  - Set of 2 containing: Ankylosis of right knee joint (disorder)
  - Set of 2 containing: Atrial Fibrillation Sotalol Hydrochloride 80 MG Oral Tablet (F
  - Set of 2 containing: Atrial Fibrillation Sotalol Hydrochloride 120 MG Oral Tablet (
  - Set of 2 containing: Atrial Fibrillation Sotalol Hydrochloride 160 MG Oral Tablet (
  - Set of 2 containing: Bacillus coagulans 10000000000 UNT Oral Capsule (RxNorm
  - Set of 2 containing: Benzocaine 100 MG/ML Topical Solution (RxNorm)
  - Set of 2 containing: Benzocaine 200 MG/ML Mucous Membrane Topical Solutio
  - Set of 2 containing: Betamethasone 0.0005 MG/MG Topical Gel (RxNorm)
  - Set of 2 containing: Betamethasone 0.0005 MG/MG Topical Ointment (RxNorm)
  - Set of 2 containing: Budesonide 0.032 MG/ACTUAT Metered Dose Nasal Spray (
  - Set of 2 containing: Calcium.ionized [Mass or Moles/volume] in Serum or Plasm
  - Set of 2 containing: Calcium.ionized [Mass/volume] in Blood
  - Set of 2 containing: Calcium.ionized [Moles/volume] in Blood
  - Set of 2 containing: Calcium.ionized [Moles/volume] in Capillary blood
  - Set of 2 containing: Calcium.ionized [Moles/volume] in Cord blood
  - Set of 2 containing: Calcium.ionized [Moles/volume] in Serum or Plasma
  - Set of 2 containing: Calcium [Mass/volume] in Serum or Plasma
  - Set of 2 containing: Calcium [Moles/volume] in Dialysis fluid
  - Set of 2 containing: Calcium [Moles/volume] in Venous blood
  - Set of 2 containing: Chordoma of lumbar spine (disorder)
  - Set of 2 containing: Chordoma of thoracic spine (disorder)
  - Set of 2 containing: Choriogonadotropin.beta subunit [Multiple of the median]
  - Set of 2 containing: Contact dermatitis caused by plants (disorder)
  - Set of 3 containing: Counseling for postnatal depression (procedure)
  - Set of 2 containing: Cromolyn Sodium 5.2 MG/ACTUAT Metered Dose Nasal Spr
- Inferred changes: 21
- Stamp coordinate
- Logic coordinate
- Edit coordinate

classification selected: Ankylosis of both knee joints (disorder)

*Equivalences Detected by Classifier*

**Figure 4.22. Equivalences Detected by Classifier**

The two Concept Detail Panels can be opened and viewed as two Concepts side by side to understand the differences and similarities between the two, their origins, and understand how to correct the equivalencies between these two Concepts. In this case, the Concept "Ankylosis of both knee joints (disorder)" would need to be retired as it belongs to the Solor Extension and the "Ankylosis of bilateral knee joints (disorder)" Concept is contained in the U.S. Edition.



Urine isoleucine measurement | Ankylosis of bilateral knee joints

**Ankylosis of bilateral knee joints** Focus OFF

Expand All

CONCEPT

CON 5 — Active in US National Library of Medicine maintained module on Development path

DESCRIPTIONS + Add

FQN 5 — Ankylosis of bilateral knee joints (disorder)

NAME 5 — Ankylosis of bilateral knee joints

NAME 5 — Ankylosis of both knee joints

AXIOMS

EL++ 1
- Ankylosis of bilateral knee joints
  - Sufficient set: Ankylosis of bilateral knee joints
    - Disease
    - [Ankylosis] [Structure of left knee joint]
    - [Ankylosis] [Structure of right knee joint]
  - Necessary set: Ankylosis of bilateral knee joints
    - Ankylosis of left knee joint
    - Ankylosis of left knee joint
    - Ankylosis of right knee joint
    - Ankylosis of right knee joint
    - Bilateral arthropathy of knee joints

