 <small>for health and social care</small>	<b>LRA - Terminology Binding Technical Specification</b>			
	<b>Programme</b>	Informatics Data Standards Programme	<b>Document Record ID Key</b>	
	<b>Sub-Prog / Project</b>	Logical Record Architecture for Health and Social Care	NPFIT-FNT-TO-DPM-0984.01	
	<b>Prog. Director</b>	D. Perry	<b>Status</b>	Draft
	<b>Owner</b>	S. Bentley	<b>Version</b>	V0.6
	<b>Lead author/Editor</b>	D. Markwell	<b>Version Date</b>	2009-06-12

## Logical Record Architecture for Health and Social Care Terminology Binding Technical Specification

**Amendment History:**

Version	Date	Amendment History
0.1	2008-10-06	Draft outline with some additional text based on earlier document from NHS CFH EHR TAG activities on terminology binding.
0.2	2008-12-05	Internal revision – partial document not released.
0.3	2008-12-23	Internal revision – partial document not released.
0.4	2009-01-16	First full draft release.
0.5	2009-03-10	Second full version
0.6	2009-06-12	Final version

**Report contributors:**

Name	Title / Responsibility
D. Markwell	Principal Consultant, The Clinical Information Consultancy Ltd

**Reviewers:**

Technical Review Group:

Name	Comments received from
Members of the LRA Generic Design Group Members of the NHS CFH Electronic Health Record Content Technical Advisory Group	Yong Gao, Robert Challen, John Arnett

**Approvals:**

Name	Title / Responsibility	Approval Status	Date	Version
S. Bentley	Acting Head, Logical Record Architecture			
G. McIntosh	Head of Programme Office, NHS Data Standards & Products			
K. Lunn	Head, Data Standards & Products			

**Distribution:**

Once approved, this document will be publicly available.

**Status**

The current document is the final draft version. It takes account of comments received up to 2009-05-31.

The XML schema and related documentation files are part of this deliverable and should be distributed with it.

1	Introduction.....	5
1.1	About this document.....	5
1.2	Other documents.....	5
2	Naming conventions.....	6
3	Integrated Terminology Binding Development.....	10
3.1	Requirement.....	10
3.2	Background.....	10
3.3	Adding terminology to existing information structures.....	10
3.4	Structural approaches to facet representation.....	12
3.5	Recommended approach to alignment.....	13
4	Terminology Binding Representation - Requirements.....	15
4.1	Overview.....	15
4.2	Requirements for different types of terminology binding.....	16
5	Representing expression constraints.....	20
5.1	Applicability of expression constraints.....	20
5.2	Expression constraint characteristics.....	20
5.3	Abstract model of expression constraints.....	22
5.4	Expression constraint – persistent representation.....	28
5.5	Relationship to other representations.....	36
5.6	Binding expression constraints to information model artefacts.....	38
6	Terminology binding tooling requirements.....	41
6.1	Introduction.....	41
6.2	Care record design – foundations and development.....	41
6.3	Terminology related tooling requirements to support design.....	44
6.4	General terminology server functions.....	46
6.5	<i>Expression constraint</i> editor.....	47
6.6	<i>Expression constraint</i> repository.....	48
6.7	Version management.....	49
6.8	<i>Expression constraint</i> validator.....	50
6.9	<i>Expression</i> example editor.....	51
6.10	<i>Expression</i> validator.....	51
6.11	<i>Expression constraints</i> and human-readable documentation.....	52
6.12	Terminology mediated access to care record designs.....	53
6.13	Terminology binding and other LRA tooling requirements.....	57
6.14	Implementation aspects of bindings.....	58
Appendix A	Expression Constraint Types - Details and Examples.....	59
A.1	Semantic expression constraints.....	59
A.2	Literal expression constraints.....	60
A.3	Semantic and literal expression constraints compared.....	60
Appendix B	Additional Terminology Binding Requirements.....	62
B.1	Information model artefact representation requirements.....	62

---

B.2	Representation of constructor bindings .....	63
B.3	Representation of retrieval bindings .....	63
Appendix C	Representing Expression Constraints – Option Review .....	64
C.1	Introduction .....	64
C.2	Extended Compositional Grammar .....	64
C.3	Existing Description Logic representations.....	64
C.4	SNOMED CT Machine Readable Concept Model.....	65
C.5	A specific XML based format aligned with the abstract model .....	66
C.6	Evaluation of options .....	66
C.7	Recommended approach .....	67
Appendix D	Terminology Binding - Location and Referencing .....	68
D.1	Possible terminology binding locations.....	68
D.2	Recommended form for referencing terminology bindings .....	69
D.3	Dereferencing representations for optimisation .....	70
D.4	Other approaches.....	70
Appendix E	Extended Compositional Grammar.....	71
Appendix F	Care Components Model.....	78

# 1 Introduction

## 1.1 About this document

This document includes technical material relevant to supporting the development, maintenance and implementation of *terminology binding* as part of the Logical Record Architecture for Health and Social Care (LRA).

The first part (Section 4 and 5) specify a (proposed) machine processable representation of *terminology bindings*. This covers the representation of different types of *terminology binding*. It also specifies a mechanism for referencing these constraints from the *LRA Care Components* model and other *information model artefacts*.

The second part (Section 6) outlines functional requirements for tools to support the creation, maintenance and application of *terminology bindings*.

## 1.2 Other documents

This document is based on and refers to the following pre-existing documents.

- Terminology Binding Requirements and Principles.
  - Describes the requirements for *terminology binding*, to minimise ambiguity and thus to maximise the reusability of clinical data.
  - Explains principles that form the foundation for a coherent approach to *terminology binding*.
  - Summarises the relative strengths of structure and terminology in respect of representing particular aspects of meaning.
  - Categorises different structural and semantic units and different types of *terminology binding*.
  - Summarises and references other relevant material and notes some issues and outstanding challenges.
- SNOMED CT publications (dated 2008-07-31)
  - SNOMED CT User Guide
  - SNOMED CT Technical Reference Guide
  - SNOMED CT Technical Implementation Guide
  - SNOMED CT Transforming Expressions to Normal Forms
  - SNOMED CT Abstract Logical Models and Representational Forms

This document also refers to the following documents that are subject to parallel LRA development activities<sup>1</sup>.

- SNOMED CT guidelines for representing Key Clinical Data and Common Recording Patterns
- Logical Record Architecture – Care Components Model

---

<sup>1</sup> Unless otherwise specified, references to these documents refer to the version current at the time of the release of this document.

## 2 Naming conventions

This document uses the following naming conventions which are subject to change to align with other LRA materials.

**Table 1. General naming of artefacts and components**

Information model artefact	A complete static information model, any subsidiary part of such a model (including abstract and concrete classes and <i>fields</i> ) and any formally specified constraints applied to any part of such a model.
Terminology binding (noun)	An instance of a link between a terminology component and an information model artefact.
Terminology binding (verb)	The process or action of making one or more terminology bindings.
Terminology component	A complete coding scheme, any subsidiary part of such a coding scheme or any reference, set of references to or constraints upon any such coding scheme or part of a coding scheme.

**Table 2. Specific naming of information model artefacts (subject to LRA discussions)**

Information model	A sharable, stable, and organised structural representation of information requirements. Includes reference models and constraints on those reference information models designed to meet specific use cases (e.g. HL7 CIM, DMIM, RMIM, Templates, EN13606 Archetypes and <i>openEHR</i> Templates).
Reference model	A general information model that provides a foundation for more specific constrained models (e.g. the HL7 Version 3 RIM and the EN13606 Reference Model). Specifically refers to the LRA Care Components model.
Constrained model	An information model derived by constraining a reference model to support a specified set of requirements (e.g. HL7 DMIM, RMIM, HMD, Template, EN13606 Archetype and <i>openEHR</i> Templates).
Class	A definition of the characteristics of particular types of objects as represented in an information model.
Field	<p>A characteristic of a class that can be represented as a data member in an object that instantiates that class.</p> <p>The data present in an instantiation of a field:</p> <ul style="list-style-type: none"> <li>➤ Is represented in accordance with a specified data type.</li> <li>➤ May be subject to specified constraints.</li> </ul>
<p>Notes:</p> <ul style="list-style-type: none"> <li>a) '<i>Field</i>' is used in preference to 'attribute' which has a more specific sense and may be confused with the SNOMED CT use of 'attribute'. Other names could be 'column' or 'slot'.</li> <li>b) This document is primarily concerned with <i>fields</i> for which one or more of the following are true: <ul style="list-style-type: none"> <li>i. The <i>field</i> may contain a SNOMED CT <i>expression</i>; or</li> <li>ii. The <i>field</i> may influence the interpretation of another <i>field</i> containing an <i>expression</i>;</li> <li>iii. The <i>field</i> may have a meaning equivalent to or subsumed by a SNOMED CT <i>expression</i>.</li> </ul> </li> </ul>	

**Table 3. Specific naming of terminology binding components**

Expression constraint	<p>A representation of a set of possible SNOMED CT <i>expressions</i>.</p> <p><i>Expression constraints</i> may be used to specify terminology bindings that limit the set of <i>expressions</i> that may be used in particular information model artefacts (e.g. in a field of a constrained class in the LRA Care Components model).</p> <p><i>Expression constraints</i> may also be used to specify query selection criteria.</p>
Semantic expression constraint	A type of <i>expression constraint</i> which limits the possible meanings that can be expressed by a field.
Literal expression constraint	A type of <i>expression constraint</i> which constrains the ways in which an expression may be constructed to represent a given meaning.
Selection support binding	A <i>terminology binding</i> that represents a range of commonly used values in a way that facilitates selection at a user interface (in clinical use or during design of more specific models or data entry protocols)
Retrieval binding	A <i>terminology binding</i> that indicates how data from one information model artefact may be used to populate another artefact. For example, to populate a check list entry from a relevant but more detailed clinical entry.
Constructor binding	A <i>terminology binding</i> that indicates how values entered in two or more related fields are combined to represent a composite meaning in a consistent form.

**Table 4. Names of SNOMED CT components (based on SNOMED CT glossaries)**

Concept	A clinical idea to which a unique <i>conceptId</i> has been assigned.
Description	A uniquely identified association between a Concept and a Term.
Relationship	A uniquely identified association between two <i>concepts</i> . The nature of the association is indicated by a third <i>concept</i> (the <i>RelationshipType</i> )
Relationship Group	A set of relationships that are logically associated with one another. For example, where a procedure includes two distinct actions applied to different target sites a separate <i>Relationship Group</i> is used to associate each action with the relevant site.
Subset / Refset	Sets of components (e.g. <i>concepts</i> , <i>descriptions</i> , etc) that are represented in a manner specified by SNOMED CT documents.  The main difference between a Subset and a Refset is that the latter conforms to a revised specification – from the perspective of this document they are interchangeable.
Term	A text string that may represent one or more Concepts.

**Table 5. Names used to describe aspects of the SNOMED CT concept model**

Concept Model	The complete set of rules that govern the ways in which SNOMED CT <i>concepts</i> are permitted to be modelled using <i>relationships</i> to other <i>concepts</i> . These rules also constrain the range of valid post-coordinated expressions.
Concept Model Constraint	A rule that must be fulfilled by the set of <i>modelled relationships</i> applied to a Concept that belongs to a specified <i>Concept Model Domain</i> .
Concept Model Domain	A set of SNOMED CT <i>Concepts</i> to which a common set of <i>Concept Model Constraints</i> apply.
Concept Model Attribute	A <i>Concept</i> that specifies the type of a <i>relationship</i> that can be used within the constraints of the <i>Concept Model</i> .
Concept Model Range	A specified set of SNOMED CT <i>concepts</i> from which the value of a defining relationship must be specified.
Machine Readable Concept Model (MRCM)	A representation of the <i>Concept Model</i> that can be read and processed by a computer.  The IHTSDO is currently developing a specification for the representation of the MRCM which is at the following URL <a href="https://thecap.seework.com/projects/388747/file/22520576/pgmrcm_document_roadmap-20081130.html">https://thecap.seework.com/projects/388747/file/22520576/pgmrcm_document_roadmap-20081130.html</a>  (requires access to IHTSDO Concept Model SIG collaborative space available on request from – <a href="mailto:support@ihtsdo.org">support@ihtsdo.org</a> )
Subsumption (of concepts)	A <i>concept</i> is <i>subsumed</i> by another <i>concept</i> if it logical implies that <i>concept</i> . For example, the concept 'bacterial pneumonia' is subsumed by 'respiratory disease' and by 'infectious disease'.  Subsumption is represented in SNOMED CT by the 'is a' <i>relationship</i> or indirectly by a transitive chain of 'is a' <i>relationships</i> .



**Table 6. Names used to describe SNOMED CT expressions (based on SNOMED CT documents)**

Clinical focus concept	The concept in an <i>expression</i> which represents the nature of a clinical finding or procedure. <i>The clinical focus concept</i> is connected to the context wrapper as the 'associated finding' or 'associated procedure'.
Close-to-user form	A valid <i>expression</i> represented in a form that is directly related to the manner in which the user entered it and/or the way that an application or protocol designer specified it.  This is the form recommended for storage and communication of SNOMED CT expressions.
Context wrapper	The part of an <i>expression</i> which indicates the context in which a clinical focus concept is being used. This includes the 'subject relationship context', 'temporal context' and either the 'finding context' or 'procedure context'.
Expression	A collection of references to one or more <i>concepts</i> used to express an instance of a clinical idea.
Logical transform rules	A set of rules which, when applied to an <i>expression</i> , change the form of the <i>expression</i> without changing its meaning. <i>Logical transform rules</i> specified in SNOMED CT documents allow different representations of information to be transformed to a common normal form in which <i>equivalence</i> and <i>subsumption</i> testing is tractable.
Normal form	An <i>expression</i> resulting from applying a set of SNOMED CT specified transformation rules to another <i>expression</i> .  This is the form required for performing comparisons between alternative ways of representing similar or equivalent meanings.
Post-coordinated expression	An <i>expression</i> including references to two or more <i>concepts</i> used to express an instance of a clinical idea.
Pre-coordinated expression	An <i>expression</i> consisting of a single reference to a <i>concept</i> used to express an instance of a clinical idea.
Refinement	The part of an expression which applies more specific or detailed information to the <i>clinical focus concept</i> .
Equivalence	An <i>expression</i> is <i>equivalent</i> to another <i>expression</i> if, following the application of the specified <i>logical transform rules</i> , the resulting <i>normal form</i> of the two <i>expressions</i> is identical.  For example, the <i>post-coordinated expression</i> 'pneumonia' with 'causative agent' = 'bacteria' is <i>equivalent</i> to the <i>pre-coordinated expression</i> 'bacterial pneumonia'.
Subsumption (of expressions)	An <i>expression</i> is <i>subsumed</i> by another <i>expression</i> if it logically implies that <i>expression</i> .  For example, the <i>post-coordinated expression</i> 'pneumonia' with 'causative agent' = 'pneumococcus' is <i>subsumed</i> by the <i>post-coordinated expression</i> 'pneumonia' with 'causative agent' = 'bacteria'. SNOMED CT specifies rules for testing <i>subsumption</i> by transforming and comparing <i>expressions</i> .

## 3 Integrated Terminology Binding Development

### 3.1 Requirement

Terminology binding is one element of the process of specifying constraints on the way that information is structured and represented.

The underlying rationale for specifying a consistent form of representation for clinical record information is to facilitate reuse. Requirements for reuse are many and varied, ranging from direct support for the care of the individual patient through to aggregate analysis for research, statistics and audit. The common theme of these requirements is the need to retrieve particular items of information reliably and consistently, irrespective of the environment in which the data was entered and stored.

### 3.2 Background

In the past, specific requirements were addressed by specifying datasets and then representing these using structures such as paper-based forms or database tables. The wide range of complex information in clinical records have encouraged the development of coding schemes and terminologies that provide a more flexible way to represent interrelated meanings. It has been clear for many years that the combinations of terminological and structural approaches are essential. More recently it has been recognised that use of a particular coding scheme within a given information model does not guarantee consistency.

In an attempt to address this, designers of information models often specify particular limited coded value sets that can be applied to a particular *field* in a model. The problem with this approach is that it presumes alignment between the semantics of the information model structures and the terminology. Studies to date have shown this alignment is difficult to achieve and cannot be maintained without continuing collaboration in areas of overlap.

### 3.3 Adding terminology to existing information structures

It is possible to design an information model structure for a given purpose and then, as an afterthought, use parts of a separate terminology solution to add the detail required. However, this approach does not result in consistent representation of information. The inherent problem with this approach is that when a *field* in an information model is designated as having a particular meaning it is no longer 'terminology neutral'.

Structural information models inevitably and necessarily represent some aspects of the semantics of a clinical record. A code or terminology expression merely represents an abstract idea. An entry in a clinical record requires an instance of that idea to be bound with an identified subject, a point or period in time and other relevant information. The way this is done affects the ability to process and reuse the information within a clinical record based on its meaning.

Decisions made about the structural representation of particular facets of information, make assumptions about the role and scope of the terminology. These assumptions are usually based on experience of particular coding schemes or a particular view of

the process and use of clinical record. However, problems arise when these implicit assumptions are applied to other terminologies and other perspectives. For example, when SNOMED CT is used in an information model that assumes that the coding scheme represent only the names of diseases or procedures, the added expressivity creates multiple inconsistent options for representation of the same information. Alternative representations present a challenge for effective retrieval and increase the risk of 'false negatives'. Well designed constraints should minimise alternative representation while retaining the required level of expressivity.

In practice, this issue is not confined to highly expressive terminologies. Indeed the drive for greater expressivity arises from similar boundary issues in earlier clinical coding schemes.

For example, when considering the coded representation of diseases, which variants of a disease should be given their own distinct codes and which should be qualified using separate fields? Table 7 illustrates the difficulty of a simple answer to this question by listing three sets of concepts with variations based on facets such as 'causative agent', 'site', 'laterality' and 'severity'.

**Table 7. Illustration of disease specificity**

Pneumonia	Gastro-enteritis	Eczema
Bacterial pneumonia	Salmonella enteritis	Eczema of hands
Pneumococcal pneumonia	Colitis	Eczema of the left hand
Lobar pneumonia	Appendicitis	Excoriated eczema
Left upper lobe pneumonia	Perforated appendix	Infantile eczema
Pneumococcal pneumonia - left upper lobe	Perforation of colon	Severe eczema

SNOMED CT seeks to address this issue by providing a range of expressivity which allows the ideas to be represented at different levels of detail using single codes (pre-coordinated expressions) or compositions of related codes (post-coordinated expressions). The boundary between pre-coordinated and post-coordinated expression is flexible rather than fixed and the potential variants resulting from this flexibility are managed by logical rules for transformation that allow comparison of alternative representations.

#### Examples

'Eczema of left hand' cannot be represented by one SNOMED CT concept. However, a valid post-coordinated expression can be used to refine the 'finding site' of a more general 'eczema' concept so that it specifically applies to the 'left hand'. It can be even more specific – e.g. the 'proximal phalanx of left ring finger'.

In contrast, 'appendicitis' can be expressed using a single concept, rather than requiring the site to be specified separately (e.g. 'inflammation' with 'site' = 'appendix'). The definition of the concept 'appendicitis' specifies its finding site and morphology and thus pre and post-coordinated representations can be determined to be related or equivalent.

### 3.4 Structural approaches to facet representation

An information model that provides separate *fields* for a facet such as 'site' of a disease creates an alternative representation. Unless there is a one-to-one correspondence between these *fields* in the information model and the refinements permitted by the *concept model*, transformation and comparison will not be possible<sup>2</sup>. Even where *fields* do not appear to overlap, aspects of the information model may influence interpretation in ways that interfere with comparison.

