GETTING GROOVY WITH SNOMED CT – SOLVING PRACTICAL PROBLEMS WITH SCRIPTING IN SNOW OWL
What is Groovy?

Groovy is a dynamic programming language that builds upon the strengths of Java with a very flat learning curve for people familiar with object-oriented languages. It is easy to read and learn and has scripting and domain-specific language support which makes it ideal for providing scripting support for SNOMED CT tools written in Java.
Groovy highlights

- **Runs on a Java VM**
  - Call Groovy code from Java and to call Java code from Groovy.
  - Works with well-tested frameworks such as Spring
  - Groovy classes/Groovy scripts

- **Everything is an object**
  - -1.abs()
  - println "This is a string 321".toSet().sort().join().reverse().replaceAll(" ", ");

- **Static or optional typing**
  - str = “I’m a string”

- **Collections are native to the language**
  - code_systems = [‘SNOMED CT’, ‘ICD-10’, ‘ATC’]
Groovy highlights - Closures

A closure is a piece of code wrapped up as an object.

def closure = { param -> println("hello \${param}") } 
def closure = { println "hello " + it } 
closure.call("world!") //hello world!

Closure envelope = { person -> new Letter(person).send() } 
addressBook.each (envelope) 
addressBook.each { new Letter(it).send() }

def value = [1, 2, 3].findAll { it > 1 } 
assert value == [2, 3]

def service = new SnomedHierarchicalService() 
def targetConcepts = service.getTargetConcepts(CLINICAL_FINDING_ID, ASSOC_FINDING_ID) 
targetConcepts.each { 
    println "ID: $it.id label: $it.label"
}
def service = new SnomedHierarchicalService()
def rootConcepts = service.rootConcepts
def rootNodes = service.getSubtypes(rootConcepts[0].id)
def writer = new StringWriter()
def xml = new MarkupBuilder(writer)
xml.rootConcepts() {
    rootNodes.each {
        def directChildren = service.getSubtypes(it.id)
        rootNode(id: it.id, label: it.label) {
            directChildren.each {
                childNode(id: it.id, label: it.label)
            }
        }
    }
}
println writer.toString()

<rootConcepts>
    <rootNode id='78621006' label='Physical force'>
        <childNode id='80032004' label='Fire' />
        <childNode id='18213006' label='Electricity' />
        ...
        <childNode id='125676002' label='Person' />
    </rootNode>
</rootConcepts>

<rootNode id='370115009' label='Special concept'>
    <childNode id='362955004' label='Inactive concept' />
    <childNode id='363743006' label='Navigational concept' />
</rootNode>

<rootNode id='363787002' label='Observable entity'>
    ...
</rootNode>
Groovy highlights - Dynamic Object Orientation

- Meta Object Protocol
- Builders
- ExpandoMetaClass

```java
//extending a SNOMED CT concept representation
ICloneableMetaClass getToHtmlString {
    "<tr><td>
    delegate.id + "</td><td>
    delegate.label + "</td></tr>"
} println it.toHtmlString

- DSL support
Snow Owl’s Groovy support

- Snow Owl provides an Integrated Development Environment (Groovy-Eclipse) that allows editing, compiling, running, and debugging Groovy scripts and classes from within Snow Owl.

- The integrated editor includes syntax highlighting, type inferencing, formatting, debugging, refactoring, auto-completion.

- Ideal for terminologist, medical informatics specialist or anyone interested in terminology management and authoring.
Snow Owl’s Groovy editor

```groovy
import groovy.text.SimpleTemplateEngine

def escQuery = ""

<<404684003|Clinical finding|:
  246454002|Occurrence|: 255399007|Congenital,
  370135005|Pathological process|:<<263680009|Autoimmune

"

//initialize a service for evaluating an ESCG query expression
def escgEvaluator = new EscgEvaluatorService()

//evaluate the ESCG query
def concepts = escgEvaluator.evaluate(escQuery)

//initialize a service
def indexEvaluator = new IndexQueryEvaluatorService(concepts)

//find all SNOMED CT concepts with 'congenital' in a set
concepts = indexEvaluator.evaluate('congenital')