EL++ 4
- Ankylosis of bilateral knee joints
  - Sufficient set: Ankylosis of bilateral knee joints

No Activities | Ankylosis of both knee joints

**Ankylosis of both knee joints** Focus OFF

Expand All

CONCEPT

CON 4 — Active in SOLOR Module on Development path

DESCRIPTIONS + Add

FQN 4 — Ankylosis of both knee joints (disorder)

NAME 4 — Ankylosis of bilateral knee joints

NAME 4 — Ankylosis of both knee joints

AXIOMS

EL++ 1
- Ankylosis of both knee joints
  - Sufficient set: Ankylosis of both knee joints
    - Ankylosis of left knee joint
    - Ankylosis of lower limb joint
    - Ankylosis of right knee joint
    - Knee joint ankylosis
    - [Ankylosis] [Structure of right knee joint]
  - Necessary set: Ankylosis of both knee joints
    - Ankylosis of left knee joint
    - Ankylosis of left knee joint
    - Ankylosis of right knee joint
    - Ankylosis of right knee joint
    - Bilateral arthropathy of knee joints

EL++ 5
- Ankylosis of both knee joints
  - Sufficient set: Ankylosis of both knee joints

*SNOMED CT® Equivalences between Modules*

**Figure 4.23. SNOMED CT® Equivalences between Modules**

# 5. Next Steps

Tinkar is a logical model and therefore may be implemented using relevant implementable models and technology. This section will focus on the use of Tinkar alongside preexisting HL7 exchange standards. In practice, the Tinkar logical model can be used to create practical implementation guidance in the future (e.g., implementation guides, profiles, value sets), and can be applied to design change management solutions.

## 5.1. Tinkar Serialization Data

Tinkar will include a serialization format that will serialize any or all the information stored in a Tinkar implementation. This serialization format will allow third parties to read data created by a Tinkar compliant implementation. A serialized Tinkar file can also be used to transfer information between different Tinkar implementations. This will ensure that no Tinkar data is lost.

C# and Java code implementations of the Tinkar Data Model will be made available to integrators, allowing third parties to more easily use Tinkar.

Specifically, third party implementers can use Java or C# to:

1. Read Tinkar data into memory.

2. Manipulate and modify Tinkar data using Tinkar data structures.

3. Write Tinkar data out to a serialized file.

## 5.2. Tinkar interfacing with FHIR

Tinkar data can be used to create FHIR compliant data. Tinkar data can be used to generate FHIR data. Only a subset of the Tinkar data model would be converted to FHIR. Roundtripping of data from Tinkar to FHIR and back is likely to result in Tinkar data loss.

Tinkar data items lost in FHIR conversion:

1. STAMP data: Loss of the STAMP data means that the ability to look at terminology at a specific point in time is lost. The FHIR data is a snapshot at a fixed point in time.

2. Semantic data: FHIR data does not contain a concept similar to Tinkar’s semantic relationships.

There are two main areas in which Tinkar and FHIR are expected to interface, as described in the subsequent sections: "Dynamic FHIR Terminology Servers" and "Static FHIR CodeSystem and ValueSet Resources."

### 5.2.1. Dynamic FHIR Terminology Servers

FHIR terminology servers are designed to provide knowledge artifacts in real time to support clinical system operations. A system may request $expand on a ValueSet, for instance, to populate a drop-down list in a user interface, or $validate-code to confirm content in an interface conforming to a profile; a request for $subsumes on a CodeSystem or $translate on a ConceptMap can be used to translate codes when transforming native records to a standard format, and so forth.

FHIR queries against the FHIR terminology server could use Tinkar originated terminology data as well as other non-Tinkar generated terminology to fulfill the requests.

The serialization file described above would be one way to import data from a Tinkar system to a FHIR terminology server. Implementers for each FHIR Terminology Server would have to write code specific to their implementation to import Tinkar data into their own implementation dependent data storage.