One response to concerns about non-alignment is to extend the structural aspects of the information model. For example, in an attempt to align with SNOMED CT, a model might be extended by adding a set of *fields* that replicates all the *concept model attributes*. The idea behind this approach is to retain a structure-based design paradigm while incorporating features from the terminology.

This approach is limited in three respects outlined below:

- a) Pre-coordination and post-coordination boundary cases
  - The flexibility between encapsulation of pre-coordinated expressions (e.g. a disease with a specific site and cause) and composition (e.g. a disease to which site and cause may be added) is limited by splitting facets of an expression across multiple fields.
- b) Overlapping semantics
  - Alignment by extension of an information model on its own does not address issues caused by overlaps between other aspects of the model and the SNOMED CT *concept model*.
  - A key aspect of any integration between an information model and a chosen terminology is to constrain both these components to minimise overlaps. Some overlaps may be inevitable or even beneficial and in these cases, there is a need for explicit rules on interpretation.
- c) Impairment of graceful evolution
  - One of the desiderata for an effective terminology is the ability to 'evolve gracefully' to address error and enhance expressivity<sup>3</sup>. Inevitably, such evolution is impaired if fixed information model structures are based on earlier versions of the terminology model.
  - The SNOMED CT *concept model* has known imperfections that will be addressed over time. Some of these changes may involve addition or removal of particular *concept model attributes*. Such changes can be managed using transformation and other guidance applied to SNOMED CT *expressions*. If these also require addition, removal or changes to *fields* in the information model, this increases the change management burden.

---

<sup>2</sup> The defining relationships of a concept would also overlap with information model fields that duplicate concept model attributes. Therefore, restricting a field to pre-coordinated expressions would not remove this issue of multiple representations.

<sup>3</sup> Cimino J, Desiderata for Clinical Terminology in the Twenty First Century.

The first two points could in theory be addressed by detailed guidance on every boundary case and transformation rules for correct interpretation of all semantic overlaps. However, change management of a growing legacy of guidance and rules is a problem which is unlikely to have a scalable solution. Therefore, this approach cannot be recommended as a basis for the NHS Logical Record Architecture. In contrast, the recommended approach described in the next section is based on rigorous constraint of structure to minimise overlaps.

Application developers may adapt and extend their structural information models to meet LRA requirements provided the practical issues are addressed in ways that provide an aligned common view. However, these requirements are expressed as constraints on a relatively sparse structural model that is designed to be adequately expressive while reducing the number of ways to represent the same information.

### 3.5 Recommended approach to alignment

The recommended approach involves integration of the processes of specifying information model and terminological representations. This requires a reference information model designed to complement the SNOMED CT *concept model*. It also requires the ability to constrain the use of SNOMED CT in ways that take account of the information model and known anomalies and areas of incompleteness in SNOMED CT.

The assumptions made in designing the information model must be explicit so they are open to rational discussion and revision. Explicit statement of these assumptions also provides a foundation for rational and consistent requests for enhancement of SNOMED CT (including *concept model* issues and missing or erroneous content).

The LRA Care Components model, developed as part of the LRA infrastructure, is designed to address these requirements. It is an EN13606 compliant model with derived constrained classes designed for use with SNOMED CT. The LRA Care Components model is also closely related to the HL7 approach to clinical statements and builds on the recommendations of the HL7 Terminology project<sup>4</sup> in relation to use of SNOMED CT in HL7 Version 3.

The LRA Care Components model has three main differences from other EN13606 or HL7 Version 3 derived models.

- Fewer distinct classes/fields
  - Reducing the range of different ways that the coded representation of an item of information can be split across different information model artefacts.
- Explicit alignment between derived classes in the Care Components model and the SNOMED CT *concept model domains* that are most frequently used to express clinical information<sup>5</sup>:

---

<sup>4</sup> See the HL7 Draft Standard for Trial Use 'Guide to the Use of SNOMED CT in HL7 Version 3'.

<sup>5</sup> SNOMED CT *concept model domains* most commonly used to express clinical information include 'clinical findings', 'procedures', and 'situations with explicit context'.

- The Care Components model recognises the distinction between activities (i.e. procedure situations) and observations (i.e. finding situations). In contrast, the HL7 RIM treats all clinical statements as types of acts and requires the Observation.code attribute to express the nature of the action of observation.
- Exclusion of structural codes and specialisations that conflict with or duplicate SNOMED CT semantics.

The result of these refinements is a reduction in the range of ways in which the same information can be represented by overlaps between the information model and terminology.

The LRA Care Components model is a foundation for the development of consistent representation of clinical information. It forms the basis for constrained information model patterns which need to be represented in a consistent manner. The proposed LRA approach to serialised representation of these constrained patterns is to use XMI (XML Model Interchange format). An important aspect of these constraints is the need to incorporate constraints on SNOMED CT *expressions* used in particular fields. This requires a serialised representation of *expression constraints* and a way to bind these constraints to relevant points in the information model serialisation. Section 4 discusses this requirement and Section 5 specifies a recommended approach to representation of *expression constraints*.

An appropriate mechanism for representation of *expression constraints* within the information model is one part of the requirement for alignment. In addition, the process of developing and documenting these constraints need to be aligned and combined. This requires tools that allow terminology and information model components to be designed, reviewed, refined and extended in a rational integrated way. Requirements for terminology tooling (e.g. to support development of *expression constraints*) and the integration of these with general design and documentation tools are addressed in Section 6.

## 4 Terminology Binding Representation - Requirements

### 4.1 Overview

This section of the document discusses the requirements for effective representation of terminology bindings to support the effective application of the Logical Record Architecture.

A terminology binding is an instance of a link between a *terminology component* and an *information model artefact*. Therefore, the document considers the representation of the required *terminology components* and the way these are associated with relevant *information model artefacts*. The *terminology components* to which this document applies are SNOMED CT expressions and representations of constraints on SNOMED CT expressions.

The *information model artefact* to which a *terminology binding* is applied may be a field of a class in a static model or a collection of *fields* of one or more related classes. The *information model artefacts* to which this document applies are described by the *LRA Technical Model Infrastructure Specification*. In particular, the Care Components part of that model covers the aspects health records in which SNOMED CT expressions are bound to represent key aspects of processable meaning.

A consistent mechanism is required to represent constraints on the use of SNOMED CT expressions in particular classes in the information model. Without such a mechanism there are multiple ways to represent the same item of information. These alternative representations cannot be satisfactorily retrieved or compared; and some are frankly ambiguous.

The Care Components model itself specifies some general terminology bindings applicable to broad types of clinical information. These bindings form an important foundation for consistent modelling. However, on their own they are insufficient and require refinement to cover more specific requirements.

The classes specified by the Care Components model are building blocks from which patterns will be derived to meet particular use cases. Each terminology bound class used within a pattern needs to be associated with refined terminology bindings so it is used in a way intended.

The representation of terminology bindings is an integral part of the representation of constrained information models. In many ways a terminology expression constraint is like any other constraint that applies to the value of a field within a constrained model. However, the way in which such constraints are represented and evaluated must take account of the relationships with an external resource (the SNOMED CT release files) and the description logic rules that apply to this resource (the SNOMED CT concept model).

This document specifies an expression constraint representation that is designed to meet this requirement. These constraints may be included, either verbatim or by reference, in a machine processable representation of the constrained information model.

## 4.2 Requirements for different types of terminology binding

### 4.2.1 Terminology bindings that apply to individual expressions

Earlier work identified requirements for several types of terminology binding<sup>6</sup>. Two of these, *semantic expression constraints* and *literal expression constraints*, relate directly of the application of constraints to a single terminology expression. The key features of these types of constraint are summarised in Table 8. More details and examples of these constraints are provided in Appendix A .

**Table 8. Bindings that apply to a field that supports use of a SNOMED CT expression<sup>7</sup>**

Binding type	SNOMED CT component
<b>Semantic expression constraint</b> <i>A constraint on the possible meanings that can be expressed by a field.</i>	An <i>expression constraint</i> that <ul style="list-style-type: none"> <li>Restricts permitted expression to subtypes of a specified concept or set of concepts.</li> <li>May also constrain facets of the meaning of an expression represented by particular defining relationships or post-coordinated refinements.</li> </ul> The test for conformance with a <i>semantic constraint</i> is: Does the result of a SNOMED CT normal form transform applied to the <i>field</i> value conform to the <i>expression constraint</i> ?
<b>Literal expression constraint</b> <i>Constrains the ways in which an expression may be constructed to represent a given meaning.</i>	An <i>expression constraint</i> that <ul style="list-style-type: none"> <li>Requires, prohibits or restricts the use an attribute or the value range that can be applied to an attribute.</li> <li>May prohibit any or all types of post-coordination.</li> <li>May require a particular aspect of the information to be post-coordinated.</li> </ul> The test for conformance with an <i>expression-structure constraint</i> is: Does the literal (untransformed) <i>field</i> value conform to the <i>expression constraint</i> ?

<sup>6</sup> The different types of terminology binding are identified in section 7 of 'Terminology Binding Requirements and Principles'. The type of constraint previously referred to as an 'expression structure constraint' has been renamed as 'literal expression constraint' based on feedback on early terminology binding documents.

<sup>7</sup> Support for inclusion of a SNOMED CT expression within a *field* is typically determined by its data type and any higher level terminology bindings. A simple coded data type meets this requirement for pre-coordinated expressions. A datatype that can contain an expression represented using compositional grammar or an alternative syntax with similar capabilities is required to meet the range of structural forms that are required or permitted by the constraints.



#### 4.2.2 Terminology bindings that apply to multiple fields or classes

The document 'Terminology Binding Requirements and Principles' described two types of terminology binding that apply to multiple fields or classes (see Table 9).

*Constructor bindings* specify how values from multiple fields are assembled to create a semantically complete SNOMED CT expression. *Composite semantic constraints* specify constraints on the results of applying a *constructor binding*.

The requirement for and representation of *constructor bindings* depends on the nature of the information model and the means of identifying and referencing artefacts within the model. This document does not address this requirement in detail because the requirement and referencing method remain to be determined as the information model evolves.

A *composite semantic constraint* can be represented in the same way as a *semantic expression constraint*. The only difference is that it applies to the result of applying specified *constructor bindings*, rather than to a single expression.

**Table 9. Bindings that apply to a set of *fields* within one or more classes<sup>8</sup>**

Binding type	SNOMED CT component
<p><b>Constructor binding</b></p> <p><i>A binding that indicates how values entered in two or more related fields are combined to represent a composite meaning in a consistent form.</i></p> <p><i>(For more details see 'Terminology Binding Requirements and Principles' - section 7.4)</i></p>	<p>An expression including references to specific information model <i>fields</i> that indicate the way in which values from those points are combined and/or transformed to align with a common model of meaning.</p> <ul style="list-style-type: none"> <li>• A specific feature of this type of constraint is the need for the binding to refer to items in the information model. Most other bindings can be referenced from the information model, with no need for references back to specific <i>information model artefacts</i>.</li> <li>• Depending on the nature of the source data and target model, a <i>constructor binding</i> may support transformations in either direction. <ul style="list-style-type: none"> <li>○ Transform multiple information model <i>fields</i> into a single <i>post-coordinated expression</i>. This may include deriving context model values from <i>information model artefacts</i>.</li> <li>○ Transform a <i>post-coordinated expression</i> into a set of <i>information model artefacts</i>. This may include using context model values to determine the appropriate <i>information model artefacts</i> to be used to represent the information.</li> </ul> </li> </ul> <p>A constructor binding is not itself a constraint but the result of applying a constructor binding may be subject to an associated <i>composite semantic constraint</i>.</p>
<p><b>Composite semantic constraint</b></p> <p><i>A constraint on the meanings that can be expressed by a combination of two or more related fields.</i></p>	<p>The test for conformance with a <i>composite semantic constraint</i> is:</p> <p>Does the result of a SNOMED CT normal form transformation applied to the <i>expression</i> derived by applying a specified <i>constructor binding</i> conform to the <i>expression constraint</i>?</p>

<sup>8</sup> Individual *fields* in the set must support inclusion of SNOMED CT expressions and these will usually be subject to other terminology bindings.

### 4.2.3 Terminology bindings that relate to input and retrieval

The document 'Terminology Binding Requirements and Principles' described some types of terminology binding that apply specifically to data entry and retrieval (see Table 10).

**Table 10. Bindings that apply to entry and retrieval of SNOMED CT expressions**

Binding type	SNOMED CT component
<b>Selection support binding</b> <i>A binding that represents a range of commonly used values in a way that facilitates selection at a user interface (in clinical use or during design of more specific models or data entry protocols)</i>	<p>A subset, reference set or an <i>expression constraint</i> referring to one or more sets</p> <ul style="list-style-type: none"> <li>• May prioritise and structure the presentation of options rather than strictly constraining permitted content.</li> <li>• Depending on user-interface implementations, these may <ul style="list-style-type: none"> <li>○ populate picking-lists and check-lists</li> <li>○ constrain or order searches</li> <li>○ provide hierarchies for navigating to relevant values</li> <li>○ enable efficient access to post-coordination refinements</li> </ul> </li> </ul> <p>A <i>selection support binding</i> is <u>not</u> a constraint on logical record content. It represents the possible or common types of entry identified by clinical users. This is relevant to representing common patterns of use to a general logical model, without applying constraints that limit future use of the same model in fields in which additional logically consistent values may be required.</p>
<b>Retrieval binding</b> <i>A binding that indicates how data from one information model artefact may be used to populate another artefact. For example, to populate a check list entry from a relevant but more detailed clinical entry.</i>	<b>Retrieval binding</b> <i>A binding that indicates how data from one information model artefact may be used to populate another artefact. For example, to populate a check list entry from a relevant but more detailed clinical entry.</i>

*Selection support bindings* relate specifically to the way the user interface behaves to assist data entry. The main differences between a *selection support binding* and an *expression constraint* are as follows:

- An *expression constraint* permits, requires or prohibits particular values, while a *selection support binding* may simplify entry of particular values without necessarily prohibiting other values.
- An *expression constraint* is primarily concerned with the meaning of the expression, while a *selection support binding* may also specify which terms (e.g. preferred terms or particular synonyms) should be displayed or recorded.
- An *expression constraint* can be bound to an implementation independent information model such as the Care Components model. In contrast, a *selection support binding* may be bound to a data entry protocol, a screen form or some other application-dependent user interface component.

*Selection support bindings* can be represented in the same way as *expression constraints*. The differences noted above can be accommodated by distinguishing

between mandatory constraints and constraints that are specific to a particular data entry context.

*Retrieval bindings* as described in Table 10 relates to the way in which the data is retrieved for display in the user interface. This is just one of many requirements for meaning based retrieval.

- Selective retrieval requirements require selection criteria (predicates) to be expressed for a variety of fields in the information model. Some of these fields will be populated with SNOMED CT expressions and thus there is a requirement for a consistent representation of SNOMED CT expression predicates. The principal reason for using SNOMED CT is to support retrieval based on meaning and it follows that these requirements for expression predicates are logically identical to those for *semantic expression constraints*.

The document 'Terminology Binding Requirements and Principles' also mentions *node-fixed binding* and *choice-fixed binding*. Further analysis indicates that these are special cases of other types of terminology binding and do not need to be considered further in their own right.

*Node-fixed binding* referred to cases where the presence of a particular information model artefact implied a meaning that could be expressed using a single fixed SNOMED CT expression. This type of binding can be represented as a *literal expression constraint*, which only permits a single fixed value.

*Choice-fixed binding* referred to bindings between individual values in a set of choices and a particular SNOMED CT expression. Integration of terminology binding with the information model design process should enable lists of choices to be represented as *selection support bindings*. However, if this is not possible, a *choice-fixed binding* can be expressed by binding a *literal expression constraint* to each relevant value.

## 5 Representing expression constraints

### 5.1 Applicability of expression constraints

Most of the types of terminology binding discussed in 4.2 require a mechanism that can represent constraints on SNOMED CT expressions. This section specifies a mechanism that is intended to meet these requirements.

*Literal expression constraints* and *semantic expression constraints* can be represented in exactly the same way. The only difference is the way in which these constraints are tested. In the case of *semantic expression constraints*, the tests are applied to normal forms of the candidate expressions. In contrast, *literal expression constraints* are applied directly to the literal form of the candidate expression.

*Retrieval bindings* and other expression predicates in queries are interpreted in the same way as *semantic expression constraints*. The difference is that rather than validating that an expression conforms to a specification, the result determines whether the containing record entry should be included in the results of the query.

*Selection support bindings* may be used to simplify entry of particular values in addition to formally constraining what can be entered. Similar representational forms can meet these requirements, provided that the intended interpretation in the context of use is clear.

### 5.2 Expression constraint characteristics

#### 5.2.1 Extensional and intensional constraints

An *expression constraint* can be represented either:

- '*Extensionally*' by listing all the permitted expressions; or
- '*Intensionally*' by specifying rules that can be applied to determine whether an expression is permitted.

*Extensional* representation is usually appropriate where there is a known limited set of specifically permitted values.

*Intensional* representation provides a more powerful and flexible approach where there are large sets of permitted values that share a characteristic (e.g. being subsumed by a particular concept).

The mechanism for representing constraint expressions must support intensional specification of constraints, at least to enable subtypes of a specified concept to be included or excluded.

#### 5.2.2 Pre-coordinated expression constraints

A pre-coordinated *expression constraint* is a set of permitted concept identifiers.

Such a set can be represented *extensionally* as a list of identifiers or more formally by reference to a predefined SNOMED CT Subset or Refset. However, to limit the maintenance burden posed by large lists of concept identifiers, an *intensionally* approach should also be supported.

As a minimum, the *intensional* approach to representation of sets of concepts should support:

- Inclusions or exclusions of concepts based on subtype relationships (i.e. all subtype descendants of a specified concept).
- Combinations of sets in ways that support general set operators (i.e. union, intersection, complement and disjunction).

Pre-coordinated *expression constraints* can represent simple *semantic expression constraints*. These constraints can be applied even if the candidate expressions may be post-coordinated.

- For example, specifying a single concept and all its subtypes implicitly permits any valid post-coordinated refinements of those concepts<sup>9</sup>.

### 5.2.3 Post-coordinated expression constraints

A post-coordinated *expression constraint* explicitly specifies the permitted attributes and values that can be used as refinements in an expression.

Post-coordination is required to represent *literal expression constraints*. It is also required to represent *semantic expression constraints* that constrain specified facets of meaning specified by defining relationships or refinements.

An *extensional* representation of post-coordinated *expression constraints* would need to enumerate all permitted expressions. Each element in an expression presents an opportunity for combinatorial explosion of the range of possible expressions. Therefore, an extensional approach is unlikely to be a practical way to address requirements for post-coordinated *expression constraints*.

*Intensional* representation of *expression constraints* requires an approach that has the following features:

- Applies individual constraints to the relevant parts of the logical model of a SNOMED CT expression (see Figure 1).
- Allows constraints to be applied to:
  - Focus concept values
  - Refining attribute names, values and groups
- Permits the values in these constraints to be represented
  - either *extensionally* or *intentionally*;
  - as a pre-coordinated or nested post-coordinated expressions
- Allows sets of constraints to be combined using set operators.

---

<sup>9</sup> A valid refinement always results in a meaning that is subsumed by the refined focus concept. Therefore, a post-coordinated expression is a subtype of its focus concept. In addition, any expression that has a normal form subsumed by the normal form of the specified concept also passes such a constraint.

## 5.3 Abstract model of expression constraints

### 5.3.1 Description

The general model for *expression constraints* is a set of one or more expressions in which the individual concept identifiers are replaced by a set of one or more permitted concepts.

These sets of permitted concepts can be represented either extensionally or intensionally using the approaches described in 5.2.2 and 5.2.3.

The classes in the abstract model of SNOMED CT expressions are matched by classes in the abstract model of constraints Figure 2.