//extending the string representation of the SNOMED CT concept
IComponent.metaClass.getToHtmlString { 
  "<tr><td>" + delegate.id + "</td><"td><"td>" + delegate.name + "</td><"td>" + delegate.misc + "</td><"td>" + delegate.display + "</td><"td>" + delegate.id + "</td><"td>" + delegate.name + "</td><"td>" + delegate.misc + "</td><"td>" + delegate.display + "</td>" 
```

Press 'Space' to show Template Proposals
Press 'Tab' from proposal table or click for focus
Snow Owl service API

- Most frequently used services
  - FullTextSearchService
  - HierarchicalService
  - LookupService
  - QueryEvaluatorService
  - ExpressionService
  - CodeSystemService
  - AdminService
Candidates for scripting

- Ad-hoc queries
- Complex queries
- Ad-hoc reports in custom formats (text, markup)
- Artefact generation/updates
- Bulk terminology updates including the generation of terminology artefacts
- Terminology server access via REST protocol
  - Even from shell:
    ```
    #!/usr/bin/env
groovy println "Hello, World!"
    ```
- DSL language development
- Combinations of the above with possible automatized execution
Concrete examples - Medicinal product descriptions

- List the SNOMED CT concept ID, PT, and Synonyms for
  - all descendants of ‘Drug allergen or pseudoallergen’
  - not members in the Ingredients refset
- tab separated table format.

<table>
<thead>
<tr>
<th>ConceptId</th>
<th>PT</th>
<th>Synonym</th>
<th>Synonym</th>
<th>Synonym</th>
<th>Synonym</th>
</tr>
</thead>
<tbody>
<tr>
<td>387121001</td>
<td>Clonidine hydrochloride</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>96098007</td>
<td>Valaciclovir</td>
<td>Valacyclovir</td>
<td>Valacyclovir</td>
<td></td>
<td></td>
</tr>
<tr>
<td>372511001</td>
<td>Benazepril</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>373544004</td>
<td>Antazoline</td>
<td>Phenazoline</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6612003</td>
<td>Chloramphenicol Sodium Succinate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>88427007</td>
<td>Methyl acetylene</td>
<td>Propine</td>
<td>Allylene</td>
<td>1-Propyne</td>
<td>Propyne</td>
</tr>
<tr>
<td>38911000133101</td>
<td>Dapsone only</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>372485004</td>
<td>Tiagabine</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>404839003</td>
<td>Sodium Ibandronate</td>
<td>Ibandronate Sodium</td>
<td>Ibandronate sodium</td>
<td></td>
<td></td>
</tr>
<tr>
<td>85603004</td>
<td>Triphenamyl</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>....</td>
<td>.....</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Concrete examples – Ad-hoc report in html format

- Find all Clinical findings where the concept definition describes congenital origin and an autoimmune pathological process.
- Filter the results to concepts with any description that includes the word ‘congenital’.
- Render the results in an HTML table

<table>
<thead>
<tr>
<th>SNOMED CT ID</th>
<th>Preferred term</th>
</tr>
</thead>
<tbody>
<tr>
<td>230672006</td>
<td>Congenital myasthenia</td>
</tr>
<tr>
<td>193216006</td>
<td>Congenital and developmental myasthenia</td>
</tr>
<tr>
<td>230677000</td>
<td>Congenital end-plate acetylcholinesterase deficiency</td>
</tr>
<tr>
<td>230673001</td>
<td>Congenital end-plate acetylcholine receptor deficiency</td>
</tr>
</tbody>
</table>
Concrete examples - Medicinal product descriptions

Downstream users of NRC may have limitations in their systems as to the number of characters allowed within descriptions that are displayed. There was concern about the readability of these shorter descriptions for drugs that

- met a particular prescribing use case
- were not oral tablets
- contained less than 3 active ingredients

Therefore the customer wanted to output a list of preferred terms for these drugs along with the short description so that pharmacists could determine if there was potential for confusion when using the short names.

To answer this request, our customer first created a semantic query using the HL7 TermInfo standard to find drugs meeting a particular use case (medicinal product preparations) that did not have an oral tablet dose form. Next, the developer iterated through these results, discarding drugs that had more than 3 active ingredients. Finally, the developer created a file with a table of the preferred terms and short names.
Concrete examples – SNOMED CT DSL

- '386536003'.pt
- '386536003'.terms
- '386536003'.exist
- '126134000'.eachSynonym { println "ID: ${it.id} Term: ${it.label} »}"
- '410607006'.subTypes.each { println it.label }
- def sortedBySizeSynonyms = '126134000'.synonyms.sort {it.label.size()}
- sortedBySizeSynonyms.each { println it.label }
- escgQuery.evaluate.filter('Congenital').toHtml
Future work

- Server-side scripting, scheduled automation
- Service injection into the runtime environment (no need to import and instantiate services)
- Continue to expose Snow Owl’s functionality as a high-level API for scripting
Questions?