The Tinkar C# and Java libraries could be used by the FHIR Server Implementors to help support that import process.

### 5.2.2. Static FHIR CodeSystem and ValueSet Resources

FHIR CodeSystem and ValueSet records are designed to provide knowledge artifacts to support clinical system operations. Terminology data contained in FHIR CodeSystem and ValueSet resources are fixed at the time that the resource instances are generated.

It is expected that some automated means of creating FHIR ValueSet and CodeSystem records from Tinkar will be developed. The automated system could import data from a Tinkar system using the serialization format described above, and output desired CodeSystem and ValueSet instances. Modifications to the Tinkar data would require regeneration of the FHIR artifacts and subsequent transmission to interested users of those resource instances.

## 5.3. Tinkar interfacing with HL7 UTG, THO, HTA

The three HL7 processes for maintaining and distributing HL7 terminology content are UTG, THO, HTA. Our understanding of these HL7 Areas:

- **Unified Terminology Governance (UTG)**: A process with associated tooling to support the process strictly for governance of HL7 terminological content. UTG is the maintenance process to keep it up to date and manage change.
- **HL7 terminological content (THO)**: HL7 Terminological content refers to terminology.hL7.org (THO) which is the domain address of the entry point for the published content. HL7 terminology or THO refers to an enormous corpus of coded content that varies in quality and that has been developed in HL7 for about 30 years.
- **HL7 Terminology Authority (HTA)**: The HTA’s primary purpose is to deal memorandum of understanding and publication interfaces with non-HL7 publishers and maintainers of terminological content.

The aim of these HL7 groups, to provide standardized HL7 terminology in a consistent format, is consistent with the goal of Tinkar.

If HL7 so desired, Tinkar could be used as the format by which all HL7 terminology is exported. It is believed that current HL7 content can be converted to Tinkar with no loss. Further work would need to be done to verify that if HL7 was interested in that path.

Conversion of HL7 common terminology data to Tinkar would allow:

1. Use of tools like KOMET to import, export, maintain and validate HL7 terminology data.
2. Use of Tinkar serialization format for serialization and distribution format.
3. Use of Tinkar and third-party tooling to convert Tinkar serialized data to FHIR records and import to FHIR terminology servers.

# 6. Key Takeaways

1. Terminology Knowledge Architecture (Tinkar) intends to integrate clinical terminology and local concepts to support increased data quality for interoperable clinical information. Quality clinical data enables healthcare systems across the enterprise to conduct robust and meaningful data analysis and increase overall interoperability, which ultimately enhances quality of care across all medical facilities.

2. A Tinkar specification provides a standardized model for terminology and change management.

3. The Tinkar Reference Model could provide a single representation for all terminologies required in the U.S. (e.g.,SNOMED CT®, LOINC® , RxNorm).

4. Tinkar provides the foundation of a knowledge architecture that delivers integrated terminology to the enterprise and its information systems. In doing so, it addresses the differences in management and structure across reference terminology, local concepts, and code lists/value sets.

5. Tinkar aims to be both self-describing and completely machine processed:

    a. Self-describing machine-readable representation of terminology, such that if an application can process the metadata, it should be able to import the content/concepts and make it available to enterprise applications.

    b. The machine-readable terminology could generate human-readable documentation so that business analysts and developers can understand and apply it correctly.

6. If HL7 so desired, Tinkar could be used as the format by which all HL7 terminology is exported. It is believed that current HL7 content can be converted to Tinkar with no loss; further work would need to be done to verify that if HL7 was interested in that path.

7. In practice, the Tinkar logical model can be used to create practical implementation guidance in the future (e.g., implementation guides, profiles, value sets), and can be applied to design change management solutions.

