Date & Time
20:00 UTC Wednesday 6th November 2019

Location
Zoom meeting: https://snomed.zoom.us/j/471420169

Goals
- To progress work on
  - URIs
  - Templates
  - ECL
  - Query language

Attendees
- Chair: Linda Bird
- Project Group: Brian Carlsen, Daniel Karlsson, Anne Randorff Højjen, Peter G. Williams, Michael Lawley, Ed Cheetham, Kai Kewley

Agenda and Meeting Notes

<table>
<thead>
<tr>
<th>Description</th>
<th>Owner</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Welcome and agenda</td>
<td>Linda Bird</td>
<td>• Boolean to be added to SCG, ECL, STS and ETL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SCG &amp; ECL - See draft syntax</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- STS &amp; ETL - Use abbreviation &quot;bool&quot;</td>
</tr>
<tr>
<td>URIs</td>
<td>Peter G.</td>
<td>• Summary of discussions in Malaysia</td>
</tr>
<tr>
<td></td>
<td>Williams</td>
<td>- Published version: <a href="https://confluence.ihtsdotools.org/display/DOCURI">https://confluence.ihtsdotools.org/display/DOCURI</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Work in progress: <a href="https://confluence.ihtsdotools.org/display/WIPURI/URI+Standard">https://confluence.ihtsdotools.org/display/WIPURI/URI+Standard</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Project group was asked to review WIP updates to URI proposal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Consistent structure and examples added to each URI page</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• New proposal on URIs for languages and language instances</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• New proposal on URIs for modelling resources e.g. valueSet, fhirStructureDefinition, openEhrArchetype</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2.8 URIs for Modelling Resources</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Agreed format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <a href="http://snomed.info/valueSet/gps">http://snomed.info/valueSet/gps</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <a href="http://snomed.info/fhirStructureDefinition/conditionWithSnomedBinding">http://snomed.info/fhirStructureDefinition/conditionWithSnomedBinding</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <a href="http://snomed.info/openEhrArchetype/conditionWithSnomedBinding">http://snomed.info/openEhrArchetype/conditionWithSnomedBinding</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <a href="http://snomed.info/">http://snomed.info/</a>&lt;resourcetype&gt;/&lt;resourcename&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ResourceTypes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• valueSet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• fhirStructureDefinition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• openEhrArchetype</td>
</tr>
</tbody>
</table>
### Expression Templates

<table>
<thead>
<tr>
<th>Peter G. Williams</th>
</tr>
</thead>
</table>

*Summary of discussions in Malaysia*

- Published version: [https://confluence.ihtsdotools.org/display/DOCSTS/Template+Syntax+Specification](https://confluence.ihtsdotools.org/display/DOCSTS/Template+Syntax+Specification)
- Work in progress: [https://confluence.ihtsdotools.org/display/WIPSTS/Template+Syntax+Specification](https://confluence.ihtsdotools.org/display/WIPSTS/Template+Syntax+Specification)

- Added a 'default' constraint to each replacement slot - e.g. default (72673000 (Bone structure (body structure)))
- Enabling 'slot references' to be used within the value constraint of a replacement slot - e.g. [[ +id (<< 123037004 (Body structure) MINUS << $findingSite2) @findingSite1]] =SELF]
- Adding 'sameValue' and 'allOrNone' constraints to information slots - e.g. sameValue ($site), allOrNone ($occurrence)
- Self

*See changes in red here: 5.1. Normative Specification*

*Recap of discussions in Malaysia regarding exchange of templates*

- ReferencedComponentId
- Name
- Version
- Domain of template - ECL where template should be applied
- Template itself - ETL
- Additional rules
- Mandatory / optional

**Examples:**

```plaintext
[[+id]: [1..*] @my_group sameValue(morphology)]
[[Finding site] = [[+id (<< 123037004 (Body structure) MINUS << $site[! SELF ] ) @site ] , [Associated morphology] = [[+id @my_morphology]]]
```

- Implementation feedback on draft updates to Expression Template Language syntax
  - Use cases from the Quality Improvement Project:
    - Multiple instances of the same role group, with some attributes the same and others different. Eg same morphology, potentially different finding sites.

Note that QI Project is coming from a radically different use case. Instead of filling template slots, we're looking at existing content and asking *exactly how does this concept fail to comply to this template?*

For discussion:

```plaintext
[[0..1] ( [[0..1]] 246075003 (Causative agent) = [[+id (< 410607006 (Organism) ) @Organism]] )
```

Is it correct to say either one of the cardinality blocks is redundant? What are the implications of 1..1 on either side? This is less obvious for the self grouped case.

### Road Forward for SI

1. Generate the parser from the ABNF and implement in the Template Service
2. User Interface to a) allow users to specify template at runtime b) tabular (auto-completion) lookup STL
3. Template Service to allow multiple templates to be specified for alignment check (aligns to none-off)
4. Output must clearly indicate exactly what feature of concept caused misalignment, and what condition was not met.