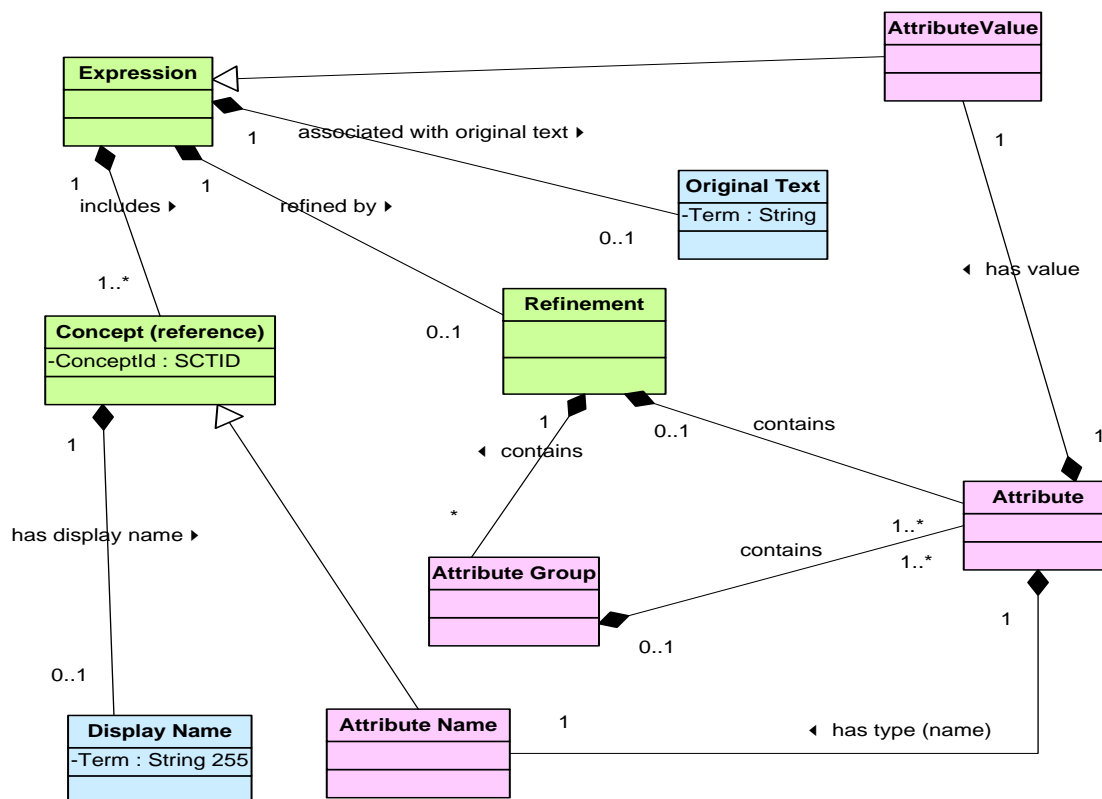


Figure 1. SNOMED CT Expressions – logical model<sup>10</sup>

<sup>10</sup> Diagram from 'SNOMED CT Abstract Models and Representational Forms' published by IHTSDO.

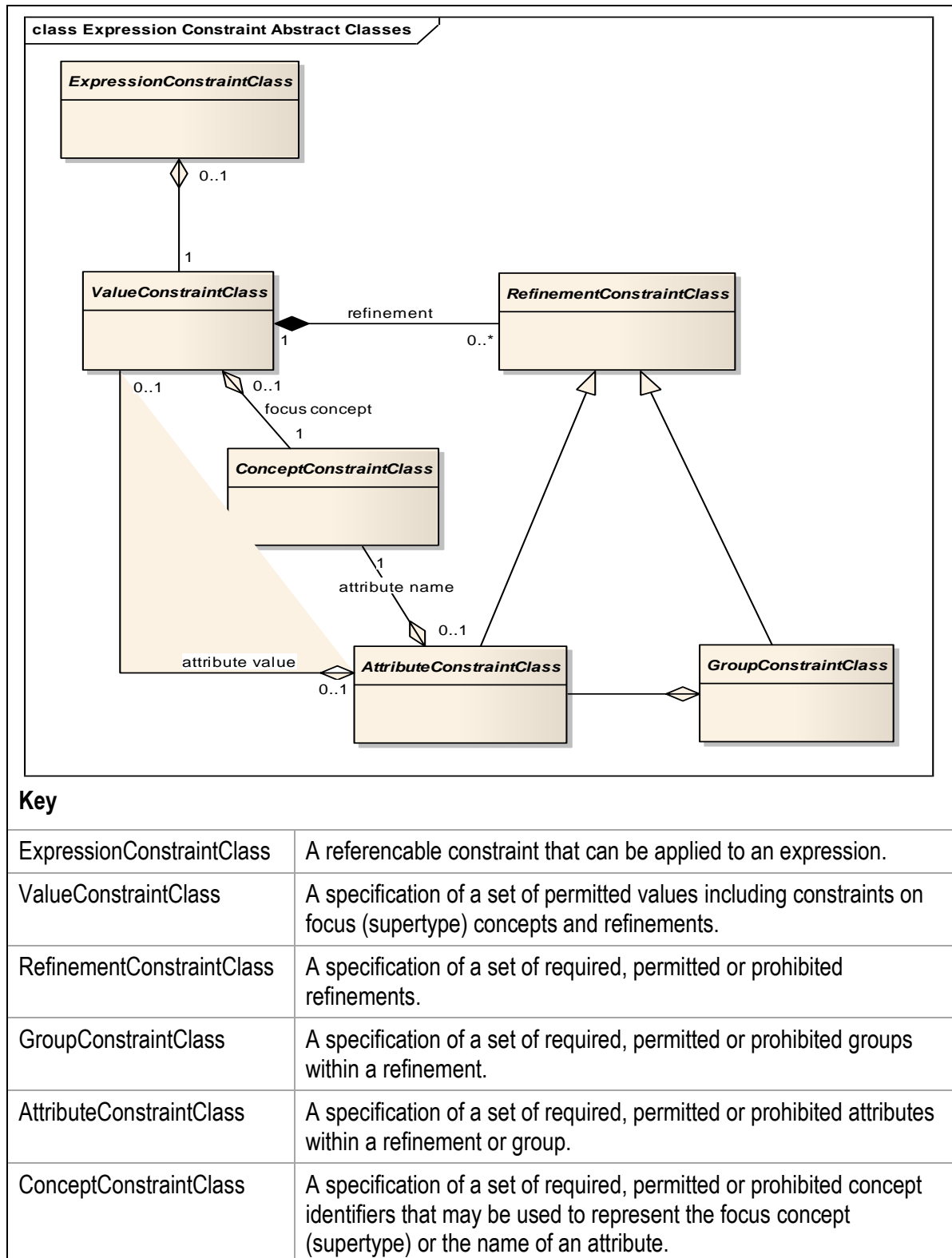


Figure 2. Abstract class model of expression constraints

### **5.3.2 Required extensions of the abstract model of constraints**

To support the requirements identified in previous sections the classes shown in the abstract model of constraints outlined in Figure 2 must be specialised with extensions that address the points in the following sections:

#### **5.3.2.1 Requirements for representing sets of constraints**

Each constraint class must be capable of representing

- A single permitted, required or prohibited value or pattern
- Sets of permitted, required or prohibited values or patterns
- Logical operators that allows sets to be combined to derive union, intersection and complement sets.

#### **5.3.2.2 Requirements for representing concept constraints**

Concept constraints must be specifiable either extensionally or intentionally. The options for representing a concept constraint should therefore include the following:

- Directly by reference to a particular conceptId.
- Indirectly by reference to a predefined Refset.
- Intensionally by specifying the inclusion or exclusion of a referenced concepts and/or subtype descendants of that concept.

#### **5.3.2.3 Requirements for representing constraints to refinement cardinality**

The SNOMED CT expression model permits refinements to include:

- multiple ungrouped attributes;
- multiple groups;
- multiple attributes within each group.

The representation of refinement constraints needs to address and where appropriate constrain the various options for multiplicity arising from this structure.

- Attribute constraints must indicate the minimum and maximum number of permitted occurrences within a refinement or within a group.
- Group constraints must indicate the minimum and maximum number of occurrences within the refinement.

Minimum and maximum occurrences can be used to specify:

- A prohibited attribute or group (maximum occurrences =0)
- An optional attribute or group (minimum occurrences =0)
- A required attribute or group (minimum occurrences =1)
- Limits to the number of instances of an attribute or group that conforms to a specified constraint.



#### 5.3.2.4 Other aspects of refinement constraint interpretation

Interpretation of sets of refinement constraints requires some additional factors to be specified.

- Does the absence of a particular attribute or group constraint indicate:
  - Other attributes or groups are prohibited; or
  - Other attributes or groups are not constrained<sup>11</sup>.
- Does the presence of a particular attribute with a specified value indicate:
  - No other unspecified uses of that attribute are permitted in the same refinement; or
  - Provided the constraint is satisfied, other instances of the same attribute are also permitted in the refinement.
- Does an attribute constraint that is not part of a group constraint indicate:
  - The constraint only applies to ungrouped attributes; or
  - The constraint applies to any occurrences of this attribute whether grouped or ungrouped.

Rational answers to these questions depend on the type of *expression constraint*.

- *Semantic expression constraints* can usually be interpreted permissively. They are intended to limit the range of meanings. Additional groups or attributes would result in a meaning that is subsumed by the constrained meaning. Therefore, unless explicitly prohibited, these additions would not still result in valid expressions.
- *Literal expression constraints* must be interpreted more strictly. They are intended to force or prevent particular aspects of post-coordination. Permitting other attributes or groups by default would undermine this type of use.

Dependency on the type of *expression constraint* makes it tempting to specify a default value that is interpreted differently according to the constraint type. However, while this is a possible option, it seems preferable to be explicit to minimise the risk of misinterpretation.

The additional attributes listed in Table 11 are recommended for inclusion in the relevant classes to explicitly indicate the intended interpretation. No implicit defaults are specified, but the table includes recommendation of initial values based on the intended use of the *expression constraint*.

---

<sup>11</sup> Even if attributes and groups that are not directly constrained they should be constrained by a higher level constraint such as the SNOMED CT concept model.

**Table 11. Additional attributes to support refinement constraint interpretation**

Class	Attribute	Values and usage
ValueConstraintClass and GroupConstraintClass	refinementMode	<p><b>open</b> - Refinements are permitted unless they conflict with one of the contained constraints.</p> <ul style="list-style-type: none"> <li>This should be the initial value for a <i>semantic expression constraint</i> or a constraint intended for use in a retrieval specification.</li> </ul> <p><b>closed</b> - Refinements that do not satisfy at least one contained constraint are prohibited.</p> <ul style="list-style-type: none"> <li>This should be the initial value for a <i>literal expression constraint</i> or a constraint intended for use as part of a selection support binding.</li> </ul> <p><u>Note:</u> Open refinement only applies to attributes that are not specified. If an attribute is specified with a particular value then other instances of that attribute are not permitted in the same refinement unless they also conform to one of the constraints.</p>
AttributeConstraint	includeGrouped	<p><b>true</b> - Attributes in groups contained in the same refinement also match this ungrouped AttributeConstraint.</p> <ul style="list-style-type: none"> <li>This should be the initial value for a <i>semantic expression constraint</i> or a constraint intended for use in a retrieval specification.</li> </ul> <p><b>false</b> - Attributes in groups do not match this ungrouped AttributeConstraint.</p> <ul style="list-style-type: none"> <li>This should be the initial value for a <i>literal expression constraint</i> or a constraint intended for use as part of a selection support binding.</li> </ul> <p><u>Note:</u> Only relevant to attributes that are not in a group and is ignored if specified in a grouped AttributeConstraint.</p>

### 5.3.3 Recommended persistent representation of constraints

The abstract model of constraints (see Figure 2) and the required extensions to this model (see 5.3.2) were used to evaluate the applicability of alternative approaches for representing constraints (see Appendix E ). None of the pre-existing approaches fully meet the requirements. Therefore, an XML representation based directly on the extended abstract model was developed.

This representation meets the requirements addressed by the Extended SNOMED Composition Grammar<sup>12</sup> but adds greater expressivity including:

- Cardinality constraints
- Clearer representation of set operations.
- Explicit support for alternative types of *expression constraints*.
- Inclusion of constraint identifiers, metadata and annotations.

The use of XML also offers several additional advantages.

- Use of XML schema to enable syntactic validation of constraint representation<sup>13</sup>.
- Simplification of constraint parsing (and caching) when testing of candidate expressions.
- Ability to use XSLT (or XQUERY) to transform relevant information to and/or from other XML representations including OWL, MRCM and DITA.
- Ability to use XSLT (or XQUERY) to transform constraints to human-readable renderings (including the Extended SNOMED Composition Grammar).

---

<sup>12</sup> Alignment between the XML representation constraints and Extended SNOMED Composition Grammar (ESCG) has been demonstrated as follows. A parser has been used to convert existing constraints expressed in ESCG into the XML form and an XSLT transform has converted this back in plain text ESCG and into a display marked up ESCG. Where the starting representation is ESCG the round trip validation has been demonstrated without any loss of information. Since ESCG is less expressive, transforms from XML representations that make use of additional features into ESCG inevitably result in some loss of information. Therefore, in these cases, the ESCG provides an incomplete human-readable view of the constraint and in future these transforms should be augmented with additional textual annotations.

<sup>13</sup> The value of this type of validation was demonstrated by the identification of several errors when converting pre-existing constraints from the extended compositional grammar representation.

## 5.4 Expression constraint – persistent representation

### 5.4.1 Overview

Expression constraints are represented as uniquely identified XML documents that can be persisted as files or entries in a repository. The model underlying this representation, shown in Figure 3, specialises and extends the general model shown in Figure 2 to meet the requirements identified in 5.3.2.

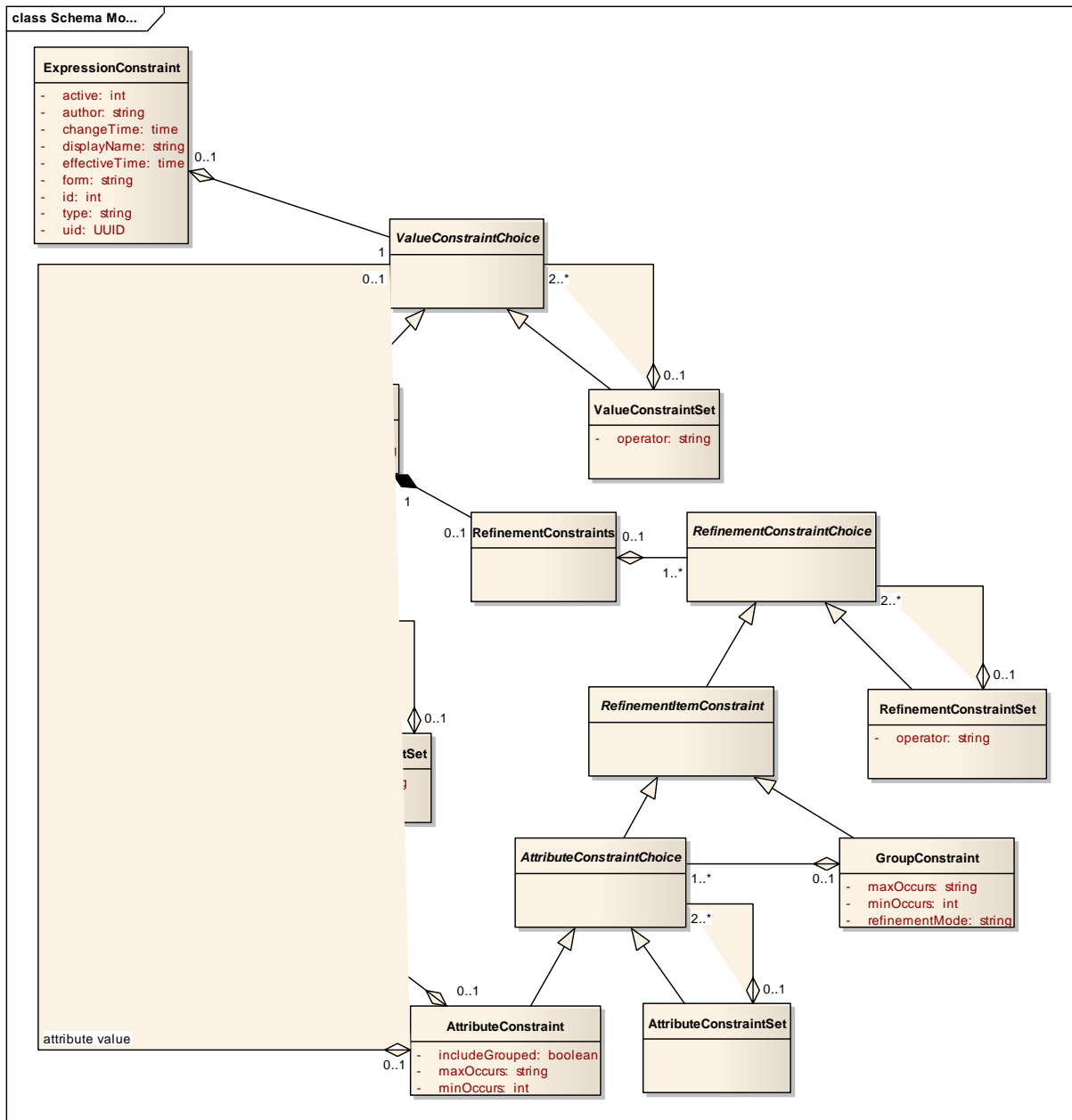


Figure 3. Expression constraint schema model

Figure 4 shows the outline of the XML Schema (sct\_constraint.xsd) which formally documents this representation.

The schema is supported by detailed technical documentation in two forms

- A navigable HTML form (sct\_constraint.html)
- A static word document  
(LRA\_SNOMED-CT\_Terminology\_Binding\_Technical\_Specification\_Schema.doc).

The schema is also supported by a folder containing sample files base on this schema and a set of XSLT transforms that can be applied to the constraints to render human readable representation in HTML and DITA.

Section 5.4.2 summarises the schema and describes the rules applied to determine whether an expression conforms to an *expression constraint*.