# A. References

1. Bodenreider O, Cornet R, Vreeman DJ. Recent Developments in Clinical Terminologies - SNOMED CT, LOINC, and RxNorm. Yearbook Med Inform. 2018 Aug;27(1):129-139. doi: 10.1055/s-0038-1667077. Epub 2018 Aug 29. PMID: 30157516; PMCID: PMC6115234.Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6115234/.

2. Capability Maturity Model Integration (CMMI) for Development, Version 1.3. Software Engineering Institute; 2010. Available from: https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=9661.

3. Institute of Medicine (2012). Health IT and Patient Safety. Building Safer Systems for Better Care. Washington, DC: National Academies Press.

4. Cimino JJ. Desiderata for controlled medical vocabularies in the twenty-first century. Methods of information in medicine. 1998;37(4-5):394-403.https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3415631/.

5. NHS England. QRISK2 update (identified code mapping errors). SystmOne National User Group; 2016. Available from: https://www.systmoneusergroup.co.uk/qrisk2-update/.

6. Iacobucci G. Computer error may have led to incorrect prescribing of statins to thousands of patients. BMJ. 2016 May 13;353:i2742. doi: 10.1136/bmj.i2742. PMID: 27178396.

7. MHRA information on TPP and QRISK®2. GOV.UK. Medicines and Healthcare products Regulatory Agency; 2016. Available from: https://www.gov.uk/government/news/mhra-information-on-tpp-and-qrisk2.

8. Clinical decision support malfunctions are widespread and persistent, AMIA says. Healthcare IT News. 2016. Available from: https://www.healthcareitnews.com/news/clinical-decision-support-malfunctions-are-widespread-and-persistent-amia-says.

9. Tharakan V. Metadata Driven Architecture For Application Development. ClaySys Technologies; 2019. Available from: https://www.claysys.com/blog/metadata-driven-application-development/.

10. Analysis Normal Form Informative Ballot. HL7 CIMI Work Group. Sept 2019. http://www.hl7.org/documentcenter/public/ballots/2019SEP/downloads/HL7_CIMI_LM_ANF_R1_I1_2019SEP.pdf. mirror link: http://solor.io/wp-content/uploads/2020/01/ANF-Ballot-Jan-2020.pdf.

11. Abstract syntax tree. Wikipedia; 2020. Available from: https://en.wikipedia.org/wiki/Abstract_syntax_tree.

# B. Related Works

1. Campbell KE. Distributed Development of a Logic-based Controlled Medical Terminology. Stanford University. June 1997.

2. Campbell KE, Das AK, Musen MA. A logical foundation for representation of clinical data. J Am Med Inform Assoc. 1994 May-Jun;1(3):218-32. doi: 10.1136/jamia.1994.95236154. PMID: 7719805; PMCID: PMC116201.

3. Spackman KA, Campbell KE, Côté RA. SNOMED RT: a reference terminology for health care. Proc AMIA Annu Fall Symp. 1997:640-4. PMID: 9357704; PMCID: PMC2233423.

4. Campbell KE, Cohn SP, Chute CG, Shortliffe EH, Rennels G. Scalable methodologies for distributed development of logic-based convergent medical terminology. Methods Inf Med. 1998 Nov;37(4-5):426-39. PMID: 9865041.

5. Bakken S, Campbell KE, Cimino JJ, Huff SM, Hammond WE. Toward vocabulary domain specifications for health level 7-coded data elements. J Am Med Inform Assoc. 2000 Jul-Aug;7(4):333-42. doi: 10.1136/jamia.2000.0070333. PMID: 10887162; PMCID: PMC61438.

6. Campbell KE, Oliver DE, Shortliffe EH. The Unified Medical Language System: toward a collaborative approach for solving terminologic problems. J Am Med Inform Assoc. 1998 Jan-Feb;5(1):12-6. doi: 10.1136/jamia.1998.0050012. PMID: 9452982; PMCID: PMC61272.