Additional note: QI project is no longer working in subhierarchies. Every 'set' of concepts is selected via ECL. In fact most reports should now move to this way of working since a subhierarchy is the trivial case. For a given template, we additionally specify the “domain” to which it should be applied via ECL. This is much more specific than using the focus concept which is usually the PPP eg Disease.

FYI Michael Chu

### Description Templates

<table>
<thead>
<tr>
<th>Kai Kewley</th>
</tr>
</thead>
</table>

*Summary of discussions in Malaysia*

- Overview of current use
- Review of General rules for generating descriptions
  - Removing tags, words
  - Conditional removal of words
  - Automatic case significance
  - Generating PTs from target PTs
  - Reordering terms
- Mechanism for sharing general rules - inheritance? include?
- Description Templates for translation
- Status of planned specification

### Expression Constraint Language

<table>
<thead>
<tr>
<th>Linda Bird</th>
</tr>
</thead>
</table>

*Agreement in Malaysia - ECL will add the following term searching syntax (no regex - just wild card and word prefix any order):*

```plaintext
```
Term Search Type

a. Wild Card Match (collation) - e.g.
   • { term = wild("*heart*") }
   • { term = wild(sv,"*hjärta*") }

a. Word Prefix Any Order - e.g.
   • { term = match("hear att") }

a. Default (word prefix any order) - e.g.
   • { term = "hear att" }
   • { term = "*heart*" }

Potential Examples

• << 64572001 [Disease] { term = "heart" }
• << 64572001 [Disease] { term = "heart", languageCode = "en" } AND 64572001 [Disease] { term = "hjärta", languageCode = "sv" }
• << 64572001 [Disease] { term = "heart", languageCode = "en" } OR << 64572001 [Disease] { term = "hjärta", languageCode = "sv" }

• << 64572001 [Disease] { term = "heart", languageCode = "en" } OR (term = "hjärta", languageCode = "sv")
• << 64572001 [Disease] { term = "heart", languageCode = "en" } OR (term = "hjärta", languageCode = "sv")
• << 64572001 [Disease] { term = "heart", languageCode = "en" } OR (term = "hjärta", languageCode = "sv")
• << 64572001 [Disease] { term = "heart", languageCode = "en" } OR (term = "hjärta", languageCode = "sv")
• << 64572001 [Disease] { term = "heart", languageCode = "en" } OR (term = "hjärta", languageCode = "sv")

Recommendation to be made on (based on investigation of grammar):
• << 64572001 [Disease] { term = "heart", languageCode = "en" } AND { term = "hjärta", languageCode = "sv" }
• << 64572001 [Disease] { term = "heart", languageCode = "en" } OR { term = "hjärta", languageCode = "sv" }
• << 64572001 [Disease] { term = "heart", languageCode = "en" } MINUS { term = "hjärta", languageCode = "sv" }

Use Cases

• Intentionally define a reference set for chronic disease. Starting point was ECL with modelling; This misses concepts modelled using the pattern you would expect. So important in building out that reference set.
• Authors quality assuring names of concepts
• Checking translations, retranslating. Queries for a concept that has one word in Swedish, another word in English
• AU use case would have at most 3 or 4 words in match
• Consistency of implementation in different terminology services
• Authoring use cases currently supported by description templates
• A set of the "ectomy"s and "itis"s

Questions

• Do we include 'typeld' - e.g. << 64572001 [Disease] { D.term = "heart", typeld = 90000000000013009 [Synonym] } 
  • NO
• Do we include 'type' - e.g. << 64572001 [Disease] { D.term = "heart", D.type = synonym ] 
  • NO
• Do we include 'languageCode' - e.g. << 64572001 [Disease] { D.term = "heart", D.type = synonym, D.languageCode = "en" } 
  • YES
• Do we include 'caseSignificanceId' - e.g. << 64572001 [Disease] { D.term = "Heart", D.caseSignificanceId = 90000000000017005 [case sensitive] } 
  • NO
• Do we include 'caseSignificance' - e.g. << 64572001 [Disease] { D.term = "Heart", D.caseSignificance = sensitive ] 
  • NO
• Do we include 'language' and 'version' - e.g. << 64572001 [Disease] { D.term = "heart" ] VERSION = http://..., LANGUAGE = (9990018810000000108[Gastro LRS], [GB English]) 
  • NO
• Do we include syntactic sugar - e.g.
  • << 64572001 [Disease] { [preferredTerm = "heart", languageRefSet = en-gb] } 
  • << 64572001 [Disease] { [fullySpecifiedTerm = "heart", languageRefSet=en-gb] } 
  • << 64572001 [Disease] { [acceptableTerm = "heart", languageRefSet=en-gb] } 
  • << 64572001 [Disease] { [preferredTerm = "heart"] FROM version = X, language = Y } 
  • NO
• Do we use/require the "D" at the start of "term"? 
  • NO

Packaging - How do we package this extension to ECL 
  • A new version of ECL - version 1.5
Recap discussions in Malaysia regarding querying historical patient records - e.g. Find all patients with a respiratory disease in the last 10 years. Do we include patients whose records contain [Recurrent chest infection]? (an inactive concept)

Solutions suggested include:

1. **Multiple queries** - Either run multiple queries against successive releases of SNOMED CT and collate results OR look at the latest snapshot before the given concept was inactivated.
2. **Use historical associations** - Either create map from inactive to active concepts, OR update EHR to replace inactive concepts with active ones.
3. **Enhanced transitive closure table** - Create an ETCT containing inactive concepts as their last known position.
4. **Augmented solution** - Check the position of each replacement to determine concepts inactivated due to wrong placement.