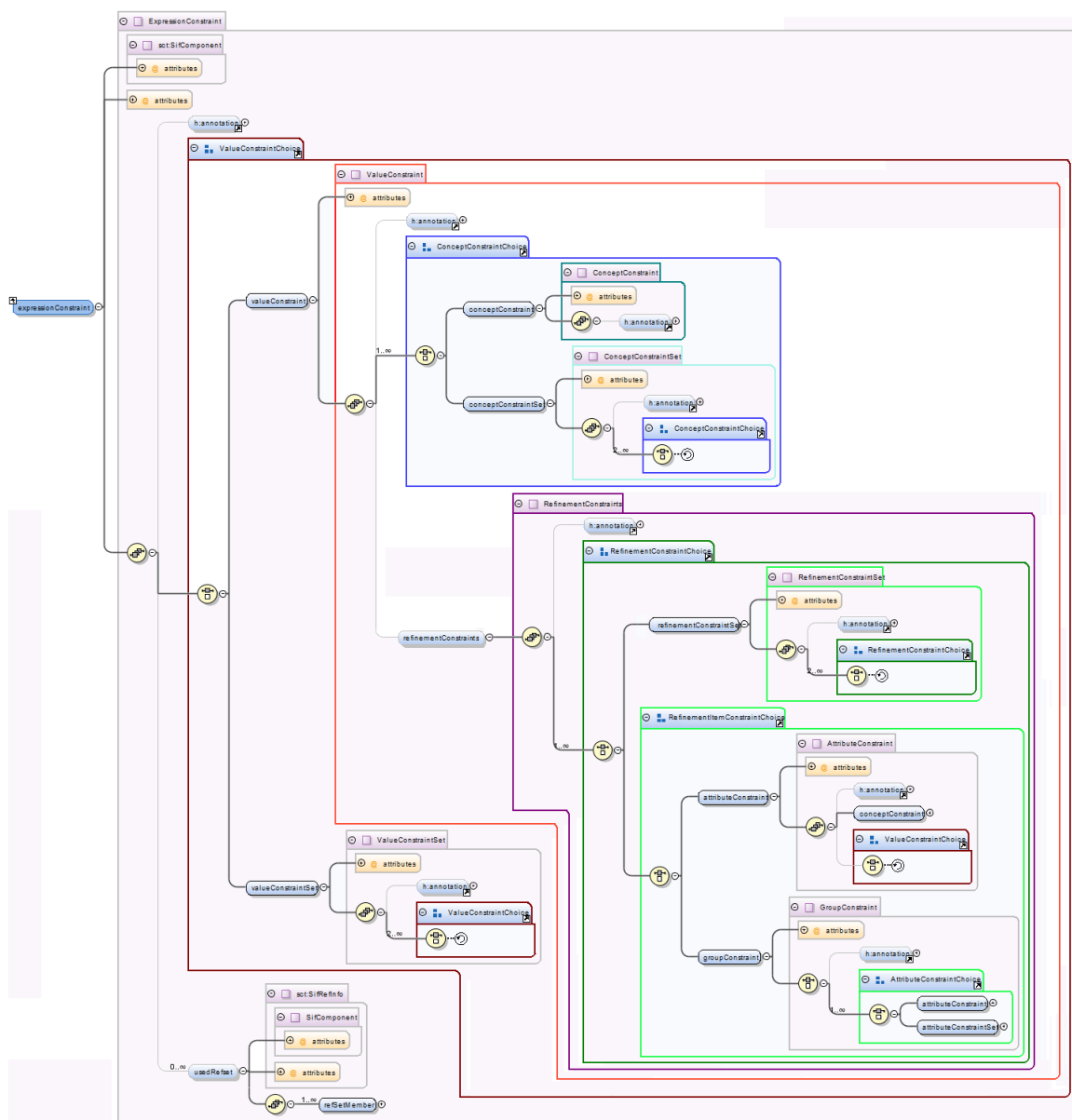


Figure 4. XML Schema overview based on the extended general model

### 5.4.2 Summary of Expression Constraint Schema

Element		Description
<b>ExpressionConstraint</b>		<p>A referencable constraint that can be applied to an expression.</p> <p>Attributes</p> <ul style="list-style-type: none"> <li>• <b>uid</b> - a unique identifier of the constraint.</li> <li>• <b>displayName</b> – an optional name for the constraint.</li> <li>• <b>type</b> – (semantic literal) indicates the type of intended interpretation.</li> <li>• <b>form</b> – (long intermediate short) indicates whether the constraint is: <ul style="list-style-type: none"> <li>○ 'full' – states all applicable constraints,</li> <li>○ 'intermediate' – does not state general concept model constraints</li> <li>○ 'short' – only states restrictions applies to another constraint</li> </ul> </li> <li>• id, active, effectiveTime, changeTime, author (see 5.4.7)</li> </ul> <p>Contained elements</p> <ul style="list-style-type: none"> <li>• [1] <i>ValueConstraintChoice</i></li> </ul>
<b>ValueConstraintChoice</b>		A specification of a set of permitted values including constraints on focus (supertype) concepts and refinements.
EITHER	<b>ValueConstraintSet</b>	A set of type <i>ValueConstraintChoice</i> .
		The constraint is satisfied if the candidate expression value conforms to one member of the resulting set of constraints.
		<p>Attributes</p> <ul style="list-style-type: none"> <li>• <b>operator</b> – (OR AND) <ul style="list-style-type: none"> <li>○ 'OR' – the set is the union of the results of all the contained <i>ValueConstraintChoices</i></li> <li>○ 'AND' – the set is the intersection of the results of all the contained <i>ValueConstraintChoices</i></li> </ul> </li> </ul> <p>Contained elements</p> <ul style="list-style-type: none"> <li>• [2..*] <i>ValueConstraintChoice</i></li> </ul>
		Contained elements
	<b>ValueConstraint</b>	A value or set of values that an expression may have.
		The constraint is satisfied if the candidate expression has a value that matches (or is within the set of values specified by) the contained <i>ConceptConstraintChoice</i> and <i>RefinementConstraints</i> (if present). If the <i>refinementMode</i> is 'open', other non-conflicting refinements are also permitted.
		<p>Attributes</p> <ul style="list-style-type: none"> <li>• <b>refinementMode</b> (open closed) <ul style="list-style-type: none"> <li>○ 'closed' - all attributes and groups in the refinement must conform to at least one specified <i>RefinementConstraintChoice</i>.</li> <li>○ 'open' – additional attributes or groups that do not conflict with a <i>RefinementConstraintChoice</i> are permitted.</li> </ul> </li> </ul> <p>Contained elements</p> <ul style="list-style-type: none"> <li>• [1] <i>ConceptConstraintChoice</i> – constrains the concept under which the candidate expression must be subsumed</li> <li>• [0..1] <i>RefinementConstraints</i> – constraints refinements in the candidate expression.</li> </ul>
		Contained elements

Element		Description
<b>RefinementConstraints</b>		Represents the permitted or required refinements that apply as part of the containing <i>ValueConstraint</i> .
		The constraint is satisfied if the candidate expression has groups and attributes that satisfy the conditions specified by the contained <i>RefinementConstraintChoice</i> (s).
		Contained element <ul style="list-style-type: none"> <li>• [1..*] <i>RefinementConstraintChoice</i></li> </ul>
<b>RefinementConstraintChoice</b>		A specification of a set of required, permitted or prohibited refinements.
ONE OF	<b>RefinementConstraintSet</b>	A set of type <i>RefinementConstraintChoice</i> .
		The constraint is satisfied if the candidate expression refinement conforms to one member of the resulting set of constraints.
		Attributes <ul style="list-style-type: none"> <li>• <b>operator</b> – (OR AND) <ul style="list-style-type: none"> <li>○ ‘OR’ – the set is the union of the results of all the contained <i>RefinementConstraintChoices</i></li> <li>○ ‘AND’ – the set is the intersection of the results of all the contained <i>RefinementConstraintChoices</i></li> </ul> </li> </ul>
		Contained elements <ul style="list-style-type: none"> <li>• [2..*] <i>RefinementConstraintChoice</i></li> </ul>
	<b>GroupConstraint</b>	Represents a constraint on groups within a refinement.
		The constraint is satisfied if the refinement contains <b><i>n</i></b> groups that conform to the set of contained <i>AttributeConstraintChoices</i> ; where the value <b><i>n</i></b> is in a range specified by the attribute minOccurs and maxOccurs.
		Attributes <ul style="list-style-type: none"> <li>• <b>minOccurs</b> (default=1)</li> <li>• <b>maxOccurs</b> (default=1)</li> <li>• <b>refinementMode</b> (open closed) <ul style="list-style-type: none"> <li>○ ‘closed’ - all attributes in the candidate group must conform to at least one specified <i>AttributeConstraintChoice</i>.</li> <li>○ ‘open’ – additional attributes that do not conflict with an <i>AttributeConstraintChoice</i> are permitted.</li> </ul> </li> </ul>
		Contained element <ul style="list-style-type: none"> <li>• [1..*] <i>AttributeConstraintChoice</i></li> </ul>
	<b>AttributeConstraint</b>	Represents a permitted attribute in a refinement or group. It may also specify the permitted values of that attribute and the number of permitted instances of the attribute.
		See details under <i>AttributeConstraintChoice</i> (page 32)

Element		Description
<b>AttributeConstraintChoice</b>		<i>A specification of a set of required, permitted or prohibited attributes within a refinement or group.</i>
EITHER	<b>AttributeConstraintSet</b>	Represents a permitted attribute in a refinement or group. It may also specify the permitted values of that attribute and the number of permitted instances of the attribute.
		The constraint is satisfied if the refinement (or group) contains <i>n</i> attributes that have a name specified by the contained <i>ConceptConstraint</i> and a value that satisfies the contained <i>ValueConstraintChoice</i> (if present); where the value <i>n</i> is in a range specified by the attributes minOccurs and maxOccurs.
		Attributes <ul style="list-style-type: none"> <li>• <b>minOccurs</b> (default=1)</li> <li>• <b>maxOccurs</b> (default=1)</li> <li>• <b>includeGrouped</b> (true false) <ul style="list-style-type: none"> <li>○ 'true' – Attributes in groups contained in the same refinement also match this ungrouped <i>AttributeConstraint</i>.</li> <li>○ 'false' – Attributes in groups do not match this ungrouped <i>AttributeConstraint</i>.</li> </ul> </li> </ul>
		Contained elements <ul style="list-style-type: none"> <li>• [1] <i>ConceptConstraint</i> (specifies the name of the attribute)</li> <li>• [0..1] <i>ValueConstraintChoice</i> (if present constrains the permitted values of the specified attribute)</li> </ul>
	<b>AttributeConstraint</b>	Represents a permitted attribute in a refinement or group. It may also specify the permitted values of that attribute and the number of permitted instances of the attribute.
		The constraint is satisfied if the refinement (or group) contains <i>n</i> attributes that have a name specified by the contained <i>ConceptConstraint</i> and a value that satisfies the contained <i>ValueConstraintChoice</i> (if present); where the value <i>n</i> is in a range specified by the attributes minOccurs and maxOccurs.
		Attributes <ul style="list-style-type: none"> <li>• <b>minOccurs</b> (default=1)</li> <li>• <b>maxOccurs</b> (default=1)</li> </ul>
		Contained elements <ul style="list-style-type: none"> <li>• [1] <i>ConceptConstraint</i> (specifies the name of the attribute)</li> <li>• [0..1] <i>ValueConstraintChoice</i> (if present constrains the permitted values of the specified attribute)</li> </ul>



Element		Description
<b>ConceptConstraintChoice</b>		A specification of a set of required, permitted or prohibited concept identifiers that may be used to represent the focus concept (supertype) or the name of an attribute.
EITHER	<b>ConceptConstraintSet</b>	A set of type <i>ConceptConstraintChoice</i> . The members of the sets are combined by applying either the operator 'OR' (union) or 'AND' (intersection).
		The constraint is satisfied if the relevant concept in a candidate expression is in the resulting set.
		Attributes <ul style="list-style-type: none"> <li>• <b>operator</b> – (OR AND) <ul style="list-style-type: none"> <li>○ 'OR' – the set is the union of the results of all the contained <i>ConceptConstraintChoices</i></li> <li>○ 'AND' – the set is the intersection of the results of all the contained <i>ConceptConstraintChoices</i></li> </ul> </li> </ul>
		Contained elements <ul style="list-style-type: none"> <li>• [2..*] <i>ConceptConstraintChoice</i></li> </ul>
	<b>ConceptConstraint</b>	A <i>ConceptConstraint</i> specifies a set of permitted concept identifiers relevant to a specified location in an expression.
		The constraint is satisfied if the relevant concept falls within the set specified by the <i>id</i> , <i>rule</i> and <i>exclude</i> attributes.
		Attributes <ul style="list-style-type: none"> <li>• <b>id</b> – the SNOMED CT identifier of a Concept or a Refset.</li> <li>• <b>rule</b> (self descendant descendant-or-self refset) <ul style="list-style-type: none"> <li>○ 'self' – the concept specified by the id.</li> <li>○ 'descendant' – any descendant subtype of the concept specified by the id.</li> <li>○ 'descendant-or-self' - the concept specified by the id or any of its any descendant subtypes.</li> <li>○ 'refset' – any member of the specified Refset.</li> </ul> </li> <li>• <b>exclude</b> (default=false) <ul style="list-style-type: none"> <li>○ 'true' – the set is the complement of the specified set (i.e. it excludes the set of concepts specified by the rule and id combination).</li> </ul> </li> <li>• <b>displayName</b> (optional) – the term of a SNOMED CT Description associated with the identified Concept or the name of the identified Refset.</li> </ul>

### 5.4.3 Expression constraint interpretation

Expression constraints can be interpreted in two distinct ways – semantically or literally.

Semantic interpretation of an expression constraint applies to the normal form of the candidate expression<sup>14</sup>. In contrast, literal interpretation is applied to the native close-to-user form entered, stored in a record or included in a message.

The intended interpretation can be specified using the ‘type’ attribute applied to the root node of the *expression constraint*. Since the general form of *expression constraint* representation is identical for both types, it is possible to apply a semantic interpretation to a *literal expression constraint* and vice-versa. However, as indicated in 5.3.2.4, the semantic constraints will usually specify ‘open’ interpretation of refinements, whereas literal constraints are more likely to be ‘closed’. Therefore, to ensure correct interpretation, separate constraints should be specified for each required type of interpretation<sup>15</sup>.

*Semantic expression constraints* specify and test requirements for the meaning of an *expression* to fall within the range of permitted meanings (based on subsumption testing). This type of constraint should be used wherever there is a requirement to:

- Limit the range of meanings expressed in a particular attribute
  - E.g. to ensure that a particular data element contains an expression that is valid for that type of record element.
- Confirm that a particular meaning is present in a composition or other part of a record
  - E.g. to confirm presence of information required by a dataset.
- Selectively retrieve record entries with a particular meaning
  - E.g. to display, report, count or otherwise analyse the contents of the record or an individual or the records of a population.

*Literal expression constraints* specify and test requirements for and/or prohibition of particular aspects of post-coordination. This type of constraint should be used wherever there is a requirement to:

- Limit the way in which a meaning is represented
  - E.g. to require or prohibit post-coordination of a particular facet of meaning.
- Specify a particular structured approach to data entry using (or preventing use of) particular post-coordinated attributes.

---

<sup>14</sup> Note: The logical requirement for semantic expression constraints to be applied to normal forms can be met in a variety of ways. Some of these require the transformation to be performed in real time. However, optimisation strategies described in IHTSDO documents can be applied to avoid the need for repetitive run-time transformations.

<sup>15</sup> Note: Earlier drafts of this document suggested that the interpretation would be specified by the referencing artefact rather than in the *expression constraint* itself. However, following more detailed assessment, this is no longer recommended.

#### 5.4.4 Expression constraint testing

Apart from the requirement for semantic expression constraints to be applied to normal form expressions, the process of testing conformance is the same.

Each component part of the expression is tested for conformance with the relevant constraints in the manner described in section 5.4.2.

Where alternative options exist, these are represented as sets of permitted patterns determined by subsidiary constraints. If a candidate pattern falls within the derived set of permitted patterns, it satisfies the constraint.

An expression, or part of an expression, only conforms if all its subsidiary nested components satisfy relevant subsidiary constraints.

#### 5.4.5 Expression constraint conformance requirements

The result of applying an *expression constraint* to an expression is either true (the expression conforms) or false (the expression does not conform). There is no indication of a particular type of partial conformance. However, an artefact that binds to *expression constraints* may specify different compliance requirements for particular constraints.

For example, a *selection support binding* may need to specify a set of commonly used expressions including common refinements. These common values may only be a small subset of a wider range of values that can be selected by the user if required. In this case, two (or more) separate expression constraints can be specified – one to specify the common requirements and one to constrain the range of possible extended entries. The referencing artefact (i.e. a form or data entry protocol) would indicate the way in which these are intended to be applied.

#### 5.4.6 Expression constraint inheritance and alternate forms

An *expression constraint* may be specified in full including all the permitted, required and prohibited values and patterns for a particular use. Alternatively an *expression constraint* can be specified as a refinement which is applied in addition to other, more general, constraints.

The alternative forms supported in the current specification are as follows:

- Short-form – including only refinements to inherited constraints
  - This would seem to be the most efficient in terms of authoring and management of content constraints.
- Intermediate-form – including refinements inherited from other *information model artefacts* but not restating rules that are part of the *concept model*.
  - This may be more efficient for processing and distribution:
    - It has no dependencies on other *expression constraints*.
    - It does not repeat common constraints that apply to all concepts with a particular domain.
- Long-form – including all applicable rules, including those inherited from the *concept model*.
  - This may be efficient for processing but entails a substantial amount of duplication with the risk of variance between constraints expressed against different versions.

## 5.4.7 Expression constraint identification and versioning

### 5.4.7.1 Identifiers

In the current specification, *expression constraints* are identified using a 'uid' attribute. This is generated using a standard algorithm for generation of a Universal Unique Identifier (UUID) which supports distributed authoring without identifier collisions. This is a developmental identifier in the sense that these identifiers can be generated and used by anyone without reference to the IHTSDO and without incurring a release maintenance obligation.

In future, if this approach to *expression constraint* representation is adopted more widely, it would be useful to migrate it into the general structure of the proposed 'SNOMED CT Release Format 2' (RF2). Based on the current RF2 proposal, the recommended XML form of *expression constraint* would be represented in an 'SRefsetMember' (i.e. a member of a Refset in which the values are all strings). This would mean that a released *expression constraint* would be assigned an SNOMED CT Identifier (SCTID) in the RefsetMember partition (with relevant namespace for a national Extension as required). This SCTID identifier would be the 'id' attribute of a row in the SRefsetMember table and the string attribute would contain the XML representation of the constraint.

### 5.4.7.2 Version tracking

In the current specification, the attributes 'active' and 'changeTime' are used to allow developmental version tracking. This follows the approach used in the proposed SNOMED CT Interchange Format update to align with the change Release Format 2 revision that replaces the enumerated status attribute with 'active' (1=active 0=inactive).

In the future, if the release version uses the SRefsetMember table (see 5.4.7.1), the effectiveTime would be used in place of the changeTime to denote the time at which a given version or state of this member became active (superseding previous rows with the same identifier).

## 5.5 Relationship to other representations

### 5.5.1 Reference Sets

The *expression constraint* representation uses the proposed SNOMED CT representation of Reference sets (*Refsets*). Since that specification is still subject to revision and approval, there is a possibility that revisions may affect this specification. Progress with *Refset* developments related to extensional definitions of membership would enhance the expressivity of the proposed constraints.

### 5.5.2 SNOMED CT Machine Readable Concept Model

The *expression constraint* representation uses the prototype SNOMED CT Machine Readable Concept Model (MRCM) as a source of *concept model* information. The *expression constraint* uses *Refsets* in the same way as the MRCM as a way to represent sets of permitted *concepts*.

The current specification uses a similar XML structure to the one used in the MRCM. However, there are some significant differences.

- a) The MRCM is represented by a more finely grained set of identifiable components. Thus each *concept model constraint* applies an identifiable test to all *concepts* in a particular domain. In contrast, a referencable *expression constraint* as specified in this document, is the collection of all the tests that are to be applied to determine if the expression conforms.
  - This difference is explained by the fact that the concern of this document is specifying, validating and retrieving instances of clinical data to meet a specified use case. Thus different constraints may apply to similar concepts to meet different business requirements. In contrast, the MRCM is asserting constraints that are always required to be true for any concept in a specified *concept model domain*.
- b) The MRCM is a flat set of constraints and does not support constraints applied to nested hierarchies. In contrast, an *expression constraint* must be able to constrain post-coordinated expressions that may include nesting.
  - As a result, a referencable constraint must either include nested constraint (or nested references to other constraints). For example, to specify constraints on laterality associated with a finding.
- c) The MRCM exists in two forms – a relational database containing a set of components and an XML changeSet format allowing communication and update between databases. In contrast, the *expression constraint* representation is specified only as an XML schema primarily intended for practical application (i.e. validation or retrieval of an expression). The repository storing *expression constraints* is assumed to treat each *expression constraint* as a versionable document (i.e. the structure within the XML schema need not be replicated by separate relational database tables).
  - Thus the schema treats each referencable *expression constraint* as a versionable component and tracks changes at that level.

### 5.5.3 Extended SNOMED CT Compositional Grammar

The Extended SNOMED CT Composition Grammar (see Appendix E ) has been used as a way to represent *expression constraints*. It was regarded by those who made the extensions as an interim measure, pending development of a more robust approach to terminology constraints.