7. Campbell KE, Giannangelo K. Language barrier. Getting past the classifications and terminologies roadblock. J AHIMA. 2007 Feb;78(2):44-6, 48. PMID: 17366992.

8. Campbell KE, Musen MA. Representation of clinical data using SNOMED III and conceptual graphs. Proc Annu Symp Comput Appl Med Care. 1992:354-8. PMID: 1482897; PMCID: PMC2248067.

9. Chute CG, Cohn SP, Campbell KE, Oliver DE, Campbell JR. The content coverage of clinical classifications. For The Computer-Based Patient Record Institute's Work Group on Codes & Structures. J Am Med Inform Assoc. 1996 May-Jun;3(3):224-33. doi: 10.1136/jamia.1996.96310636. PMID: 8723613; PMCID: PMC116304.

10. Bakken S, Parker J, Konicek D, Campbell KE. An evaluation of ICNP intervention axes as terminology model components. Proc AMIA Symp. 2000:42-6. PMID: 11079841; PMCID: PMC2244103.

11. Spackman KA, Reynoso G. Examining SNOMED from the perspective of Formal Ontological Principles: Some Preliminary Analysis and Observations. Proc. KR-MED, 2004: 72-80.

# C. Tinkar Model Concepts

The Tinkar core architecture is designed to be extremely flexible. This flexibility supports the ability to assert relationships among diverse terminologies and to support change in the design of those terminologies. However, it means that there may be more than one way to support any specific requirement. We propose the following members of the Tinkar Model Concepts (a.k.a. infrastructure concepts).

## C.1. Concepts for Semantic Fields

1. Identifier Source: An identifier used to label the identity of a unique component.
    a. UUID (Universally unique identifier): A 128-bit number used to identify information in computer systems.
2. Status: Concept that organizes the statuses of a component.
    a. Active: Concept used to represent a status for components that are active.
    b. Inactive: Concept used to represent a status for components that are no longer active.
    c. Canceled: Concept used to represent a status for components that are canceled.
    d. Primordial: Concept used to represent a status for components that have not yet been release and exist in their most basic form.
    e. Withdrawn: Concept used to represent a status for components that are withdrawn.
3. Time: Specifies the inclusive date and effectiveTime for various statuses of the component.
    a. As far as a model concept to support STAMP versioning, time is left to implementation-specific details where various frameworks could be used to represent time (e.g., Epoch, Unix time representation).
4. Author: Concept used to identify the author of a component.
    a. Default Author: An agent that may be responsible for authoring a change.
5. Module: Modules are used to organize content for maintenance and publication purposes.
    a. Tinkar Module: All concepts, semantics, and Patterns must belong to a module. A Tinkar Module would contain a release package of all content published together. A module can be dependent upon another module.
6. Path: A set of components under version control that can be managed distinctly from other components. Paths “branch” from other paths as established, and can be “merged” with other paths as well.
    a. Prerequisite path: The path that contains components that are shared across all paths and contain infrastructure components.
    b. Development path: A path that specifies that the components are currently under development.
    c. Master path: A default path for components.
    d. Sandbox path: A path for components under testing.

## C.2. Concepts for Descriptions, Language, and Dialect

1. Description type: Descriptions associate human-readable terms with the concepts that they describe.

    a. Fully qualified name description type: Fully qualified name is a description that uniquely identifies and differentiates it from other concepts with similar descriptions.

    b. Regular name description type: There may be descriptions/synonyms marked as "regular."

    c. Definition: An expression of the meaning of a term.

    d. Abbreviation: A shortened form of a term.

2. Description case significance: Identifies the case significance of descriptions.

    a. Case sensitive: Differentiating between capital and lowercase letters.

    b. First char case sensitive: First character of a string is uppercase.

    c. Not Applicable: Description case significance is not specified and therefore not applicable.

    d. Not case sensitive: Uppercase or lowercase characters can be entered.