Proposed syntax to support execution of maps (Outstanding question: ECL or Query Language? Scope and packaging needs further discussion):

- Example use cases
  - Mapping from international substance concepts to AMT substance concepts
  - Anatomy structure and part association reference set - e.g. find the anatomical parts of a given structure
  - Checking replacement concept for inactive concept in record
- Potential syntax to consider
  - **Functional** (Note: Would need a different function for each refset attribute, so not so extensible)
    - `mapTarget ([Anatomy structure and part association refset], << [Upper abdomen structure])`
      - Return the map targets from the given map refset, where the referencedComponent matches the condition.
    - `mapSource ([Anatomy structure and part association refset], << [Liver part])`
      - Return the referencedComponent from the given map refset, where the targetId matches the condition.
  - **Dot notation + Attribute refinement** (Note: Could be added to ECL)
    - `( [Anatomy structure and part association refset] . [targetComponentId] )` (Same as `^ [Anatomy structure and part association refset]`
    - `( [Anatomy structure and part association refset] . [referencedComponent] )` (Same as `^ [Anatomy structure and part association refset]`
  - **Dot notation + Filters** (Note: May be more appropriate in extended query language)
    - `( [Anatomy structure and part association refset] { [referencedComponent] = << [Upper abdomen structure] }) . [targetComponentId] )` (Same as `^ [Anatomy structure and part association refset]`
    - `( [Anatomy structure and part association refset] { [targetComponentId] = << [Upper abdomen structure] }) . [referencedComponent] )` (Same as `^ [Anatomy structure and part association refset]`
  - **Specify value to be returned** (Note: This solution enables the language to return non-concept values)
    - `? 449608002 |Referenced component|?
      - `734139008 [Anatomy structure and part association refset]`
    - `? 449608002 |Referenced component|?
      - `734139008 [Anatomy structure and part association refset]
      - `734139008 [Anatomy structure and part association refset]`
      - `734139008 [Anatomy structure and part association refset]`

- **Reverse memberOf function**
  - What refsets is a concept a member of?
Returning attributes

<table>
<thead>
<tr>
<th>Returning attributes</th>
<th>Michael Lawley</th>
<th>Proposal from Michael:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• Currently ECL expressions can match (return) concepts that are either the source or the target of a relationship triple (target is accessed via the 'reverse' notation or 'dot notation', but not the relationship type (ie attribute name) itself.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example, I can write:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• `&lt; 404684003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• `&lt; 404684003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>But I can’t get all the attribute names that are used by `&lt; 404684003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Perhaps something like:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• `? R.type ? (&lt;&lt; 404684003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• This could be extended to, for example, return different values - e.g.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• `?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query Language - Summary from previous meetings</th>
<th>Linda Bird</th>
<th>Examples: version and dialect</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• `&lt; 64572001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• `&lt; 64572001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• `&lt; 64572001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• `&lt; 64572001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• `&lt; 64572001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• `([^ { term = &quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• `([^ { term = &quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• `X MINUS Y WHERE X = <em>, Y = (</em> ([{ term = &quot;</td>
</tr>
</tbody>
</table>

Notes

• Allow nested where, version, language
• Scope of variables is inner query

Examples: where

• `X MINUS >! X WHERE X = (< 1234 : 5678 = << 6547)` |
• `X MINUS >! X WHERE X = (< 1234 : 5678 = << 6547) VERSION http://snomed.info/sct /900000000000207008/version/20180131` |
• `X MINUS >! Y WHERE X = (< 1234 : 5678 = << 6547), Y = (< 1456) VERSION http://snomed.info/sct /900000000000207008/version/20180131` |
• `X MINUS >! X WHERE X = (< 1234 : 5678 = << 6547) VERSION http://snomed.info/sct /900000000000207008/version/20180131, LANGUAGE 900000000000508004 |GB English|` |
• `X MINUS >! X WHERE X = (< 1234 : 5678 = << 6547) VERSION http://snomed.info/sct /900000000000207008/version/20180131, LANGUAGE 99901881000000108 |GB clinical extension LRS|, 900000000000508004 |GB English|` |
• `X minus >! X WHERE X = ( < M WHERE M = (< 1234)) VERSION http://snomed.info/sct /900000000000207008/version/20180131, DIALECT 99900188100000108 |GB clinical extension LRS|, 900000000000508004 |GB English|` |