The logical model underlying the *expression constraint* representation specified in this document is based closely on experience with the Extended SNOMED CT Composition Grammar. It extends the expressivity, consistency and usability of the grammar and has been tested to confirm that it supports all existing functionality of the grammar. For further information about of this evaluation see Appendix C – in particular Section C.5 to C.7.

## 5.6 Binding expression constraints to information model artefacts

### 5.6.1 Introduction

In order to be applied to an information model an *expression constraint* must be bound to an artefact in that model. There are several ways in which this binding could be done. The recommended approach is outlined in 5.6.2 and a reference-based representation is specified in 5.6.3. This specification is supported by examples of different uses in 5.6.4-5.6.6.

### 5.6.2 Recommended approach

Appendix B considers alternative ways in which *expression constraints* could be bound to information model artefacts such as a constrained model and patterns derived from the Care Components model. Of the options discussed, two seem likely to have practical value.

- References from the relevant information model artefact to one or more identified *expression constraints*.
  - For example, by including the unique identifier of an *expression constraint* in a model constraint applied to a particular coded field.
- Embedding complete *expression constraints* within the information model constraint.
  - For example, by including the XML *expression constraint* in a model constraint applied to a particular coded field.

The referencing approach is recommended as it allows reuse of constraints and also allows the constraints to be maintained across SNOMED CT releases without requiring revision of the information model artefact<sup>16</sup>. One consequence of this approach is a requirement to support a version-managed repository of *expression constraints* and to make this available as part of the LRA infrastructure.

The recommended reference based approach does not preclude the use of the embedded approach to meet some requirements. For example, it may be useful to derive a stand-alone representation that conveys a complete constraint (covering information model and terminology components). However, this should not be regarded as the primary form of representation.

---

<sup>16</sup> A change in a referenced terminology constraint implies a logical revision of the overall specification. However, separately versioning these two parts of the specification makes a clear distinction between revisions that have a structural impact and those that relate to an extension or restriction of an existing terminology binding. Intensionally specified *expression constraints* are liable to change with each release of SNOMED CT – whether or not the constraint itself is revised – therefore it makes sense to track the terminology related changes separately rather than assuming each change will be recorded in the information model artefact.

### 5.6.3 Representation of terminology bindings

Within the model the following form of *terminology binding* reference is proposed to refer to a *semantic constraint* or *literal expression constraint*.

**[fieldName].conformsTo("[constraintId]", "[effectiveTime]")**

The elements of this reference syntax are specified in detail in Table 12.

If the modelling formalism supports addition of comments, this facility may be used to include a title or a human-readable view of the constraint.

**Table 12. Details of terminology binding reference syntax**

<i>[field-name]</i>	<p>The name of a <i>field</i> in the Care Components model which is permitted to contain a value with a data type DT_CD.</p> <p>The field naming convention is as specified by the information modelling formalism. As a minimum, this formalism must ensure that taken together with its surrounding context in a constraint, the field-name uniquely identifies the field to which the constraint applies.</p> <p>For example, [BOUND_DATA_ELEMENT].<b>meaning.code</b></p>				
<i>conformsTo</i>	<p>A method applied to the <i>field</i> which returns a Boolean value based on the result of testing whether <i>expression</i> in an instance of that field conforms to the identified constraint.</p> <p><i>True</i> - if the <i>expression</i> conforms.</p> <p><i>False</i> - if the <i>expression</i> does not conform.</p>				
<i>[constraintId]</i>	<p>An identifier of the constraint to be applied. Initially work will assume the use of UUIDs in the standard string form (lower-case enclosed by curly braces).</p> <p><u>Example</u></p> <p>"{8a5c1d8f-86cc-49e6-a3a1-28a13a95d914}"</p> <p><u>Notes:</u></p> <p>Future migration to use of SCTIDs may be supported if the IHTSDO adopts the proposed constraint formalism and allocates a partition identifier for constraints.</p> <p>The identifier may be replaced with the full text of the constraint in dereferenced representations. However, this form may not be supported in some model representation forms.</p> <p>The identifier refers to a component which may be revised over time. If the constraint is locked to a specific version the effectiveTime should be used to specify this.</p>				
<i>[effectiveTime]</i> OPTIONAL	<p>An optional timestamp specifying the version to be used. If this is not specified, the most recent available revision is applied.</p> <p><u>Format:</u> as ISO date-time string without symbols. If the time component is omitted it defaults to the end of the day (23:59:59 UTC).</p> <p><u>Examples:</u></p> <table> <tr> <td>"20090115"</td><td>(15-January 2009 at 23:59:59 UTC)</td></tr> <tr> <td>"20090115171501"</td><td>(15-January 2009 at 17:13:01 UTC)</td></tr> </table>	"20090115"	(15-January 2009 at 23:59:59 UTC)	"20090115171501"	(15-January 2009 at 17:13:01 UTC)
"20090115"	(15-January 2009 at 23:59:59 UTC)				
"20090115171501"	(15-January 2009 at 17:13:01 UTC)				

#### 5.6.4 Interpretation of different types of constraints

Several types of terminology binding are discussed in Section 4.2.1 and differences in interpretation of these between *semantic expression constraints* and *literal expression constraints* are explained in 5.4.3. However, the form of the *constraint references* and the *expression constraints* themselves are the same for both these types.

The way the referenced *constraint* is applied depends on the type attribute in the *expression constraint*<sup>17</sup>.

Multiple constraints may be applied using the conventions of a constraint language appropriate to modelling environment (e.g. OCL (Object Constraint Language) or HL7 GELLO – Guideline Expression Language).

Multiple *constraint references* must be used where constraints of different types are required and may also be specified to indicate constraints inherited from a more general model and specific constraints applicable to a specialisation of the model.

##### Examples

`meaning.code.conformsTo("{8a5c1d8f-86cc-49e6-a3a1-28a13a95d914}")`

Requires that the *expression* in the *meaning* field conforms with the most recent version of the *expression constraint* identified by the UUID "{8a5c1d8f-86cc-49e6-a3a1-28a13a95d914}".

`meaning.code.conformsTo("{8a5c1d8f-86cc-49e6-a3a1-28a13a95d914}", "20090115")`

Requires that the *expression* in the *meaning* field conforms with the structure specified by the current version, at 23:59:59 UTC on 15-January-2009, of the *expression constraint* identified by the UUID "{09b85735-68ed-4607-ae81-b41a02626f1c}".

`value.code.conformsTo("{09b85735-68ed-4607-ae81-b41a02626f1c}")`

Requires that the *expression* in the *value* field conforms with the most recent version of the *expression constraint* identified by the UUID "{09b85735-68ed-4607-ae81-b41a02626f1c}".

#### 5.6.5 References to terminology constraints within queries

The logical representation of conformance tests specified in 5.6.3 should be applied to specify these constraints within a query. However, the precise syntax may need revision to appropriately trigger appropriate processing.

#### 5.6.6 Terminology binding references to information model artefacts

To support *constructor bindings* and *composite semantic constraints* and *retrieval bindings* there is a requirement to be able to refer to specific instances of classes derived from the Care Components Model.

<sup>17</sup> Note: The previous version of this specification included a *constraintType* parameter in the *conformsTo* function to specify the type of testing to be applied. However, further analysis indicated that the *expression constraints* intended for the semantic and literal expression constraints are not interchangeable due to difference the way refinement constraints need to be specified (see 5.3.2.4).



## 6 Terminology binding tooling requirements

### 6.1 Introduction

This section is concerned with requirements for tooling to support the development and maintenance of terminology bindings as an integral part of the design of clinical content models. Section 6.2 outlines the design process to identify those areas likely to require tooling support. Sections 6.3 to 6.14 specify terminology related tooling requirements to support these processes.

### 6.2 Care record design – foundations and development

#### 6.2.1 Care record design foundations

The LRA Care Components model is a foundation for consistent representation of clinical information. It specifies a general set of derived classes for different types of terminology bound clinical information. Each of these derived classes is associated with a set of high-level *expression constraints*.

The model has been further elaborated by terminology binding guidance concerned with commonly used types and patterns of clinical information. These elaborations use classes from the LRA Care Components model as a starting point for more restrictive *expression constraints* that match the requirements for recording particular items of information. The resulting constrained classes are used as building blocks to construct patterns that address requirements for particular sets of interrelated information.

Terminology binding guidance, and the associated machine processable *expression constraints*, form a generic layer built on the foundation of the LRA Care Components model. Further elaborations to meet specific use cases need to be constructed in such a way as to further specialise these guidelines and constraints.

#### 6.2.2 Care record design process

The process of care record design should apply the LRA Care Components and the related Terminology Binding advice to meet specific clinical requirements.

The following notes summarise the design process outlined in Figure 5.

1. Starting with a requirement (or set of requirements) the first step is to look for similar requirements that have already been addressed.
  - a) If an existing design fully meets the identified requirement it can be adopted as is.
  - b) If an existing design partially meets the requirement it can be used as a starting pattern for a new design. This can then be extended to address each of the differences between the new requirement and the starting pattern.
    - Each difference between an existing pattern and the new requirement can be addressed as an additional requirement by recursively following the same design process (step 1).
2. If none of the existing designs meets the requirement, the next step is to determine if the required item can be expressed by one of derived BOUND\_DATA\_ELEMENT classes in the model.

1. If so, the relevant class is selected and the appropriate expression constraint is created (or selected) and bound to the constrained element.
  - Any items of additional related information can be considered as separate requirements (step 1)
    - The results of designing the subsidiary items are then linked to the BOUND\_DATA\_ELEMENT.
2. If the required item cannot be expressed by a BOUND\_DATA\_ELEMENT it may be possible to subdivide the required information into subsidiary items which can be represented by inter-related BOUND\_DATA\_ELEMENT.
  - If so each of these is considered as a distinct requirement (step 1)
    - and the resulting BOUND\_DATA\_ELEMENT may be linked together to meet the composite requirement.
  - Otherwise the required information must be represented by an UNBOUND\_ELEMENT.

### 6.2.3 Parallel development and risk of design divergence

The care record designs for many different specialties will be developed in parallel. These specialties will have different but overlapping requirements. This creates risks of inadvertent divergence and duplication of effort.

To minimise the risk of divergence all those engaged in the design process must follow a consistent set of principles.

- The range of alternative options available to designers needs to be limited. For any specific type of requirement there should be one clearly preferred approach.
- Where alternative approaches are required to meet specific requirements they should still follow common patterns, avoid inadvertent duplication and offer clear advice on consistent use.
- A new design may be needed to add detail to an item that could otherwise be represented using a more general design. The specific design should be specified in a way that retains compatibility with queries based on the more general design.
  - Where possible, the more detailed pattern should be a valid extension of a simpler pattern.
  - In other cases, a transform or an extended query that meets the general retrieval requirement should be documented.

To enable effective sharing, comparing and aligning of care record designs the products of each design stream:

- Must be delivered in a consistent structured form. Wherever possible, these products should be able to be automatically validated and readily compared with deliverables that address similar or overlapping requirements.
- Must be structured in ways that allow them to be reused in parallel and subsequent activities. Therefore, modular representations should be favoured rather than monolithic specifications.
- Must include an accessible log of design issues, discussions and decisions. Otherwise, it is likely the same issues will arise again – possibly leading to an arbitrarily different decision.

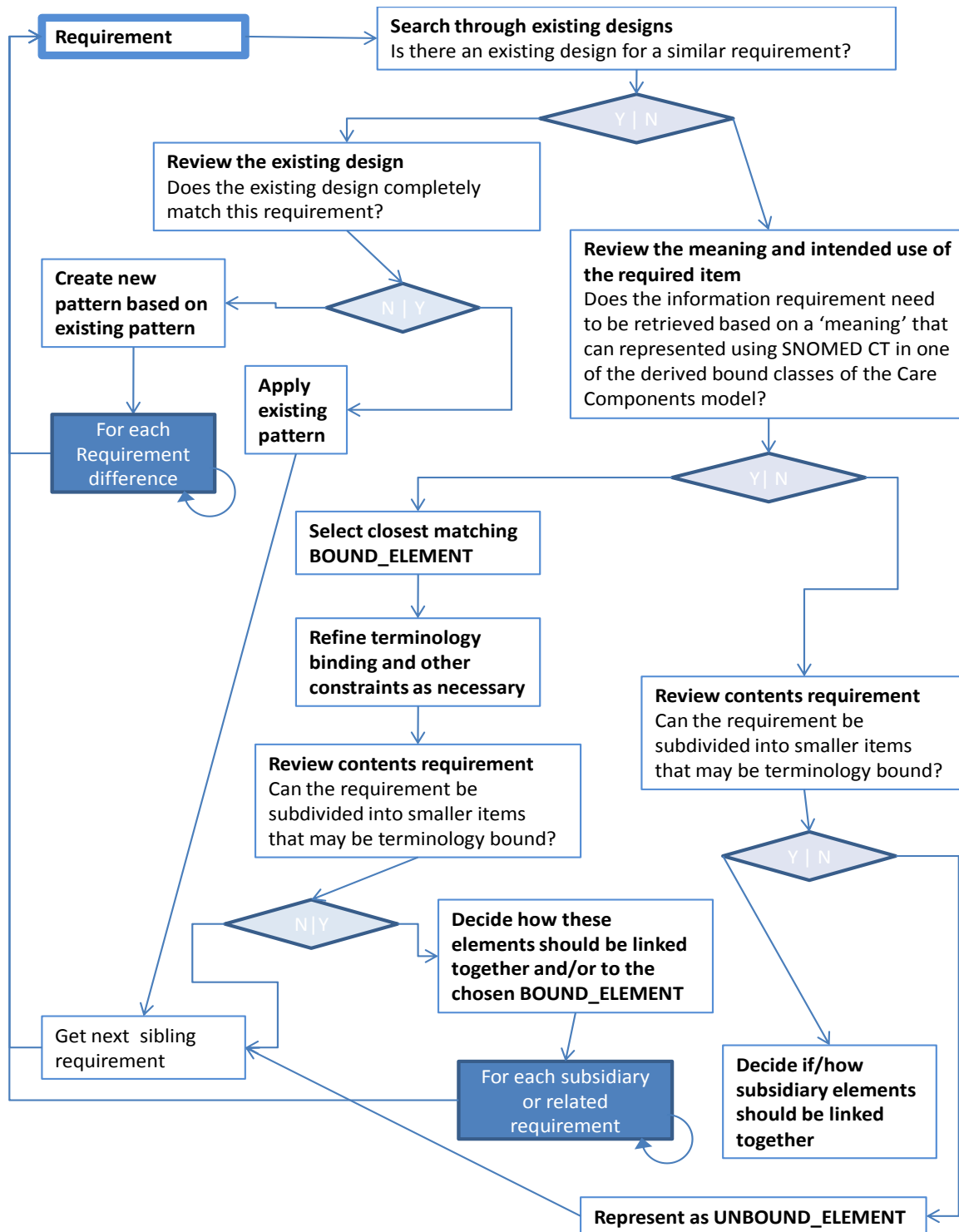


Figure 5. Summary of the care record design process

## 6.3 Terminology related tooling requirements to support design

### 6.3.1 Overview of tooling requirements to support care record design

Tools are required to support this process in ways that:

- Provide effective and appropriate access to existing work:
  - Including
    - Foundational material;
    - Work already completed in other domains;
    - Work in progress in other domains.
  - Supporting
    - Easy access to relevant guidance documentation;
    - Human review and machine processing of existing relevant constraints;
    - Ability to reuse similar existing designs (or parts of designs) as starting patterns for new designs to meet similar requirements.
- Facilitate creation and maintenance of consistent specifications:
  - Including
    - Machine processable information model constraints;
    - Associated terminology bindings (especially *expression constraints*);
    - Human-readable documents designed for relevant audiences;
    - Machine and human-readable instance examples
      - Demonstrating both valid and invalid instances.
- Enable effective review, comparison and version management
  - Including
    - Validate output to confirm consistency
    - Comparing and aligning similar information in different domains
    - Taking releases of SNOMED CT content and changes to the SNOMED CT concept model
- Enable practical use by the relevant communities
  - Including
    - Implementation in clinical systems
    - Validating instance data to determine
    - Query development to enable appropriate retrieval and reuse

The terminology related tools and services required to contribute to meeting these requirements are outlined in the following sections. In most cases these tools need to be closely integrated with other parts of the LRA tooling infrastructure rather than separate tools. Therefore, the descriptions in this section are in many cases incomplete summaries. Future LRA tooling developments needs to build these features into the overall specification of the required toolset.

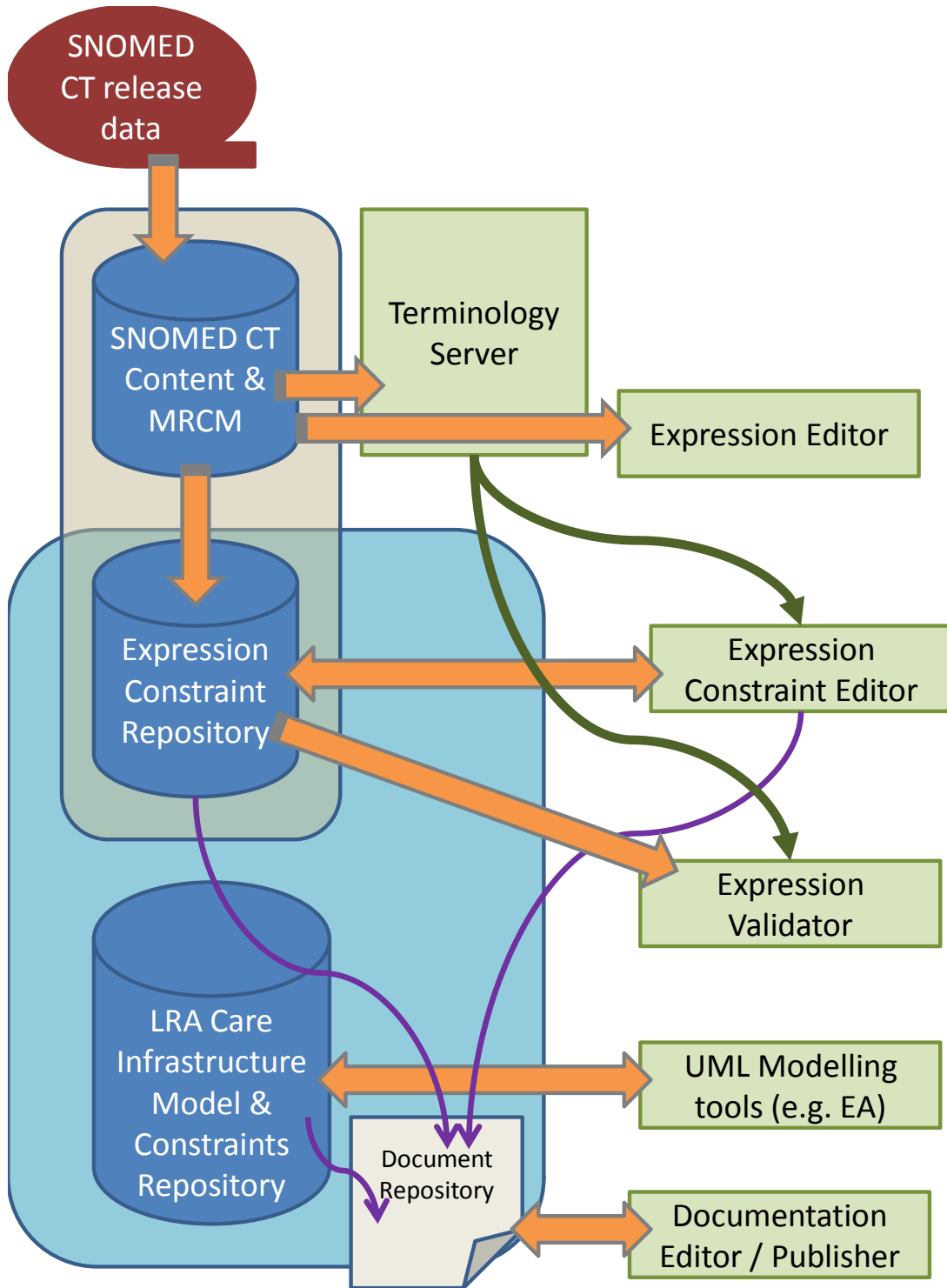


Figure 6. Overview of terminology binding related tooling requirements

## 6.4 General terminology server functions

The following general terminology server and browser functions are essential to support LRA Terminology binding:

- Access to SNOMED CT content including:
  - UK Extensions
  - Previous releases (to the extent required to meet maintenance and historical review requirements)
- Access to SNOMED CT Components by ConceptId (and Read Code)
  - All concepts, descriptions and relationships
- Rapid searches of SNOMED CT content
  - Search approaches including:
    - Complete term matches
    - Partial term matches
    - Multiple (unordered) word matches
    - Regular expression matching
    - Lucene search features
  - Search filtering by:
    - Component status
    - Language and dialect
    - Subsets of Concepts and Descriptions
    - Subtype descents of one or more specified concepts
    - Expression constraints
  - Hierarchical navigation using
    - SNOMED CT subtype hierarchy
    - Alternative navigational hierarchies
- Rapid<sup>18</sup> subsumption testing applicable to:
  - Pre-coordinated expressions
  - Post-coordinated expressions
    - Either by normal form generation or by formal classification.
- The way(s) in which SNOMED CT can represent a required type of information  
Including:
  - Matching focus concepts available in different domains (i.e. clinical findings, observable, procedures etc).
  - Refinements supported by the *SNOMED concept model* that could be permitted or constrained for the particular use case.