3. Description acceptability: Identifies the acceptability of a description.

    a. Acceptable: Specifies that a description is acceptable, but not preferred within a language or dialect.

    b. Preferred: Specifies that a description is preferred within a language or dialect. There will be one preferred description for each description type.

4. Language: Specifies the language of the description text.

    For example:

    a. Chinese language

    b. Czech language

    c. Danish language

    d. Dutch language

    e. English language

    f. French language

    g. German language

    h. Irish language

    i. Italian language

    j. Japanese language

    k. Korean language

   l. Lithuanian language
   m. Polish language
   n. Russian language
   o. Spanish language
   p. Swedish language

5. Dialect: Specifies the dialect of the language.

   For example:

   a. GB English dialect
   b. U.S. English dialect
      i. U.S. Nursing dialect
   c. Spanish dialect
      i. Latin American Spanish dialect
   d. French dialect
   e. Korean dialect
      i. Standard Korean dialect
   f. Polish dialect
   g. Irish dialect
   h. Czech dialect
   i. Russian dialect

6. Membership type: Used to identify the different types of memberships associated with components.
   a. Marked parent: A membership type that identifies 1 or more parent for a component. In a particular navigation view, this is a parent of a member.

7. Literal value: A literal is any number, text, or other information that directly represents a value.
   a. Boolean literal: A datatype that represents either true or false.
   b. Float literal: A datatype that represents numbers that have a decimal point or an exponential part.
   c. Instant literal: A datatype that represents a specific moment on the time line. The instant is defined as an offset since the origin (called an epoch).
   d. Integer literal: A datatype that represents an integer value.
   e. String literal: A datatype that represents a string value.

8. Precedence: Precedence determines the order in which the operators in an expression are evaluated.

   a. Path precedence: Determines the order in which the path is evaluated.

   b. Time precedence: : Determines the order in which the time is evaluated.

9. Relationship type in source terminology: the associations that there exist between the meanings of terms from source terminologies (e.g., 'IsA').

# C.3. Concepts for Description Logic

Below is an example of how Tinkar represents Description Logic concepts such as those represented by OWL EL ++. A similar approach can be taken for other standards, such as BPM+ Health. Tinkar Model Concepts include a concept for each logical operator in the EL profile; for other standards such as BPM+, Tinkar could include concepts for various nodes.

1. EL profile set operator

   a. Necessary set: A set of relationships that is always true of a concept. A concept that only contains necessary conditions is considered primitive.

   b. Property set: A set of relationships that are non-defining and are not used by a description logic classifer.

   c. Sufficient set: A set of relationships that differentiate a concept and its subtypes from all other concepts. A concept that contains at least one set of necessary and sufficient conditions is considered defined.

2. Connective operator

   a. And: An operator used to assert that two (or more) parts of a concept definition or expression constraint must both be true.

   b. Abstract Definition root: This is a conceptual parent of all specific definition root concepts, such as 'OWL EL Definition root'. All specific definition root instances can be found by searching for children of this concept.

   c. OWL EL Definition root: In an OWL EL++ property graph, the meaning field of the root node will always be the 'OWL EL Definition root' concept. This identifies that a property graph instance describes an OWL EL++ description logic graph. The OWL EL Definition root is a 'child of' the 'Abstract Definition root' concept.

   d. Disjoint with: An operator that asserts that two concepts are disjoint and cannot have common objects. By default, all concepts are assumed to have potential overlapping classes unless they are explicitly stated to not have them.

   e. Or: An operator used to assert that at least one of two (or more) parts of a concept definition or expression constraint must be true.

3. Logical feature: Description logic features

   a. Inverse property: Any logic property that is used to link two objects can have an inverse. For example, the property IsA can have the inverse property of hasChild.

   b. Transitive property: A logic property is transitive if a property that relates object a to object b and object b to object c, can then also be used to infer that object a is also related to object c. For example, the PartOf relationship would be an example of a transitive property.