Notes

• Allow nested variable definitions, but recommend that people don’t due to readability
• Scope of variables is the inner query
• No recursion e.g. `X WHERE X = 1234 MINUS X`
  • ie can’t use a variable in its own definition
  • ie X is only known on the left of the corresponding WHERE, and not on the right of the WHERE
Keywords for Term-based searching:

- **D.term**
  - D.term = "*heart"
  - D.term = wild:"*heart"
  - D.term = regex:"*heart"*
  - D.term = match:"hear att"
  - D.term = (sv) wild: "*heart"

- **D.languageCode**
  - D.languageCode = "en"
  - D.languageCode = "es"

- **D.caseSignificance**
  - D.caseSignificance = "insensitive"
  - D.caseSignificance = "sensitive"
  - D.caseSignificance = "initialCharInsensitive"

- **D.type**
  - D.type = "FSN"
  - D.type = "fullySpecifiedName"
  - D.type = "synonym"
  - D.type = "textDefinition"

- **D.acceptability**
  - D.acceptability = "acceptable"
  - D.acceptability = "preferred"

Additional Syntactic Sugar

- **FSN**
  - FSN = "*heart"
  - D.term = "*heart", D.type = "FSN"
  - D.term = "*heart", D.type = "fullySpecifiedName"

- **synonym**
  - synonym = "*heart"
  - D.term = "*heart", D.type = "synonym"

- **synonymOrFSN**
  - synonymOrFSN = "*heart"
  - D.term = "*heart", (D.type = "synonym" OR D.type = "fullySpecifiedName")

- **textDefinition**
  - textDefinition = "*heart"
  - D.term = "*heart", D.type = "definition"

- **Unacceptable Terms**
  - (D.term = "*heart") MINUS (D.term = "*heart", D.acceptability = "LANGUAGEX")
Language preferences using multiple language reference sets

- LRSs that use the same Language tend to use 'Addition' - i.e. child LRS only includes additional acceptable terms, but can override the preferred term
  - E.g. Regional LRS that adds local dialect to a National LRS
  - E.g. Specialty-specific LRS
  - E.g. Irish LRS that adds local preferences to the en-GB LRS
- LRSs that define a translation to a different language tend to use 'Replacement' - i.e. child LRS replaces set of acceptable and preferred terms for any associated concept
  - E.g. Danish LRS that does a partial translation of the International Release
    - 999999 [Danish language reference set] ELSE [GB English reference set]

- Discuss and plan next steps

Next steps

<table>
<thead>
<tr>
<th>Action</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm next meeting date /time</td>
<td>Linda</td>
</tr>
</tbody>
</table>

Next steps

Linda Bird

Discuss and plan next steps

Confirm next meeting date /time

Linda Bird

**Date & Time**

20:00 UTC Wednesday 6th November 2019

**Location**

Zoom meeting: [https://snomed.zoom.us/j/471420169](https://snomed.zoom.us/j/471420169)

**Goals**

- To progress work on
  - URIs
  - Templates
  - ECL
  - Query language

**Attendees**

- Chair: Linda Bird
- Project Group:

**Agenda and Meeting Notes**

<table>
<thead>
<tr>
<th>Description</th>
<th>Owner</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Welcome and agenda</td>
<td>Linda Bird</td>
<td></td>
</tr>
</tbody>
</table>
URIs

Peter G. Williams

- Summary of discussions in Malaysia
  - Published version: https://confluence.ihtsdotools.org/display/DOCURI
  - Work in progress: https://confluence.ihtsdotools.org/display/WIPURI/URI+Standard
- Project group was asked to review WIP updates to URI proposal
  - Consistent structure and examples added to each URI page
  - New proposal on URIs for languages and language instances
  - New proposal on URIs for modelling resources e.g. valueSet, fhirStructureDefinition, openEHRArchetype

2.8 URIs for Modelling Resources

Agreed format

- http://snomed.info/valueSet/gps
- http://snomed.info/fhirStructureDefinition/conditionWithSnomedBinding
- http://snomed.info/openEhrArchetype/conditionWithSnomedBinding
- http://snomed.info/<resourcetype>/<resourcename>

ResourceTypes:
- valueSet
- fhirStructureDefinition
- openEHRArchetype

Expression Templates

Peter G. Williams

- Summary of discussions in Malaysia
  - Published version: https://confluence.ihtsdotools.org/display/DOCSTS/Template+Syntax+Specification
  - Work in progress: https://confluence.ihtsdotools.org/display/WIPSTS/Template+Syntax+Specification
  - Added a 'default' constraint to each replacement slot - e.g. default (72673000 |Bone structure (body structure)|)
  - Enabling 'slot references' to be used within the value constraint of a replacement slot - e.g. [[ +id << 123037004 |Body structure| MINUS << $findingSite2 ) @findingSite1]]
  - Allowing repeating role groups to be referenced using an array - e.g. $rolegroup[1] or $rolegroup[! =SELF]
  - Adding 'sameValue' and 'allOrNone' constraints to information slots - e.g. sameValue ($site), allOrNone ($occurrence)
  - Self
  - See changes in red here: 5.1. Normative Specification