---

<sup>18</sup> An average pre-coordinated subsumption test should take no longer than 0.1 milliseconds.

- Relevant context information that could be permitted or constrained for the particular use case.

## 6.5 *Expression constraint* editor

### 6.5.1 Creating new expression constraints

Authorised users should be able to create *expression constraints* either de-novo or as refinements of existing *expression constraints*.

- It is essential that authoring policies are established to determine rights and responsibilities in relation to editing.
  - When the expression editor is used with a shared expression constraint repository these policies must be strictly enforced. This requires the repository to support versioning, change control and other features of configuration management.
  - The identity of the author of an *expression constraint* (or a revision of such a constraint) needs to be tracked along with the time of such changes.

It must be possible to start the process of creating a new *expression constraint* by cloning an existing constraint and editing that clone. Three distinct types of editing may be applied to a cloned *expression constraint*:

- Refinement only – ensuring that the resulting *expression constraint* is a valid restriction of the original *expression constraint*.
- Extension only – allowing a more general constraint to be created in a way that ensures the original *expression constraint* is a valid restriction of the new *expression constraint*.
- Boiler plate – allowing unconstrained changes to be carried out. The *original expression constraint* provides a starting point for the development of the new *expression constraint* but there is no persisting relationship between the constraints.

The editing process must allow the construction valid *expression constraints* using all the features supported by the schema and associated specification.

- In particular, it should provide appropriately constrained searches for attributes and values to ensure these conform to the concept model (and where relevant with referenced *expression constraints*).

The editor must be able to access and interpret the membership of Refsets.

- The editor should also be able to edit Refsets either directly or by invoking a separate Refset Editor.

Expressions must be accessing in the XML representation specified in this document (and by the associated XML Schema). This enables transformations to be applied to support alternative human-readable renderings.

## 6.5.2 Modifying existing expression constraints

Authorised users should be able to edit *expression constraints* using facilities similar to those used to create new *expression constraints*.

Modifications should be constrained where relationships between constraints are specified in the repository. It should not be possible to modify a constraint so that a constraint that is declared to be a restriction on another constraint ceases to be a valid restriction of that constraint.

All revision to *expression constraint* must be stored in a repository with appropriate version management services that supports roll-back. Where a shared repository is used, conflict detection and resolution must also be supported, and this must be subject to appropriately defined authoring policies.

## 6.6 Expression constraint repository

### 6.6.1 Introduction

The recommended approach to representation of *expression constraints* requires that each *expression constraint*:

- Has a unique identifier
- Is maintained within a repository which tracks changes and retains previous versions.
- Is accessible from the repository
  - In a read only 'release' form for implementation;
  - In an updatable form subject to appropriate access control.

### 6.6.2 Locating existing expression constraints

Tools used to design *information model artefacts* must, as part of the design process, find existing *expression constraints*, so that these can be reviewed and reused as appropriate.

This has the following implications for the design and indexing of *expression constraints*.

- It must be possible to search for *expression constraints* based on relevant parameters including:
  - Focus concept constraints
  - Context wrapper constraints
  - Use of refinements
  - The information models artefact that reference them
- Access mediated by using SNOMED CT searches should be possible to enable *expression constraints* involving concepts related to a selected concept (e.g. supertype, subtype or sibling), to be retrieved where required.(e.g. when there is no direct match).
- An *expression constraint* must have a name and/or short human-readable rendering that can be displayed in response to a search.
  - If feasible, a short human-readable rendering derived from the constraint is preferable as this reduces the risk of inappropriate naming.
  - A stored name (or title) may be more practical for complex constraints.



### 6.6.3 Reviewing existing expression constraints

It must be possible to display relevant information about a selected *expression constraint* to allow it to be reviewed to determine if it meets or assists in meeting a requirement.

This has the following implications for the design of *expression constraints* and the facilities available to review them.

- It must be possible to display an accurate human readable rendering for a selected constraint, so it can be evaluated against a requirement.
  - For simple constraints, a transformation to the Extended Compositional Grammar may meet this requirement.
  - For more complex constraints, an initial high-level rendering may need to allow drill-down to view subsidiary constraint.
- It must be possible to display notes about a selected *expression constraint* to clarify its intended use. It should also be possible to add to these notes, to document rationale for use, known issues and proposed changes.

#### 6.6.3.1 Applying and revising references to selected expression constraints

- It must be possible to select and reference a specific version of an *expression constraint*.
  - It should be possible to choose to reference an *expression constraint* in a manner which automatically applies the most recent version at run time.
- It must also be possible to review all referenced *expression constraints* that have been updated since the time of a previous review.
  - During this review, it must be possible to choose to update the reference, so the new version of the constraint is applied.
- It should be possible to review *information artefacts* in which a selected *expression constraint* has been used.

## 6.7 Version management

### 6.7.1 Managing versions of terminology binding and the associated model

- Tracking changes at the *expression constraint* and subsidiary component levels.
- Managing and synchronising changes to *expression constraints* and *information model constraints*.
- Merging changes from different work streams.
- Peer review checking for overall consistency.

### 6.7.2 Managing information model changes

- Consequences of information model changes for terminology binding should be analysable and open for manual review where relevant.
- Minor changes to information model should not require manual reassertion of all bindings.

### 6.7.3 Managing terminology updates and alignment

- Content updates
  - International Edition releases
  - UK Extensions releases
- *Subset/Refset* updates
  - Managing the consequences of changes to set membership as they affect terminology bindings.
- Concept Model updates
  - International Edition
  - UK Extensions

### 6.7.4 Identifying significant changes in SNOMED CT

The following changes in SNOMED CT are liable to be particularly significant and need to be detected, reported and used to specify relevant updates to *expression constraints*:

- Content changes affecting existing *expression constraints*.
- Content changes allowing workarounds to be replaced by more robust and consistent solutions.
- Concept model changes affecting existing *expression constraints*.
- Concept model changes allowing workarounds to be replaced by more robust and consistent solutions.

### 6.7.5 Managing known issues

The tools should track known documented issues arising from limitation of the terminology, the information model or implemented applications. The artefacts affected by these issues should be references to

- Support consistent application of workarounds for known terminology issues.
- Facilitate systematic updates to support migration from 'work rounds' to more robust solutions.

## 6.8 *Expression constraint validator*

The tools must support testing of the following aspects of an *expression constraint* to be tested:

- Structural validation against the relevant XML Schema.
- Validity as a restriction of the SNOMED CT concept model.
- Validity as a restriction of other specified *expression constraints*.
- against a specified *expression constraint* or set of *expression constraints*.

The tools should also enable:

- Confirmation of the validity of display names in a *constraint expression*.
- Correction and/or translation of display names in a *constraint expression*.

## 6.9 *Expression* example editor

Expression examples are required for documentation and in order to test the effect of constraints.

The terminology tools must enable creation of valid post-coordinated expressions either de-novo or based on pre-existing expressions.

- Post-coordinated expression creation should be supported by appropriately constrained options to
  - Select refining attributes
  - To search for and apply values to attributes
- The options for expression creation should be constrainable by
  - Defining and qualifying relationship
  - Concept model constraints
  - Expression constraints

The expression example editor must be able to render expression in accordance with the SNOMED CT Compositional grammar.

Expressions should also be accessible in an XML form that complies with the abstract model for SNOMED CT expression as this enables alternative transformations for efficient processing and to add mark-up to human-readable renderings.

## 6.10 *Expression* validator

The tools must support testing the validity of an *expression* against a specified *expression constraint* or set of *expression constraints*.

The tools should also

- Confirmation of the validity of display names in an expression.
- Correction and/or translation of display names in an expression.

## 6.11 *Expression constraints* and human-readable documentation

Tools for supporting creation, editing and references to *expression constraints* should also support integration with human-readable documentation. This integration should:

- Allow appropriately marked up human-readable renderings of *expression constraints* to be included in documentation.
  - Insert the unique identifier of the *expression constraint* in the relevant part of the source documentary form from which documents are generated.
- Enable this identifier to be used
  - To support hyperlinks from published document to the relevant machine processable constraints.
  - To allow human-readable renderings of *expression constraints* in the source documentary form to be validated against and where necessary updated to align with the machine processable representation.

The source form for human-readable documentation of *expression constraints* and care record designs should be maintained at a level of granularity that is aligned with the machine processable artefacts. This allows maintenance to be synchronised and avoids a small change to constraint requiring an update to a single large document. This approach will also allow the repository or documents to grow to accommodate new patterns as they are created<sup>19</sup>.

---

<sup>19</sup> Use of the Darwin Information Typing Architecture standard (DITA - <http://dita.xml.org/>) during development of the Terminology Binding Key Clinical Data and Common Recording Patterns documents has demonstrated the feasibility of this type of approach to integrate topic level human readable documentation with machine processable representations.

## 6.12 Terminology mediated access to care record designs

### 6.12.1 Terminology based indexing of the care record design repository

A key part of an effective set of design tools is effective storage, indexing and retrieval of design guidelines. SNOMED CT terminology binding can contribute to this process by allowing existing patterns relevant to particular types of concept to be located.

Section 6.6.2 recommends an approach that allows *expression constraints* to be accessed by selecting relevant SNOMED concepts using a terminology based search. This approach should be extended to allow access:

- To the information model artefacts from which such constraints are referenced;
- Related guidelines and designs documentation.

This functionality can be supported by combining:

- The approaches to *expression constraint* indexing recommended in 6.6.2; and
- A lookup table linking each *expression constraint* identifier to all the information model artefacts and documents in which it is used.

In this way terminology based searches should be able provide access to the following information.

- Existing design pattern information associated with the concepts found.  
Including:
  - Information model class and field usage and constraints.
  - Terminology component usage and constraints.
  - Related documentation of:
    - the use case to which the patterns apply;
    - rationale for the design decisions;
    - known issues and work-rounds and/or proposed future evolution.
- Existing design patterns associated with similar concepts:  
Information available for a selected 'similar concept' should be as for patterns associated with the concept itself (see above).  
Similar concepts listed should include:
  - Supertypes (i.e. more general concepts)
  - Subtypes (i.e. more specific concepts)
  - Members of the same *concept model domain* (e.g. other laboratory procedures).

### 6.12.2 Terminology-based approach to care record design

The terminology-based approach to locating constraints and documentation outline in the 6.12.1 would enable many of the steps in the care record design process outlined in Figure 5 to be initiated from SNOMED CT searches. This approach is illustrated by the example in Table 13.

Selection of a *concept* would return a list of existing constraint patterns applied to the selected concept or similar *concepts*. Relevant patterns and concepts should then be considered, along with the related design notes and discussions. An existing pattern would then be chosen and, if necessary, elaborated to meet the specific requirement. The resulting materials, including discussion of the rationale for revisions, would then be added to the repository indexed by relevant SNOMED CT *concepts*.

This process can be extended to link together relevant additional information associated with more specific types of action or finding. For example, an activity such as 'intravenous injection' needs to be associated with the substance injected.


**Table 13. Illustration of design process for a pattern to represent a procedure**

<u>Summary of requirement:</u>	
<b>Representation of information related to a caesarean section operation</b>	
1. Select a relevant initial pattern	<ul style="list-style-type: none"> <li>• A search of SNOMED CT retrieves 11466000   caesarean section   as the most general relevant concept.</li> <li>• This concept is a subtype of various concepts including 387713003   surgical procedure   , 386637004   obstetric procedure   and 236973005   delivery procedure  </li> <li>• Existing patterns for these and related concepts should be made available for consideration.</li> <li>• If there are no other relevant patterns, the starting point is the general pattern for ACTIVITY_ELEMENT (as the concept is a subtype of 71388002   procedure  ).</li> </ul>
2. After selecting a starting point, further decisions informed by the specific requirements need to be made:	<ul style="list-style-type: none"> <li>• Are context constraints required?</li> <li>• Is this pattern concerned only with caesarean sections that have been done or is it also used for recording a planned caesarean section.</li> <li>• Are refinement constraints needed?</li> <li>• By default, all refinements permitted by the <i>concept model</i> are allowed but this can be constrained as necessary.</li> <li>• What additional detailed, subsidiary or related information is required?</li> </ul>
3. If additional detailed information is needed,	<ul style="list-style-type: none"> <li>• The design process is repeated recursively for each item.</li> <li>• The items are linked together using instance of COMPONENT_RELATIONSHIP_ELEMENT with appropriately constrained meaning values.</li> </ul>

### 6.12.3 Representing similar information in different circumstances

The type of terminology-based elaboration illustrated in Table 13 is a valuable starting point for pattern development. However, it has limitations, since different levels of detail may be required to record the same general concept from different perspectives. This point is illustrated in Table 14, which summarises different levels of detail that may be required to represent an 'emergency caesarean section' from a variety of perspectives.

**Table 14. Different views of the same procedure**

Representation of past history of caesarean section	4908009   history of   363589002   associated procedure   = <b>274130007   emergency caesarean section  </b>
Summary representation of a caesarean section from the maternal perspective	Date: 01-12-2008 Time: 14:15 <b>274130007   emergency caesarean section  </b> <b>169826009   single live birth  </b> "No complications"
Outline of detailed record of a caesarean section operation (from draft NHS openEHR archetype for caesarean section)	
Summary representation of caesarean section from the perspective of the child	<b>407615002   born by emergency caesarean section  </b>

The fact that a patient had an 'emergency caesarean section' can be recorded with a single SNOMED CT expression, including an appropriate temporal context (e.g. past history or current).

- The simplest of the requirements shown in Table 14 can be readily accommodated by using a very general record entry in which the meaning is represented by the relevant SNOMED CT expression.
  - Record entries would include the date and time of the procedure and a mechanism to support references to participants (e.g. surgeon, assistant, etc).

- Operative notes and summaries related to a caesarean section need to include additional information over and above the fact that the procedure was done.
  - These may include observations about the outcome (e.g. the successful delivery of a live baby) and other subsidiary procedures (e.g. the administration of the anaesthetic).
- Each of the additional items can be represented as a separate entry conforming to constraints applicable to the particular type of entry. However, a specification of a particular set of additional data associated with a caesarean section requires both the perspective and the nature of the recorded information to be considered.

Therefore, a flexible approach should takes account of the circumstances in which the information is required, as well as the types of SNOMED CT concepts that need to be recorded. Thus each constrained information structure needs to be indexed by the circumstances of recording and the nature of the recorded information.

Table 15 illustrates the way in which simple generic representation applicable to any procedure is augmented by some requirements that are particular to refinements of the concept and others that are determined by specific uses.

**Table 15. Example of relevance of concepts and circumstances to information structures**

<b>Type of Concept</b>	<b>Circumstances (examples)</b>		
	<i>General clinical history</i>	<i>Procedure notes</i>	<i>Discharge note/summary</i>
71388002 / procedure /	Single activity entry within relevant context wrapper and with refinements permitted subject to constraints	Activity entry structure with any additional detail relevant to the specific procedure.	
387713003 / Surgical procedure /		Additional detail relevant to surgical procedure (e.g. anaesthetic, etc)	Additional detail linked directly to the procedure (e.g. per-operative information)  Addition detail relevant to overall care (e.g. relevant post-operative information, biopsy results).
236973005   Delivery procedure		Additional detail relevant to delivery of baby.	Additional detail related to delivery and subsequent health of the baby to the extent needed for any maternity stay.
11466000 / Caesarean section /		As for combination of surgical procedure and delivery procedure plus additional detail specific to caesarean section procedure.	As for combination of surgical procedure and delivery procedure plus any additional detail specific to caesarean section hospital stays.



## 6.13 Terminology binding and other LRA tooling requirements

### 6.13.1 Integration of terminology binding tools

Ideally, terminology binding should be fully integrated within tools used for creating or editing information model artefacts. This would allow decisions on modelling and use of terminology components to be made in a harmony.

Without this integrated approach, those designing the information model artefacts are liable to make incorrect assumptions about what the terminology can or should do. This makes subsequent terminology binding much more difficult and leads to inconsistent results.

Some of these requirements for integration can be met by providing interfaces from the tools used for information model design to appropriate terminology aware tools. For example,

- An interface to the IHTSDO Workbench might enable some of the Refset, search and Description Logic functions to be supported.
- Other interfaces might allow *expressions* and *expression constraints* created with other tools and browsers to be tested or utilised within the LRA development environment.

### 6.13.2 Interim approaches

There is a requirement for decisions to be made about the way in which different elements of the tooling support for the LRA are expected to fit together.

From the view point of a designer, an integrated solution dealing with all aspects of clinical content design is the optimal approach. The remainder of this section provides an initial outline summary of the design of such a tool. However, other relevant perspectives must also be considered.

These include:

- Using existing off the shelf products (e.g. Enterprise Architect)
- Requirements for version management, access control and a shared development repository.
- The need for rapid progress based on the LRA timetable.

These other factors may act as counter balance to development of an integrated solution. Therefore, an interim approach based around more loosely coupled tools that already address (or can be rapidly revised to address) specific aspects of the requirement may be more appropriate.

## 6.14 Implementation aspects of bindings

The following list summarises issues related to implementation.

In order to be implemented LRA care record designs and their associated *expression constraints* must be made available to implementers, conformance testers, secondary data users and end-user systems. These organisations will then need to apply these artefacts in the following ways:

- Conformance testing:
  - Data content specifications incorporating *expression constraints* (by reference) must be sharable with conformance tester for applying the tests.
  - They must also be provided to system implementers to support development and their own pre-conformance checks.
- Assisting consistent use of implemented applications:
  - Data content specifications incorporating *expression constraints* may be used to constrain (semantic or structural constraint) or assist (selection support) data entry.
- Assisting consistent implementation:
  - A future (fully LRA compliant) application providing a revised content specification may allow semi-automated revision and reconfiguration of a system. (OPTIONAL)
- Assisting consistent terminology updating:
  - *Expression constraints* are constraints on terminology and can also be used as predicates in queries. When SNOMED CT content is updated and particularly when the *content model* is revised, *expression constraints* should be used to detect, review and assist resolution of the impact of these changes.
- Enabling construction of queries including *expression constraint* predicates:
  - This requires tools that enable construction of *expression constraints* to be integrated with query writing software.
  - The query tool integration requirement is similar to the requirement for integration with content design tools.
- Supporting execution of queries including *expression constraint* predicates:
  - This requirement is similar to conformance testing and validation requirement. However, it applies at run time rather than in a test environment.
  - To support any or all of these cases, the form distributed needs to be readily processable by applying documented algorithms. In addition, provenance and version information must be clear and appropriately secured.