4. Role operator: Concept that is used to describe universal vs existential restrictions.

   a. Existential restriction (Some): Existential restrictions describe objects that participate in at least one relationship along a specified property to objects of a specified class.

   b. Universal restriction (All): Universal restrictions constrain the relationships along a given property to concepts that are members of a specific class.

5. Role

   a. Role Group: Used to represent a role group similar to the ones found in SNOMED CT.

6. Logic origin: Origin of the logic form

   a. EL++ inferred form: A representation of EL++ concept definitions that is logically derived by applying a description logic classifier to the EL++ stated form.

   b. EL++ stated form: A representation of EL++ concept definitions consisting only of assertions made or revised by terminology authors.

7. Concrete domain operator: Operators used when specifying a concrete domain.

   a. Equal to: Value is equal to the concrete domain

   b. Greater than: The value should be greater than the concrete domain

   c. Greater than or equal to: The value should be greater than or equal to the concrete domain

   d. Less than: The value should be less than the concrete domain

   e. Less than or equal to: The value should be less than or equal to the concrete domain

8. Description-logic classifier: Concept used to organize the available classifier used within Tinkar. For example:

   a. SnoRocket classifier: Concept used to identify the SnoRocket classifier

   b. ELK classifier: Concept used to identify the ELK classifier

9. Description-logic profile: Used to identify the Description logic profile

   a. EL++ profile: Represents the EL++ profile

10. Navigation Operator: Provides a navigation abstraction for specifying description logic.

   a. Parent of: Concept used to represent that a Concept is a parent of another Concept.

   b. Child of: Concept used to represent that a Concept is a child of another Concept.

11. Property pattern implication: Property pattern implications are used to represent the description logic feature called property chains. Property chains are rules that allows you to infer the existence of a property from a chain of properties. The current release of SNOMED CT includes one property chain:

    363701004 |direct substance| $_{o}$ 127489000 |has active ingredient| $_{\rightarrow}$ 363701004 |direct substance|

12. Sufficient concept definition operator:

   a. Necessary but not sufficient concept definition: Concept used to identify concepts that have necessary, but not sufficient definitions.

   b. Sufficient concept definition: Concept used to identify concepts that contain sufficient definitions.

## C.4. Patterns

1. Concept: An identifier for a concept or instance. The identifier contains no information; all information specifying or describing the concept is asserted with Semantics.
2. Pattern: A class defining a set of predicates and object types that can be asserted about a class of subjects. All Semantics follow Patterns.
3. Semantic: A class containing a set of predicates and objects about a subject. Semantics perform the descriptive work in Tinkar.
4. Concept chronology: The set of versions having a STAMP for a concept. A concept identifier specifies a ConceptChronology; specifying a ConceptVersion requires a rule or parameter for selecting among STAMP values.
5. Pattern chronology: The set of versions having a STAMP for a pattern.
6. Semantic chronology: The set of versions having a STAMP for a Semantic.
7. Concept version: A single instance of an identifier for a concept with a STAMP.
8. Pattern version: A single instance of a pattern with a STAMP.
9. Semantic version: A single instance of a Semantic with a STAMP.
10. String: A sequence of characters, either as a literal constant or as a variable. Strings could be used to represent terms from code systems or URLs, textual definitions, etc.
11. Integer: Data type that represents some range of mathematical integers.
12. Float: Represents values as high-precision fractional values.
13. Boolean: Represents the values true and false.
14. Byte array: Represents the values true and false.
15. Instant: Models a single instantaneous point on a timeline.
16. Component id list: An ordered list of Component IDs.
17. Component id set: An unordered list of Component IDs.
18. Planar point: Position in a two-dimensional space (a plane).
19. Spatial point: Position in a three-dimensional space.
20. UUID (Universally unique identifier): A 128-bit number used to identify information in computer systems.
21. Property graph: A type of graph model where data are organized as nodes, relationships, and properties (data stored on the nodes or relationships).
22. Directed Tree or Ditree - a graph obtained from an undirected tree by replacing each undirected edge by two directed edges with opposite directions.
23. DiGraph: A graph in which a set of objects are connected where all the edges are directed from one vertex to another.
24. Vertex - The fundamental unit of data that makes up a graph or tree.