- Recap of discussions in Malaysia regarding exchange of templates

Examples:

[[+id]]; [[1..*] @my_group sameValue(morphology)] (Finding site = [[ +id <<123037004 |Body structure| MINUS << $findingSite2 ) @findingSite1]], [Associated morphology] = [[ +id @my_morphology ]])

- Implementation feedback on draft updates to Expression Template Language syntax
- Use cases from the Quality Improvement Project:
  - Multiple instances of the same role group, with some attributes the same and others different. Eg same morphology, potentially different finding sites.

Note that QI Project is coming from a radically different use case. Instead of filling template slots, we're looking at existing content and asking "exactly how does this concept fail to comply to this template?"

For discussion:

[[0..1] [[0..1]] 246075003 Causative agent = [[+id << 410607006 |Organism|] @Organism]]

Is it correct to say either one of the cardinality blocks is redundant? What are the implications of 1..1 on either side? This is less obvious for the self grouped case.

Road Forward for SI

1. Generate the parser from the ABNF and implement in the Template Service
2. User Interface to a) allow users to specify template at runtime b) tabular (auto-completion) lookup STL
3. Template Service to allow multiple templates to be specified for alignment check (aligns to none-off)
4. Output must clearly indicate exactly what feature of concept caused misalignment, and what condition was not met.

Additional note: QI project is no longer working in subhierarchies. Every 'set' of concepts is selected via ECL. In fact most reports should now move to this way of working since a subhierarchy is the trivial case. For a given template, we additionally specify the "domain" to which it should be applied via ECL. This is much more specific than using the focus concept which is usually the PPP eg Disease.