## Appendix A Expression Constraint Types - Details and Examples

### A.1 Semantic expression constraints

Semantic expression constraints restrict what it is possible say.

A *semantic expression constraint* asserts that only terminology expressions that have a meaning that falls within a domain specified by the constraint can be applied as values to a particular node of field in the reference model.

A *semantic expression constraint* is concerned with ensuring that the expression used conveys a meaning that is appropriate to the structural component.

#### Examples

- A node representing diagnosis might be required to be a value that is a subtype of 64572001 | disease |

A *semantic expression constraint* may explicitly require or exclude a particular facet of information to be expressed.

#### Examples

- A node describing a procedure on a kidney might be required to specify laterality.
- A node representing the action of administering a drug may be required to exclude any mention of the substance administered, as this may be expressed in a separate node.

Thus 32282008 | subcutaneous injection | would be permitted but not 308755006 | subcutaneous injection of insulin |.

A *semantic expression constraint* is not concerned whether that meaning is conveyed as a pre-coordinated or post-coordinated expression.

#### Examples

- If 71620000 | fracture of femur | is valid then an equivalent post-coordinated expression would also be valid (e.g. 125605004 | fracture of bone | : 363698007 | finding site | = 71341001 | bone structure of femur | ).
- The requirement to specify laterality for a kidney procedure would be met by any of the following
  - a) 108022006 | kidney excision | : 272741003 | laterality | = 7771000 | left |
  - b) 65801008 | excision | : 405813007 | procedure site - Direct | = ( 64033007 | kidney structure | : 272741003 | laterality | = 7771000 | left | ) }
  - c) 65801008 | excision | : 405813007 | procedure site - Direct | = 18639004 | left kidney structure |

## A.2 Literal expression constraints

Literal expression constraints restrict how it possible to say something.

A *literal expression constraint* specifies the permitted or required post-coordination of an expression that may be applied to the value of a particular field.

### Examples

- Prohibition of any post-coordination:

71620000 | fracture of femur | is permitted but semantically equivalent post-coordinated expressions (e.g. 125605004 | fracture of bone | : 363698007 | finding site | = 71341001 | bone structure of femur | ) are not permitted.

- Requirement for the substance responsible for an allergy to be represented by post-coordination:

106190000 | allergy | : 246075003 | causative agent | = 373270004 | penicillin -class of antibiotic- | is permitted but the semantically equivalent pre-coordinated concept 91936005 | allergy to penicillin | is not permitted.

## A.3 Semantic and literal expression constraints compared

*Literal expression constraints* and *semantic expression constraints* are interrelated.

- A literal expression constraint imposes some constraints on semantics (e.g. if post-coordination is not permitted, then meanings for which no pre-coordinated concept exists in SNOMED CT cannot be represented).
- A semantic expression constraint may indirectly constrain the literal expression (e.g. if the prohibiting a specific semantic facet prohibits that aspect of post-coordination).

Despite these interdependencies, the use of *literal expression constraint* can add value to a *semantic expression constraint* and address different requirements.

*Literal expression constraints* can be used to limit variation in forms of expression and thus simplify implementation.

### Example

- A *literal expression constraint* might prohibit use of the 'associated with' attribute to prevent creation of complex expressions being used to combine multiple findings in a single field. This would not prohibit use of a concept that has an 'associated with' attribute in its definition.

*Literal expression constraints* can also be used to require explicit post-coordination where issues with the consistency and completeness of SNOMED CT content are expected to interfere with a mission critical processing requirement.

### Example

- A *literal expression constraint* may require 246075003 | causative agent | to be post-coordinated rather than using a pre-coordinated *expression* that appears to represent the same meaning. This would avoid overlooking a

known allergen due to a missing or erroneous 246075003 | [causative agent](#) | attribute in a concept definition.

## Appendix B Additional Terminology Binding Requirements

### B.1 Information model artefact representation requirements

To support terminology binding, it must be possible to associate bindings with relevant *information model artefacts*.

- Many types of binding can be represented by including the binding as meta-data of the relevant *information model artefact* or by including a reference from the artefact to a terminology binding representation.
  - A reference based approach is likely to be more effective than explicit inclusion of the binding within the artefact for several reasons:
    - The same bindings may be shared by several artefacts.
    - Allows the terminology aspect of the binding to be maintained without making explicit changes to the structural aspect of the model. Terminology release version changes can be managed and tracked while kept distinct from structural revisions.
    - Allows decoupling of the representation of the terminology binding from the information model formalism. The same terminology binding representations can be utilised even if the internal modelling formalism used by an application differ.
- Binding types that apply to multiple related *fields* (see Table 9) have some specific additional requirements:
  - It must be possible to reference a terminology binding from a class or collection of related classes rather than from an individual *field* of a class.
  - Terminology bindings must be able to reference *information model artefacts*. For example, to reference a set of related *fields* that are combined by a *constructor binding* and tested by a *composite semantic constraint*.
    - The style of representation of references to *information model artefacts* depends on the information model formalism<sup>20</sup>.

---

<sup>20</sup> For example, *openEHR*, has an established approach to using a short form of XPATH to refer to constituent nodes using local identifiers and names. Similarly, HL7 Version 3 has a *templateId* attribute which provides an anchor for references from which XPATH queries could be applied. Another, potentially more powerful option is the use of the HL7 Standard object oriented query language such as GELLO.

## B.2 Representation of constructor bindings

*Constructor bindings* are used to specify the way in which a set of related *fields* contribute to a valid SNOMED CT *expression* that encapsulates the relevant context and refinements. Strictly speaking, the terminology component of a *constructor binding* is not a constraint but a blueprint. This blueprint determines how multiple data points should be transformed into a single *expression* that fits within a specified common pattern for representing a particular type of information.

A *constructor binding* can be represented using the SNOMED CT compositional grammar with the individual concept identifiers replaced by references to relevant data points in the Care Components model.

## B.3 Representation of retrieval bindings

Retrieval binding are queries and should be expressed using an appropriate query form. These queries need to be expressed using the Care Components model. A standard query language such as SQL (Structured Query Language) or XPATH (XML Path Language) is likely to be the best starting point. However, some specific extensions are required to manage specific issues such as:

- Computation of subsumption between expressions and expression predicates.
- Representation of post-coordinated expression predicates.
- Testing required for these predicates includes subsumption testing and testing of specific *concept model* attribute values.
- Temporal relationships between entries.
- Optimisation of expressing subject focused with entries.
- Managing potential duplicate entries for the same occurrence.

The requirements for representation and processing of post-coordinated expression predicates are similar to those for representing *expression constraints* (Section 5). In most clinical cases the requirements are for *semantic expression constraints* – that is constraints that specify the range of meanings that fall within scope of the predicate.

For the most part, the terminology aspects of retrieval bindings are the same as those discussed for *expression constraints* in 5. However, there are two additional requirements.

The expression predicates form part of a query, therefore the way they are represented or referenced must fit within the overall query syntax.

The expression predicates must be capable of optimised interpretation and execution against instance data in the Care Components model.

## Appendix C Representing Expression Constraints – Option Review

### C.1 Introduction

This section considers several existing approaches that have been suggested for representing *expression constraints*. These were compared with the abstract model of *expression constraints* (see 5.3) to assess their fitness for purpose.

### C.2 Extended Compositional Grammar

An extension of the SNOMED compositional grammar in which the concept identifiers can be replaced by constraints specifying concept sets

- This is used in the HL7 'Guide to the Use of SNOMED CT with HL7 Version 3' DSTU (one of the Terminology work items) and in NHS CFH work on terminology binding with *openEHR*.
- This approach is closely aligned with the general model described in 5.3. The constraint syntax matches the standard SNOMED CT expression syntax but optionally replaces concept identifiers with sets.
- Another advantage is the combination of human-readability with machine processability.
- A disadvantage is that some constraint requirements require extensions that are not self-contained in this formalism (e.g. *Refsets* and Version management) and these undermine human-readability as well as adding hidden complexity.
- A further disadvantage is that while the syntax is machine processable, it requires a specific parser rather than being processable by XML parsers or other standard tools. This disadvantage is accentuated by the added complexity of the extensions added to support representation of constraints.

### C.3 Existing Description Logic representations

Use of standard Description Logic representations such as KRSS (Knowledge Representation System Specification) and OWL (Ontology Web Language)

- This has been considered by the MRCM project. It has not been adopted as the primary representation for the MRCM but may be supported in future as an alternative export form.
- An advantage of this approach is that it allows use of standard tools.
- A disadvantage is that these forms do not meet the full set of *expression constraint* requirements without adding customisations. These customisations undermine the ability to use standard tools.



## C.4 SNOMED CT Machine Readable Concept Model

The IHTSDO Machine Readable Concept Model (MRCM) project uses an enhancement of the proposed distribution and interchange formats to represent the rules for modelling SNOMED CT components.

This approach uses the component versioning technique built into the proposed SNOMED CT Enhanced Release Format (ERF/RF2) and reuses existing SNOMED CT component structures (e.g. *Refsets*, *Concepts* and *Relationships*). It adds components to represent constraints and reusable subsidiary components that represent specific types of tests required to apply a constraint.

- An advantage of this approach is that it allows LRA terminology bindings to be directly linked into the MRCM development. This would assist the evolution of this approach and allow good alignment with IHTSDO activities.
- A possible disadvantage is the loss of the human-readability. This disadvantage could be removed using transforms that export and render relevant aspects of constraints in alternative forms.
- A possible disadvantage is that this approach does not allow direct use of standard DL tools. This disadvantage could be removed using transforms that export and render relevant aspects of constraints in alternative forms.
- SNOMED CT concept definitions do not include nested refinements of the type that occur in expressions. This is a result of the technical environment used to model, classify and distribute SNOMED CT content. The MRCM design focuses on concept definitions with a planned future extension to cover expression constraints.
- Other practical disadvantages were found when trying to apply the MRCM model directly to LRA Terminology Binding. These relate to the fact that the form of representation used in MRCM is designed to assist and validate modelling of concepts against a global or national set of constraints whereas an *expression constraint* is relevant to a particular field in a particular set of constrained classes.
  - MRCM constraints are relevant to *expression constraints* but the relevance of each *expression constraint* is limited to a constrained class and derived refinements of that class.
  - Each MRCM constraint specifies a constraint on a single attribute, cardinality or dependency. This allows effective reuse of the same constraint in various situations. In contrast, an *expression constraint* specifies a set of such rules which are applied together. It is inconvenient to have to refer to multiple separate constraints.
  - The MRCM constraints that test 'relationships' and 'cardinalities' have been well tested and are sufficiently expressive. However, feedback on MRCM 'dependency' constraints indicates these are difficult to specify and interpret. *Expression constraints* need to support some types of interdependency and following mirroring the abstract model of *expressions* facilitates this without introducing the more complex style of dependency constraint representation used in the MRCM.

## C.5 A specific XML based format aligned with the abstract model

The final option considered was a specific XML format, developed to match the identified requirements. This type of representation would have the advantage of being directly based on the abstract model described in 5.3. It would also be able to evolve to address gaps identified during development.

This approach could meet the requirements addressed by the Extended SNOMED Composition Grammar while adding the required greater expressivity including:

- Cardinality constraints
- Clearer representation of set operations.
- Explicit support for alternative interpretations (i.e. *semantic* and *literal expression constraints*)
- Inclusion of constraint identifiers, metadata and annotations.

The use of XML also offers several additional advantages.

- Use of XML schema to enable syntactic validation of constraint representation.
- Simplification of constraint parsing when testing of candidate expressions.
- Ability to use XSLT (or XQUERY)
  - To transform relevant information to and/or from other XML representations including OWL, MRCM and DITA;
  - To transform constraints to human-readable renderings (including the Extended SNOMED Composition Grammar).

This format could also be integrated with the proposed SNOMED CT Release Format 2 by using the revised proposal for representing Refsets.

A disadvantage is that this is not based on existing standards or specifications. However, use of XML would allow transforms to other standard representations.

## C.6 Evaluation of options

The extended compositional grammar is the closest match to recommended general model for *expression constraints*. However, experience with more complex constraints demonstrated that lack of easy processability and validation was a real practical issue. This was compounded by deterioration in human-readability (of one of the key strengths of the composition grammar) as constraints become more complex.

An existing DL representation such as OWL could be customised to represent most aspects of *expression constraints*. However, this customisation would not necessarily be readily applicable using standard tools. Adapting this as the initial approach in a way that was outside the standard use of OWL would be likely to add to complexity and would probably provoke justifiable criticism for 'misuse' of the standard. Nevertheless it is likely that in the future standard representations would be useful to replace or augment alternative representations.

*Expression constraints* need to be aligned with the SNOMED CT Concept model as represented by the MRCM. However, as discussed above, the MRCM model is designed to meet requirements for supporting and validating concept modelling. As a result the representational form does not directly address the requirements identified for *expression constraints*.

A customised XML representation that aligns with the abstract model and can be transformed to and from other representations is to be strong candidate. An appropriately designed XML representation would be able to fully support all the types of constraints in the Extended SNOMED Compositional Grammar (ESCG) which has been demonstrated to be useful by earlier work in the IHTSDO, HL7 and in the NHS<sup>21</sup>. It would add to this additional expressivity and greater syntactic consistency. While supporting relevant transforms to and from other representations, it would avoid overloading or customising existing standards to meet requirements that go beyond their existing scope.

## C.7 Recommended approach

Based on this evaluation, the approach chosen was to specify an XML format that captures the SNOMED CT abstract model of expressions and extends it to support constraints.

The XML representation specified is based on the recommended general model for *expression constraints* and is fully aligned with the Extended SNOMED Composition Grammar<sup>21</sup>. However, it also provides a representation that is easier to process and transform to other representations to meet specific purposes.

A key advantage of this approach is that it facilitates validation of the constraint representation. The process of converting existing expression and constraints represented from compositional grammar to the new form identified several inconsistencies and a few significant errors.

---

<sup>21</sup> The XML *expression constraint* representation specified in this document has been tested using an Extended SNOMED Compositional Grammar (ESCG) parser and a set of XSLT stylesheets. These tests have demonstrated reliable round-trip conversion of existing constraints expressed in ESCG into the XML form and back in plain text ESCG, HTML marked up ESCG and DITA marked up versions of the grammar. Where the starting representation is ESCG, this round-trip testing does not result in any loss of information. The XML representation is more expressive than ESCG. Therefore, when *expression constraints* that make use of additional features in the XML representation are transformed to ESCG, there is an inevitable loss of this specificity. In these cases, the ESCG provides a consistent but incomplete human-readable view of the constraint. In future, these transforms could be augmented with additional textual annotations to complete the human-readable information. However, reversible processing of these annotations is unlikely to be reliable. Therefore, the XML representation should be regarded as the reference form.

## Appendix D Terminology Binding - Location and Referencing

### D.1 Possible terminology binding locations

There are various possible locations for terminology bindings.

1. An *information model artefact* could directly include the *expression constraint*.
  - This approach would require each terminology binding to be expressed in full and does not allow any reuse even where this is appropriate.
  - In theory, all versioning would be managed in the information model.
    - In practice this is not true where 'intensional' definitions specify the sets as set membership may be changed by terminology updates.
2. An *information model artefact* could reference the relevant *expression constraint*.
  - This approach would allow reuse of terminology constraints.
  - Changes in referenced constraints could materially alter the information model and as a result the composite of terminology, terminology constraints and *information model artefacts* would need to be versioned.
3. An independent representation could be used to bind identified constructs in an *information model artefact* to referenced *expression constraint*.
  - This approach maximises reuse and places greater flexibility in the hands of those doing the bindings.
  - It allows experimentation that is not dependent on support for appropriate constraints or references in *information model artefacts*.

Option 1 is feasible if the *information model artefacts* are lightweight constraints against the Care Components model. An integrated design would enable an integrated representation. However, subsequent refinement and maintenance of terminology bindings (e.g. to accommodate terminology updates) may be more difficult if the terminology bindings are scattered throughout a large number of *information model artefacts*. This approach may also result in issues with compatibility and/or representational efficiency due to interaction between a rich terminology constraint syntax and an established modelling format (e.g. XMI or ADL).

Option 2 is feasible and only requires that the information model has a way to refer to externally represented constraints. The detailed representation of the terminology binding can then be maintained separately, although it remains closely bound to the relevant artefacts. This approach also supports appropriate reuse of the same terminology binding in derived *information model artefacts*.

Option 3 is well-suited to experimental bindings as the only requirement it imposes on the information modelling environment is that it should be possible to uniquely reference each *information model artefact*<sup>22</sup>. However, in an operational environment it is probably preferable for the terminology binding information to be more closely associated with the relevant *information model artefacts*.

---

<sup>22</sup> Whichever approach is followed, reference to *information model artefacts* is required to support *constructor bindings*, *composite semantic constraints* and *retrieval bindings*.

## D.2 Recommended form for referencing terminology bindings

The recommended approach to authoritative representation of terminology bindings, for the purposes of authoring and maintenance, is to include these *by reference* from *information model artefacts* (Option 2).

Within the *information model representation* each instance of a *expression constraint* shall be represented by a *terminology constraint reference*. Each reference shall identify a version managed terminology *expression constraint* within a shared repository<sup>23</sup>.

This approach has the following advantages:

- It allows reuse of terminology constraints in multiple *information model artefacts*.
  - For example, a constraint that requires an entry related to a kidney operation must specify laterality, could be applied in several different constrained models - related to scheduling and carrying out kidney operations.
- It reduces the terminology binding development and maintenance burden.
  - For example, changes such as those required to accommodate the merger of the 'onset' and 'course' attributes into the combined 'clinical course' attribute would be applied by updating any *constraint expressions* that refer to these attributes. The changes would not need to be repeated for each clinical finding related *information model artefact* (Note: This *concept model* revision was made during 2007-2008).
- It does not impose artificial restrictions or interdependencies on the information model or terminology constraint formalism.
  - For example, it is possible to embed a *terminology expression constraint* syntax directly within XMI. However, this requires the use of XMI extension properties. In addition, the relevant XMI property is represented as an XML attribute; thus, if the *terminology constraint* is also expressed in XML, it becomes obfuscated by entity sequences. While not an absolute barrier, this complicates manual validation and debugging of specifications which may be required during development.
- It allows the same terminology constraints to be referenced in different model formalisms.
  - For example, the same *terminology constraint* can be referenced from an XMI representation of a UML model, from an EN13606 archetype, an HL7 RMIM or template.

---

<sup>23</sup> . Logically the information model and terminology constraint repository must be managed as a single resource of interdependent components. The assumption is that the common repository of terminology bindings will be hosted in the same environment with common version management. The use of referencing is recommended for reasons of syntactic clarity and authoring efficiency. It does not imply independence or a division of responsibility.

- It allows the same terminology constraints to be referenced in query languages, used to support retrieval (see B.3) and content validation.
  - For example, the reason for requiring allergies to be recorded in a manner restricted by terminology constraints is to ensure they can be reliably retrieved for display and decision support. It makes sense if the expressions used to represent the constraint, can also be applied in a retrieval query.
- Terminology bindings inherently require reference to terminology resources. Therefore, even if a constraint is fully represented within an information model artefact, it cannot be interpreted as a standalone resource.
  - For example, even if a constraint requiring an *expression* to be subsumed by 387713003|surgical procedure| is expressed in full, it can only be tested by reference to the released content of SNOMED CT.
- Some terminology bindings are inherited from the SNOMED CT *concept model* and should not need to be restated in each information model artefact.
  - For example, a constraint to prohibit use of 'finding site' to refine a 'procedure' should not need to be restated as it is not permitted by the *concept model*.