25. In Tinkar, property graphs are used as a general-purpose data pattern to represent an abstract syntax tree (AST), such as OWL EL++. This allows for data types without requiring custom nodes.

    i. An AST may be used "during semantic analysis, where the compiler checks for correct usage of the elements of the program and the language. The compiler also generates symbol tables based on the AST during semantic analysis. A complete traversal of the tree allows verification of the correctness of the program. After verifying correctness, the AST serves as the base for code generation. The AST is often used to generate an intermediate representation, sometimes called an intermediate language, for the code generation." [11]

    ii. An AST is made up of nodes and branches. In Tinkar, every tree will always have roots, but they are specific: "An OWL EL root" vs. a "BPMN root", etc. Each node must have 0 or more children.

26. Here is an example of Tinkar output of semantics that reference multiple concepts.

```
Field 1: [
EL++ Stated terminological axioms: DiTreeEntity{
   [0]→[8] Definition root
      [8]→[7] Necessary set
         [7]→[5,1,6] And
            [5]→[4] Role type: Role group
               •Role operator: Existential restriction
               [4]→[3] And
                  [3]→[2] Role type: Is modification of
                     •Role operator: Existential restriction
                     [2] Concept reference: Acetylsalicylic acid
            [1] Concept reference: Salicylate
            [6] Concept reference: Antiplatelet agent
}]
```

27. In this output, one can see a sufficient set and necessary set. Bulleted items are properties in the node. The output is printed as a "depth first search." Each depth adds 3 characters of padding and shows how OWL EL++ definitions, using only terminology and a standard property graph data structure, are represented. The 1st one is node index 0 which has a child of node index 8. Node index 0 is the OWL EL++ definition root. Node 8 points to Node 7, and the meaning of Node 8 is that it is a necessary set. Node 7 is 'And' and points to Node 5,1,6. Node 5's meaning is 'Role Type', Value is 'Role group', and its other property is 'Role Operator.' Node 5 points to Node 4. Node 4 is 'And.' Node 3 is 'Role Type.' Node 2 is Concept Reference. 7 also points to 1 and 6 (Concept References).

28. The property graph model demonstrates that each vertex has a meaning. Tinkar can use concepts to represent anything end users might need at nodes. This allows for data types without requiring custom nodes. With no changes to the underlying data structures, Tinkar can represent more than OWL EL++. With updates to terminology, Tinkar can represent any parsable standard, such as BPMN and DMN, using this property graph model and a proper set of terminology concepts and semantics represented using Tinkar.

## C.5. Concepts for Copyright & Content License

1. Copyright: A type of intellectual property that gives its owner the exclusive right to make copies of a creative work, usually for a limited time.

    a. SNOMED CT License Agreement: License Agreement between The International Health Terminology Standards Development Organisation and the person or organization to which the International Release of SNOMED CT (whether on its own or as part of a Member’s National Release of SNOMED CT) is distributed or otherwise made available.

   b. Regenstrief Public License: License Agreement between Regenstrief Institute, Inc. for LOINC to be made available for use, modification, and distribution in accordance with the Regenstrief Public License Version.

   c. HL7 License: HL7 licenses its standards and select IP free of charge. HL7 licenses describe the permitted uses of materials.

2. Content License: Official permission to use intellectual property.

   a. Apache 2 license: A permissive free software license written by the Apache Software Foundation.

   b. Creative Commons BY license: A license used when an author wants to give other people the right to share, use, and build upon a work the author created.

3. US Government Work: A work created by U.S. federal government employees, that is in the public domain.