FYI Michael Chu
<table>
<thead>
<tr>
<th>Description Templates</th>
<th>Kai Kewley</th>
<th></th>
<th><strong>Summary of discussions in Malaysia</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview of current use</strong></td>
<td><strong>Review of General rules for generating descriptions</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Removing tags, words</td>
<td>• Conditional removal of words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Automatic case significance</td>
<td>• Generating PTs from target PTs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Reordering terms</td>
<td><strong>Mechanism for sharing general rules - inheritance? include?</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Description Templates for translation</strong></td>
<td><strong>Status of planned specification</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression Constraint Language</th>
<th>Linda Bird</th>
<th></th>
<th><strong>Agreement in Malaysia - ECL will add the following term searching syntax (no regex - just wild card and word prefix any order):</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Term Search Type</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a. Wild Card Match (collation) - e.g.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• {{ term = wild(&quot;&quot;heart&quot;&quot; )}}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• {{ term = wild (sv);&quot;hjärta&quot;}}}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a. Word Prefix Any Order - e.g.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• {{ term = match:=&quot;hear att&quot;}}}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• {{ term = &quot;hear att&quot;}}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• {{ term = &quot;heart&quot;}}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Potential Examples</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &lt;&lt; 64572001 {Disease} ({{ term = &quot;heart&quot;}})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &lt;&lt; 64572001 {Disease} ({{ term = &quot;heart&quot;, languageCode = &quot;en&quot;}})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &lt;&lt; 64572001 {Disease} ({{ term = &quot;heart&quot;, languageCode = &quot;en&quot;}}) AND &lt;&lt; 64572001 {Disease} ({{ term = &quot;hjärta&quot;, languageCode = &quot;sv&quot;}})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &lt;&lt; 64572001 {Disease} ({{ term = &quot;heart&quot;, languageCode = &quot;en&quot;}}) OR &lt;&lt; 64572001 {Disease} ({{ term = &quot;hjärta&quot;, languageCode = &quot;sv&quot;}})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &lt;&lt; 64572001 {Disease} ({{ term = &quot;heart&quot;, languageCode = &quot;en&quot;}}) OR ({{ term = &quot;hjärta&quot;, languageCode = &quot;sv&quot;}})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (&lt;&lt; 64572001 {Disease} ({{ term = &quot;heart&quot;, languageCode = &quot;en&quot;}}) OR {{ term = &quot;hjärta&quot;, languageCode = &quot;sv&quot;}}) AND (&lt;&lt; 64572001 {Disease} ({{ term = &quot;heart&quot;, languageCode = &quot;en&quot;}}) OR ({{ term = &quot;hjärta&quot;, languageCode = &quot;sv&quot;}})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Recommendation to be made on (based on investigation of grammar):</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &lt;&lt; 64572001 {Disease} ({{ term = &quot;heart&quot;, languageCode = &quot;en&quot;}}) AND ({{ term = &quot;hjärta&quot;, languageCode = &quot;sv&quot;}})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (&lt;&lt; 64572001 {Disease} ({{ term = &quot;heart&quot;, languageCode = &quot;en&quot;}}) OR ({{ term = &quot;hjärta&quot;, languageCode = &quot;sv&quot;}}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (&lt;&lt; 64572001 {Disease} ({{ term = &quot;heart&quot;, languageCode = &quot;en&quot;}}) MINUS (&lt;&lt; 64572001 {Disease} ({{ term = &quot;heart&quot;, languageCode = &quot;en&quot;}}) OR ({{ term = &quot;hjärta&quot;, languageCode = &quot;sv&quot;}})</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Cases</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Intentionally define a reference set for chronic disease. Starting point was ECL with modelling; This misses concepts modelled using the pattern you would expect. So important in building out that reference set.</td>
<td></td>
</tr>
<tr>
<td>• Authors quality assuring names of concepts</td>
<td></td>
</tr>
<tr>
<td>Checking translations, retranslating. Queries for a concept that has one word in Swedish, another word in English</td>
<td></td>
</tr>
<tr>
<td>• AU use case would have at most 3 or 4 words in match</td>
<td></td>
</tr>
<tr>
<td>• Consistency of implementation in different terminology services</td>
<td></td>
</tr>
<tr>
<td>• Authoring use cases currently supported by description templates</td>
<td></td>
</tr>
<tr>
<td>• A set of the &quot;ectomy&quot;s and &quot;itis&quot;s</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Questions</th>
<th></th>
</tr>
</thead>
</table>
• Do we include 'typeId' - e.g. << 64572001 |Disease| {D.term = "**heart**", typeId = 900000000000013009 |Synonym| })  
  NO
• Do we include 'type' - e.g. << 64572001 |Disease| {D.term = "**heart**", D.type = synonym}}  
  NO
• Do we include 'languageCode' - e.g. << 64572001 |Disease| {D.term = "**heart**", D.type = synonym, D.languageCode = "en"}}  
  YES
• Do we include 'caseSignificanceId' - e.g. << 64572001 |Disease| {D.term = "**Heart**", D.caseSignificanceId = 900000000000017005 |case sensitive|}}  
  NO
• Do we include 'caseSignificance' - e.g. << 64572001 |Disease| {D.term = "**Heart**", D.caseSignificance = sensitive}}  
  NO
• Do we include 'language' and 'version' - e.g. << 64572001 |Disease| {D.term = "**heart**"} VERSION = http://... LANGUAGE = (999001881000000108|Gastro LRS|, |GB English|})  
  NO
• Do we include syntactic sugar - e.g.  
  • << 64572001 |Disease| {preferredTerm = "**heart**", languageRefSet = en-gb})  
  • << 64572001 |Disease| {fullySpecifiedTerm = "**heart**", languageRefSet=en-gb})  
  • << 64572001 |Disease| {acceptableTerm = "**heart**", languageRefSet=en-gb})  
  • << 64572001 |Disease| {preferredTerm = "**heart**") FROM version = X, language = Y  
  NO
• Do we use/require the "D" at the start of "term"?  
  NO
• Packaging - How do we package this extension to ECL  
  • A new version of ECL - version 1.5
Recap discussions in Malaysia regarding querying historical patient records - e.g. find all patients with a respiratory disease in the last 10 years. Do we include patients whose records contain "Recurrent chest infection"? (an inactive concept)

Solutions suggested include:

1. **Multiple queries**
   
   Reverse memberOf function
   
   - What refsets is a concept a member of?

2. **Use historical associations** - either create map from inactive to active concepts, or update EHR to replace inactive concepts with active ones

3. Look at the latest snapshot before the concept is inactivated

4. Multiple queries run against successive releases of SNOMED CT with results collated

5. Update the EHRs to the latest concept using historical associations

6. Create an enhanced transitive closure table containing inactive concepts as their last known position

7. Augmented solution checks the position of replacement to determine concepts inactivated due to wrong placement

Proposed syntax to support execution of maps (Outstanding question: ECL or Query Language? Scope and packaging needs further discussion)

- Example use cases
  - Mapping from international substance concepts to AMT substance concepts
  - Anatomy structure and part association reference set - e.g. find the anatomical parts of a given structure

- Potential syntax to consider
  - **Functional**
    
    - `mapTarget (Anatomy structure and part association refset, << Upper abdomen structure)`
      
      Return the map targets from the given map refset, where the referencedComponent matches the condition

    - `mapSource (Anatomy structure and part association refset, << Liver part)`
      
      Return the referencedComponent from the given map refset, where the targetId matches the condition.