### D.3 Dereferencing representations for optimisation

References are recommended as the primary authoritative form for authoring and maintenance. However, a dereferenced form (Option 1) may have specific benefits for implementation optimisation. Therefore, if the information modelling formalism allows this, an alternative dereferenced form, that includes *expression constraint* in-situ, may also be specified.

This offers the following specific benefits:

- This would allow *terminology constraint references* to be expanded to an exhaustive literal representation of constraints should this be required for a particular implementation.
- It would also allow migration to integrated representation in future if practical experience suggests that such a move would be beneficial.

### D.4 Other approaches

Option 3 which uses an independent resource to bind *terminology components* to *information model artefacts* is not recommended as a means of representing constraints. The use of an independent layer between the two formalisms adds an additional degree of freedom and, while this increases flexibility, it does so at the expense of adding to the complexity of configuration management.

While this is deprecated for representation of *expression constraints*, it may be useful for associating other types of *terminology binding* with *information model artefacts*.

## Appendix E Extended Compositional Grammar

This annex specifies extensions to the SNOMED CT Compositional Grammar which have been used in HL7 Terminology and in early NHS work to represent constraints on expressions. This material, previously published in the NHS CFH document on 'Terminology Binding Requirements and Principles' (version 1.0 May 2008) is repeated here for reference.

Although this document recommends a new XML representation of *expression constraints* (see Section 5), the Extended SNOMED Compositional Grammar (ESCG) is used as the basis for human-readable rendering of constraints.

Table 16 provides an overview of the SNOMED CT Compositional Grammar which is documented in 'SNOMED CT Guide to Abstract Models and Representational Forms'. The 'Extended Compositional Grammar' enhances the SNOMED CT Compositional Grammar in the following ways:

To improve the clarity and processability of references to SNOMED CT concepts and expressions within blocks of narrative text:

- To enable simple representation of constrained value-sets of concepts and expressions based on post-coordinated refinement.
- To support clear documentation of relatively simple constraints, an informal extension has been made to the compositional grammar.

An informal extension has been made to the compositional grammar to represent constraints. This extension includes:

- Additional symbols (specified in Table 17) to represent different types of constraint.
- Logical 'AND' and 'OR' operations (see Table 19)
- To enable more effective representation of value-set constraints, the use of subset (or *Refset*) identifiers is permitted in place of concept identifiers when specifying constraints. Subset/Refset identifiers are prefixed with the caret ^ symbol.

This human-readable rendering of constraints arose from changes to the grammar proposed in the 'Guide to Use of SNOMED CT in HL7 Version 3' and the NHS CFH report on 'Design of Adverse Reaction Archetypes and Templates for the Vaccination Summary Record'; the pilot project on use of EN13606 and SNOMED CT.

**Table 16. Summary of SNOMED CT Compositional Grammar**

Symbol	Notes	Examples
digits	ConceptId	<p>A sequence of digits in an expression represents a SNOMED CT concept identifier. The two exceptions to this are:</p> <p>1) Where digits occur between a pair of pipe symbols, in which case the digits are part of the display name (see   <i>text</i>   row in this table).</p> <p>2) Where a string of digits is immediately preceded by a caret symbol ^. In this represents a subset (or <i>Refset</i>) identifier (see Table 17)</p> <p>The simplest expression is a concept identifier on its own. For example: 87628006</p>
<i>text</i>	Display name delimiter	<p>A pair of pipe   symbols is used to delimit an optional display name for the immediately preceding concept identifier. For example: 87628006   <i>bacterial infectious disease</i>  </p> <p>The display name may be the term string of any of the descriptions associated with the concept in a current version of SNOMED CT<sup>24</sup>. For example, any the following are a sample of valid representations of the same concept:</p> <p>87628006   <i>bacterial infectious disease (disorder)</i>    87628006   <i>disease caused by bacteria</i>    87628006   <i>enfermedad infecciosa bacteriana</i>    87628006   <i>maladie infectieuse bactérienne</i>  </p>
space tab linefeed return	Whitespace characters	<p>Whitespace characters are ignored and can thus be used to format the appearance of an expression where this aids clarity. The only exception to this rule is that spaces are not ignored within a display name.</p> <p>Note: Spaces before or after the last non whitespace character of a display name are ignored. The text between the pair of pipe characters is trimmed of any surrounding whitespace, but spaces within the enclosed text are treated as part of the display name.</p>
:	Refinement prefix	<p>A colon : precedes a refinement of meaning of the concept to the left of the colon. A refinement consists of one or more attributes and/or attributes groups and these are illustrated by examples in subsequent rows of this table.</p>
=	Attribute value prefix	<p>Each of the attributes that make up a refinement consists of an attribute name and an attribute value. The attribute name precedes the value and is separated from it by an equals sign = .</p> <p>The attribute name is represented by a concept identifier and the attribute value. The attribute value may be represented by a concept identifier as in the following example or by a nested expression (see example later in this table).</p> <p>The following example specifies a bacterial infectious disease caused by streptococcus pneumoniae.</p> <p>87628006   <i>bacterial infectious disease</i>   :  246075003   <i>causative agent</i>   = 9861002   <i>streptococcus pneumoniae</i>  </p>

<sup>24</sup> In constraint expressions where a *Refset* identifier is used the | *text* | is the name of the *Refset*.



Symbol	Notes	Examples
,	Attribute separator	<p>A refinement may include more than one attribute. In this case, a comma , is used to separate attributes from one another.</p> <p>The following example specifies a bacterial infectious disease affecting the lung and caused by streptococcus pneumoniae.</p> <p>87628006   bacterial infectious disease   :  246075003   causative agent   = 9861002   streptococcus pneumoniae    , 363698007   finding site   = 45653009   structure of upper lobe of lung  </p>
( exp )	Nested expression	<p>The value of an attribute may be represented by a nested expression rather than a single concept identifier. In this case, the nested expression is enclosed in parentheses ( ).</p> <p>The following example specifies a bacterial infectious disease affecting the left upper lobe of the lung and caused by streptococcus pneumoniae. The nested expression localises and lateralises the site of the disease.</p> <p>87628006   bacterial infectious disease   :  246075003   causative agent   = 9861002   streptococcus pneumoniae    , 363698007   finding site   =( 45653009   structure of upper lobe of lung   :  272741003   laterality   = 7771000   left   )</p>
{ grp }	Attribute group	<p>In some cases, different sets of attributes apply to different facets of the same concept. For example, some common fractures involve two adjacent bones and the nature of the fracture of each bone may differ. Similarly, some procedures involve removal of one structure and repair of another and different refinements of these actions may be required.</p> <p>In SNOMED CT, concepts that have multiple facets are defined with each facet represented by a separate relationship group. When these concepts are refined, it may be necessary to specify which group is being refined. In these cases, curly braces { } are used to group together sets of attributes that act together.</p> <p>The following example represents a fracture of the shaft of the tibia and fibula. The tibia has a spiral fracture while the nature of the fracture of the fibula is incomplete.</p> <p>271577005   fracture of shaft of tibia and fibula   :  { 116676008   associated morphology   = 30543000   fracture, incomplete    , 363698007   finding site   = 113224005   bone structure of shaft of fibula   }  ,{ 116676008   associated morphology   = 73737008   fracture, spiral    , 363698007   finding site   = 52687003   bone structure of shaft of tibia   }</p>
+	Combination	<p>87628006   bacterial infectious disease   + 50043002   disorder of respiratory system  </p> <p>This means a disorder that is both a bacterial disease and disorder of the respiratory systems. For example "bacterial pneumonia".</p> <p>It does <b>not</b> mean two separate disorders that for some reasons are being linked. For example, this use of the plus sign would <b>not</b> be the appropriate way to represent that someone has a non-bacterial respiratory disorder (e.g. allergic asthma) and also has a bacterial disease (e.g. impetigo).</p>

**Table 17. Compositional Grammar extension - Constraint symbols**

Symbol	Notes	Examples
	This concept (No symbol prefix)	71388002   procedure    The concept "procedure" SHALL be used. Note: By default, unless the surrounding context states otherwise, this implies this precise concept (i.e. not one of its subtypes). However, the context within a sentence or parsable expression may imply a less specific requirement. For example, if the concept is followed by any options for addition of refinements, these implicitly permit refinement of the concept.
^	The identifier (and optional text) that follows refers to a <i>Refset</i> and means that any member of that set it permitted.	^ 8181000000134   Encounter disposition    A concept that is a member of the 'Encounter disposition' subset SHALL be used.  Note: This symbol cannot be combined with the "<<" or "<" symbols. This could conflict with the membership definition for the set. Where subtypes of members of a set are intended to be included, this must be a property of the set (or of its members) and the set must be specified using and 'intension definition'.
<<	This concept or any subtype permitted	<< 71388002   procedure    Either the concept "procedure" or one of its subtypes SHALL be used.  Note: this differs from the "<=" symbol used to indicate the same constraint in other HL7 specifications. The reason for the difference is to limit the use of "=" as the operator that joins an attribute name and an attribute value in the unextended compositional grammar
<	Any subtype of this concept (but not the concept itself)	71388002   procedure   : 363704007   procedure site   =( 29836001   hip region structure   : 272741003   laterality   = < 182353008   side   )  The procedure site SHALL be the value "hip region structure" and SHALL include the attribute "laterality"  The value of "laterality" SHALL be a subtype of "side" but SHALL NOT be "side" itself.
~	Optional attribute (only applicable as a prefix to AttributeName)	71388002   procedure   : << 363704007   procedure site   = ( << 29836001   hip region structure   : ~ 272741003   laterality   = < 182353008   side   )  The attribute "procedure site" or one of its subtypes (e.g. "procedure site - direct") SHALL be applied and its value SHALL be "hip region structure" or one of its subtypes.  The attribute "laterality" MAY BE applied and if present its value SHALL be a subtype of "side" but SHALL NOT be "side" itself.

Symbol	Notes	Examples
!	This concept is prohibited and SHALL NOT be used.	<p>71388002   procedure    : 363704007   procedure site   =( 29836001   hip region structure    : ! 272741003   laterality   )</p> <p>The procedure site SHALL be the value "hip region structure" and SHALL NOT include the attribute "laterality".</p> <p>Note: This example conflicts with the SNOMED CT compositional grammar as no value is supplied for the laterality attribute. However, in a constraint, if an attribute is not permitted, there is no need to provide a value. The most general value permitted by the concept model for this attribute (e.g. &lt;&lt; 182353008   side   ) could be specified, but this would decrease rather than enhance clarity.</p>
! <	This concept and all its subtypes are prohibited and SHALL NOT be used.	<p>71388002   procedure   :  363704007   procedure site   =( 29836001   hip region structure   :  272741003   laterality   = ! &lt; 66459002   unilateral   )</p> <p>The procedure site SHALL be the value "hip region structure" and MAY include the attribute "laterality"</p> <p>The value of "laterality" SHALL NOT be "unilateral" or a subtype of "unilateral".</p>
! ^	The identifier (and optional text) that follows refers to a <i>Refset</i> and means that NO member of that set is permitted.	<p>! ^ 10301000003139   Clinical Exclusions  </p> <p>A concept that is a member of the 'Clinical exclusions' subset SHALL NOT be used.</p>

**Table 18. Compositional Grammar Extension - Constraining elements**

Element	Notes and examples
<b>ConceptId</b>	<p>A constraint symbol MAY directly precede a ConceptId. In this case, it requires, allows, or prohibits use of the referenced concept (and/or subtypes of that concept) in that logical position in the expression.</p> <p>Unless otherwise stated, the comparison between an instance expression and a constraint assumes both are transformed to normal forms before testing.</p> <p>The definitions of concepts 38102005   <a href="#">cholecystectomy</a>   and 80146002   <a href="#">appendectomy</a>   both include:</p> <ul style="list-style-type: none"> <li>subtypes of 71388002   <a href="#">procedure</a>  </li> <li>with 260686004   <a href="#">method</a>   = 129304002   <a href="#">excision - action</a>  .</li> </ul> <p>Therefore, the following constraint:</p> <p>71388002   <a href="#">procedure</a>   :</p> <p>260686004   <a href="#">method</a>   = &lt;&lt; 129304002   <a href="#">excision - action</a>  </p> <p>Permits expressions such as 38102005   <a href="#">cholecystectomy</a>   or 80146002   <a href="#">appendectomy</a>  </p>
<b>Attribute Name</b>	<p>A constraint symbol MAY directly precede the ConceptId that specifies the name of an attribute. In this case it requires, allows or prohibits use of that attribute (or a subtype of that attribute). Unless the use of the attribute is prohibited, the value of that attribute MAY be separately constrained.</p> <p>The following example asserts that the attribute "procedure site" or one of its subtypes (e.g. "procedure site - direct") SHALL be applied and its value SHALL be "hip region structure" or one of its subtypes.</p> <p>71388002   <a href="#">procedure</a>   :</p> <p>&lt;&lt; 363704007   <a href="#">procedure site</a>   = &lt;&lt; 29836001   <a href="#">hip region structure</a>  </p>
<b>Nested Expression</b>	<p>A constraint symbol may directly precede an expression enclosed in parentheses. In this case, it requires, allows or prohibits inclusion of the parenthesised expression (and/or subtypes of that expression) in that logical position in the expression.</p> <p>Note: It is generally clearer to specify the individual constraints on the elements within the nested expression rather than to apply a constraint to the nested expression as a whole.</p>
<b>Attribute Group</b>	<p>A constraint symbol MAY directly precede an attribute group. In this case, it requires, allows or prohibits inclusion of the specified group (and/or subtypes of that group) in that logical position in the expression.</p> <p>The following example asserts that the group shown or a subtype of that group must be present. Thus the following constraint includes any abdominal excision.</p> <p>71388002   <a href="#">procedure</a>   :</p> <p>&lt;&lt; 260686004   <a href="#">method</a>   = 129304002   <a href="#">excision - action</a>  </p> <p>, 405813007   <a href="#">procedure site - Direct</a>   = 113345001   <a href="#">abdominal structure</a>  </p>
<b>Other</b>	<p>The constraints cannot be used elsewhere in the expression. In particular a constraint cannot be applied to a refinement as a whole or to a display name. Therefore, the constraint symbols cannot immediately follow the concept identifier, nor can they precede the pipe (" ") or colon (":") symbols.</p>

**Table 19. Grammar Extension - Logical constrain combinations**

Symbol	Notes	Examples
<b>OR</b>	Where two or more values are permitted the set of conditions and the individual expressions SHALL both be enclosed in standard curved brackets () and the word "OR" SHALL be placed between the expression.	<p>71388002   procedure   :</p> <p>363704007   procedure site   = ( 29836001   hip region structure   :</p> <p>~ 272741003   laterality   =</p> <p>( 7771000   left   )</p> <p>OR ( 24028007   right   ) )</p> <p>The procedure site SHALL be the value "hip region structure" and MAY include the attribute "laterality"</p> <p>The value of "laterality" SHALL be either "left" or "right".</p>
<b>AND</b>	Where two or more conditions are both required to apply the individual expression SHALL be enclosed in standard curved brackets and the word "AND" shall be placed between the expressions. ((exp1) AND (exp2))	<p>71388002   procedure   :</p> <p>363704007   procedure site   = ( 29836001   hip region structure   :</p> <p>~ 272741003   laterality   =</p> <p>( &lt; 182353008   side   )</p> <p>OR ( ! &lt;&lt; 51440002   bilateral   ) )</p> <p>The procedure site SHALL be the value "hip region structure" and MAY include the attribute "laterality"</p> <p>The value of "laterality" SHALL be a subtype of "side" AND SHALL NOT be "bilateral" or a subtype of "unilateral".</p>

## Appendix F Care Components Model

Figure 7 shows the derived classes in the Care Components model that are referred to in this document. This diagram is based on NPFIT-FNT-TO-DPM-0935.01 WORKING DRAFT / 0.3.2 dated 2009-06-02. This model is subject to revision and is included here only as a point reference for the current version of this document.

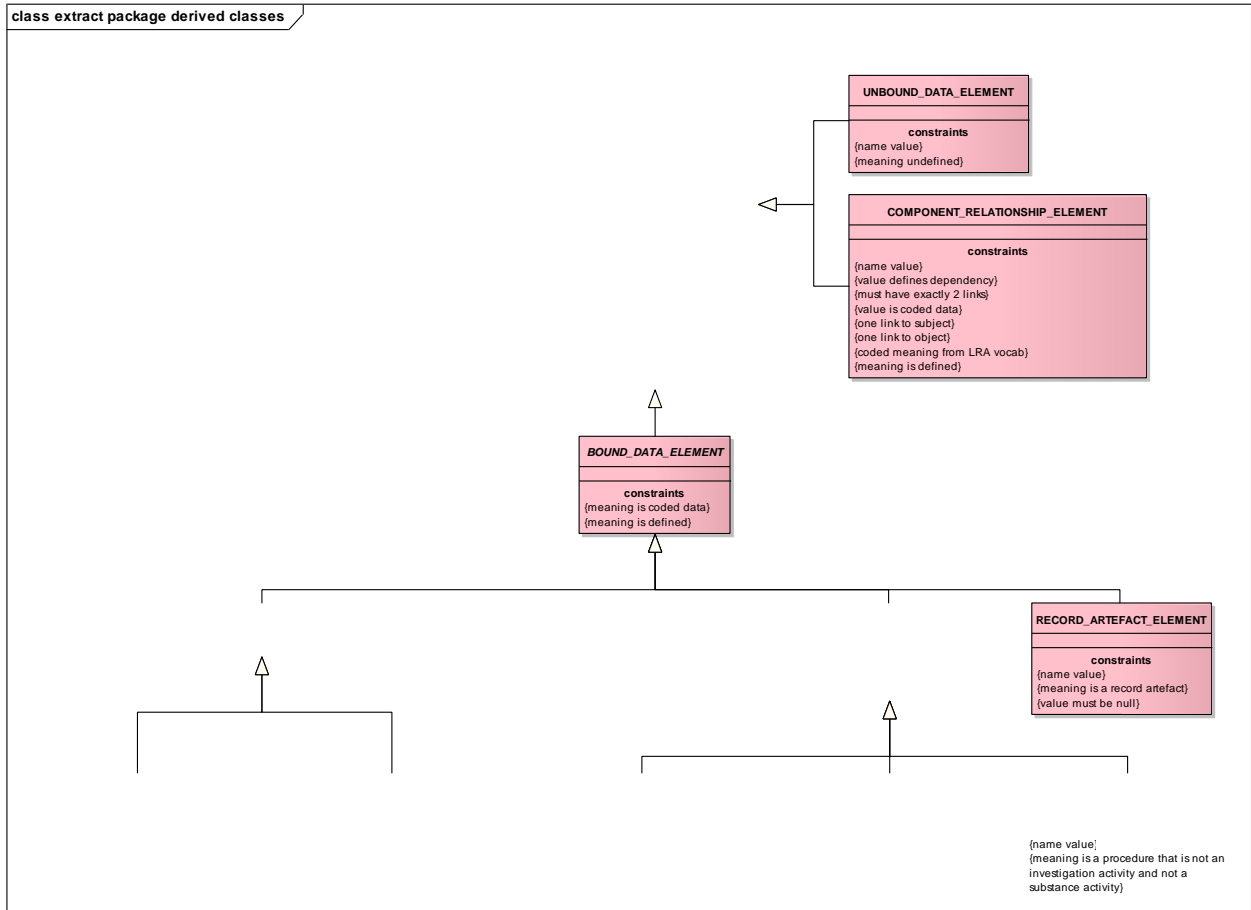


Figure 7. Care Components derived classes