  - **Dot notation + Attribute refinement**
    
    - `Anatomy structure and part association refset . mapTarget`
    
    - `Anatomy structure and part association refset . referencedComponent` (Same as `^ Anatomy structure and part association refset`)
    
    - `Anatomy structure and part association refset; referencedComponent = << Upper abdomen structure; mapTarget`

  - **Dot notation + Filters**
    
    - `Anatomy structure and part association refset; referencedComponent = << Upper abdomen structure)`
    
    - `Anatomy structure and part association refset; mapTarget = << Upper abdomen structure)`

  - **Specify value to be returned**
    
    - `?mapTarget? Anatomy structure and part association refset`
    
    - `?mapTarget? Anatomy structure and part association refset; referencedComponent = << Upper abdomen structure)`

  - **Reverse memberOf function**
    
    - What refsets is a concept a member of?
**Returning attributes**

<table>
<thead>
<tr>
<th>Michael Lawley</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposal from Michael:</td>
</tr>
<tr>
<td>- Currently ECL expressions can match (return) concepts that are either the source or the target of a relationship triple (target is accessed via the 'reverse' notation or 'dot notation', but not the relationship type (ie attribute name) itself.</td>
</tr>
<tr>
<td>- For example, I can write:</td>
</tr>
<tr>
<td>&lt;404684003</td>
</tr>
<tr>
<td>&lt;404684003</td>
</tr>
<tr>
<td>- But I can’t get all the attribute names that are used by &lt;404684003</td>
</tr>
<tr>
<td>- Perhaps something like:</td>
</tr>
<tr>
<td>? R.type ? (&lt;404684003</td>
</tr>
<tr>
<td>- This could be extended to, for example, return different values - e.g.</td>
</tr>
<tr>
<td>?</td>
</tr>
</tbody>
</table>

**Query Language - Summary from previous meetings**

<table>
<thead>
<tr>
<th>Linda Bird</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples: version and language</td>
</tr>
<tr>
<td>- &lt;&lt; 64572001</td>
</tr>
<tr>
<td>- &lt;&lt; 64572001</td>
</tr>
<tr>
<td>- &lt;&lt; 64572001</td>
</tr>
<tr>
<td>- &lt;&lt; 64572001</td>
</tr>
<tr>
<td>- &lt;&lt; 64572001</td>
</tr>
<tr>
<td>- &lt;&lt; 64572001</td>
</tr>
<tr>
<td>- (* { { term = &quot;<em>heart</em>&quot; } }) VERSION <a href="http://snomed.info/sct/900000000000207008/version/20180131">http://snomed.info/sct/900000000000207008/version/20180131</a>, LANGUAGE Z MINUS (* { { term = &quot;<em>heart</em>&quot; } }) VERSION <a href="http://snomed.info/sct/900000000000207008/version/20170731">http://snomed.info/sct/900000000000207008/version/20170731</a>, LANGUAGE W</td>
</tr>
<tr>
<td>- X MINUS Y WHERE X = <em>, Y = (</em> { { term = &quot;<em>heart</em>&quot; } }) VERSION <a href="http://snomed.info/sct">http://snomed.info/sct</a> /900000000000207008/version/20180131, LANGUAGE W</td>
</tr>
</tbody>
</table>

**Notes**

- Allow nested where, version, language
- Scope of variables is inner query

**Examples: where**

<table>
<thead>
<tr>
<th>Linda Bird</th>
</tr>
</thead>
<tbody>
<tr>
<td>- X MINUS &gt;! X WHERE X = (&lt;&lt; 1234 : 5678 = &lt;&lt; 6547)</td>
</tr>
<tr>
<td>- X MINUS &gt;! X WHERE X = (&lt;&lt; 1234 : 5678 = &lt;&lt; 6547) VERSION <a href="http://snomed.info/sct">http://snomed.info/sct</a> /900000000000207008/version/20180131</td>
</tr>
<tr>
<td>- X MINUS &gt;! X WHERE X = (&lt;&lt; 1234 : 5678 = &lt;&lt; 6547), Y = (&lt;&lt; 1456) VERSION <a href="http://snomed.info/sct">http://snomed.info/sct</a> /900000000000207008/version/20180131</td>
</tr>
<tr>
<td>- X MINUS &gt;! X WHERE X = (&lt;&lt; 1234 : 5678 = &lt;&lt; 6547) VERSION <a href="http://snomed.info/sct">http://snomed.info/sct</a> /900000000000207008/version/20180131, LANGUAGE 999001881000000108</td>
</tr>
<tr>
<td>- X MINUS &gt;! X WHERE X = (&lt;&lt; 1234 : 5678 = &lt;&lt; 6547) VERSION <a href="http://snomed.info/sct">http://snomed.info/sct</a> /900000000000207008/version/20180131, LANGUAGE 999001881000000108</td>
</tr>
<tr>
<td>- X MINUS &gt;! X WHERE X = (&lt;&lt; 1234 : 5678 = &lt;&lt; 6547) VERSION <a href="http://snomed.info/sct">http://snomed.info/sct</a> /900000000000207008/version/20180131, LANGUAGE 999001881000000108</td>
</tr>
<tr>
<td>- X minus &gt;! X WHERE X = (&lt;&lt;&lt;&lt; M WHERE M = (&lt;&lt; 1234)) VERSION <a href="http://snomed.info/sct">http://snomed.info/sct</a> /900000000000207008/version/20180131, LANGUAGE 999001881000000108</td>
</tr>
</tbody>
</table>

**Notes**

- Allow nested variable definitions, but recommend that people don’t due to readability
- Scope of variables is the inner query
- No recursion e.g X WHERE X = 1234 MINUS X
  - ie can’t use a variable in its own definition
  - ie X is only known on the left of the corresponding WHERE, and not on the right of the WHERE
Keywords for Term-based searching:

- **D.term**
  - D.term = "*heart*"
  - D.term = wild:"*heart*"
  - D.term = regex:"*heart*"
  - D.term = match:"hear att"
  - D.term = (sv) wild: "*heart*"
- **D.languageCode**
  - D.languageCode = "en"
  - D.languageCode = "es"
- **D.caseSignificance**
  - D.caseSignificance = 900000000000448009 |entire term case insensitive|
  - D.caseSignificance = 900000000000017005 |entire term case sensitive|
  - D.caseSignificance = 900000000000200002 |only initial character case insensitive|
- **D.caseSignificanceId**
  - D.caseSignificanceId = "insensitive"
  - D.caseSignificanceId = "sensitive"
  - D.caseSignificanceId = "initialCharInsensitive"
- **D.typeId**
  - D.typeId = 900000000000003001 |fully specified name|
  - D.typeId = 900000000000013009 |synonym|
  - D.typeId = 900000000000550004 |definition|
- **D.type**
  - D.type = "FSN"
  - D.type = "fullySpecifiedName"
  - D.type = "synonym"
  - D.type = "textDefinition"
- **D.acceptabilityId**
  - D.acceptabilityId = 900000000000549004 |acceptable|
  - D.acceptabilityId = 900000000000548007 |preferred|
- **D.acceptability**
  - D.acceptability = "acceptable"
  - D.acceptability = "preferred"

Additional Syntactic Sugar

- **FSN**
  - FSN = "*heart"
    - D.term = "*heart", D.type = "FSN"
    - D.term = "*heart", D.typeId = 900000000000003001 |fully specified name|
  - FSN = "*heart" LANGUAGE X
    - D.term = "*heart", D.type = "FSN", D.acceptability = *LANGUAGE X
    - D.term = "*heart", D.typeId = 900000000000003001 |fully specified name|, acceptabilityId = *LANGUAGE X

- **synonym**
  - synonym = "*heart"
    - D.term = "*heart", D.type = "synonym"
    - D.term = "*heart", D.typeId = 900000000000013009 |synonym|
  - synonym = "*heart" LANGUAGE X
    - D.term = "*heart", D.type = "synonym", D.acceptability = *LANGUAGE X
    - D.term = "*heart", D.typeId = 900000000000013009 |synonym|, (D.acceptabilityId = 900000000000549004 |acceptable| OR D.acceptabilityId = 900000000000548007 |preferred|) LANGUAGE X

- **synonymOrFSN**
  - synonymOrFSN = "*heart"
    - synonym = "*heart" OR FSN = "*heart"
    - D.term = "*heart", (D.type = "synonym" OR D.type = "fullySpecifiedName")
    - synonymOrFSN = "*heart" LANGUAGE X
    - synonym = "*heart" OR FSN = "*heart" LANGUAGE X
    - D.term = "*heart", (D.type = "synonym" OR D.type = "fullySpecifiedName"), D.acceptability = *LANGUAGE X

- **textDefinition**
  - textDefinition = "*heart"
    - D.term = "*heart", D.type = "definition"
    - D.term = "*heart", D.typeId = 900000000000550004 |definition|
    - textDefinition = "*heart" LANGUAGE X
    - D.term = "*heart", D.type = "definition", D.acceptability = *LANGUAGE X
    - D.term = "*heart", D.typeId = 900000000000550004 |definition|, D.acceptabilityId = *LANGUAGE X

- **Unacceptable Terms**
  - (D.term = "*heart") MINUS (D.term = "*heart", D.acceptability = *LANGUAGE X)
### Language preferences using multiple language reference sets

- LRSs that use the same Language tend to use 'Addition' - i.e. child LRS only includes additional acceptable terms, but can override the preferred term
  - E.g. Regional LRS that adds local dialect to a National LRS
  - E.g. Specialty-specific LRS
  - E.g. Irish LRS that adds local preferences to the en-GB LRS
- LRSs that define a translation to a different language tend to use 'Replacement' - i.e. child LRS replaces set of acceptable and preferred terms for any associated concept
  - E.g. Danish LRS that does a partial translation of the International Release
    - 999999 [Danish language reference set] ELSE [GB English reference set]

### Next steps

<table>
<thead>
<tr>
<th>Next steps</th>
<th>Linda Bird</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Discuss and plan next steps</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Confirm next meeting date /time</th>
<th>Linda Bird</th>
</tr>
</thead>
</table>

---

File: Microsoft Excel Spreadsheet *RegexCheat.xlsx*  
Modified: 2019-Nov-06 by Linda Bird