Use of GELLO v.1.x, GLIF 3.5, SNOMED-CT and EN 13606 archetypes

Peter Scott
Andrew McIntyre
Jared Davision
Peter Tattam

www.medical-objects.com.au
peter@medical-objects.com.au
1. Standards integration achieves real implementations even in v.2 environments like Australia

2. A GELLO v.1.x parser/interpreter/engine can be used to:
   a) enable decision criteria in GLIF
   b) ‘script’ queries and processing on retrieved EHR data as well as SNOMED-CT knowledge and LOINC information space.

3. A virtual medical record (vMR) helps abstract away the "my information model is better than your information model" problem as well as the "my reference information model is not fully RIM compliant" problem.
Medical-Objects’ HL7 GELLO v.1.0 projects

- Projects – 1. CDS for lymphoma 2. Setup of a small pathology laboratory
- GLIF was used for the former. Both used EN 13606 archetypes converted into a form accessible from the vMR.
- Decision was made to use GELLO encoded logic.
- GELLO v. 1 features
  - HL7/ANSI standard
  - OCL derived, but extended for health informatics functionality
  - Rich querying facilities.
  - Strongly typed, Object oriented, Declarative
  - vMR - Integrates well with HL7 and EN 13606 archetypes
  - Support for SNOMED-CT, LOINC, ICD-10 through vMR/CodedValue and more-so with CTS
- Implemented what we believe to be some of the first practical implementations of GELLO v.1 worldwide.
Rector’s Model of models

Diagram modified from Alan Rector’s “Model of models” found at:
Working with GELLO

- We completed the adaption of GLIF to the GELLO language
- We used EN 13606 archetypes as data structures within the GLIF and as the basis for templates. The templates are GELLO enabled.
- Peter Tattam and Andrew McIntyre improved GELLO spec’s BNF, and built some tools (all available to informatics colleagues for playing and comments):
  - GELLO authoring tool and parser/interpreter
  - GLIF authoring tool and editor
  - SNOMED-CT server and
  - CEN 13606 Template editor
About complex information to be communicated

Extract -> Folder -> Composition ->(Section) -> Entry - > (Cluster) -> Element

<table>
<thead>
<tr>
<th>EHR HIERARCHY COMPONENT</th>
<th>DESCRIPTION</th>
<th>EXAMPLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>EHR_EXTRACT</td>
<td>The top-level container of part or all of the EHR of a single subject of care, for communication between an EHR Provider system and an EHR Recipient.</td>
<td>(Not applicable)</td>
</tr>
<tr>
<td>FOLDER</td>
<td>The high-level organisation within an EHR, dividing it into compartments relating to care provided for a single condition, by a clinical team or institution, or over a fixed time period such as an episode of care.</td>
<td>Diabetes care. Schizophrenia. Cholecystectomy. Paediatrics. St Mungo's Hospital. GP Folder. Episodes 2000-2001, Italy.</td>
</tr>
<tr>
<td>ENTRY</td>
<td>The information recorded in an EHR as a result of one clinical action, one observation, one clinical interpretation, or an insertion. This is also known as a clinical statement.</td>
<td>A symptom, an observation, one test result, a prescribed drug, an allergy reaction, a diagnosis, a differential diagnosis, a differential white cell count, blood pressure measurement.</td>
</tr>
<tr>
<td>CLUSTER</td>
<td>The means of organising nested multiset data structures such as time series, and to represent the columns of a table.</td>
<td>Audiology results. Electroencephalogram interpretation. weighted differential diagnoses.</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>The leaf node of the EHR hierarchy, containing a single data value.</td>
<td>Systolic blood pressure, heart rate, drug name, symptom, body weight.</td>
</tr>
</tbody>
</table>
Embedded GELLO v.1.x

- Embedded component within a GLIF and Archetypes framework.
- Developed using a LALR(1) parser framework in conjunction with a Delphi Object Pascal HL7 framework.
- It is interpretive in nature.
- Gello expressions are compiled at run time and stored as an internal object oriented expression tree.
- Execution speed is facilitated by the use of object oriented techniques.
- There is no “byte code” to execute, all calls are made natively to the HL7 framework.
- GELLO expressions can be implemented using an embedded IDE called “Mowgli”.
- Library facilities have been developed whereby frequently used GELLO expressions can be run indirectly from within another GELLO expression.
- This library now has a syntax checker for whole library.
let a:integer = 1
let b:integer = 2

a+b
a GELLO v.1.x example

```
let a:integer = 1
let b:integer = 2

a+b
```

**Outer Expression**

- `let a:integer = 1`  
- `let b:integer = 2`

**Inner Expression**

- `a+b`
GLIF 3

- States
- Action steps

- Review history and examination
- Review/order pathology
- Review/order imaging
- Review staging
States
Action steps
Decision steps
  Conditional
    User driven
    automated
GLIF

- **States**
- **Action steps**
- **Decision steps**
  - Conditional
  - User driven
  - Automated
  - Branched
Treatment with chemotherapy or pelvic irradiation may lead to infertility and, given the long life expectancy following successful treatment and young age at which many patients present, it is crucial to address reproductive issues before treatment planning commences, except in cases where emergency treatment is required. Where relevant, that is, when treatment carries a significant risk of affecting reproductive function, referral for harvesting and storage of sperm should be made and appropriate specialist consultations arranged to discuss preservation of fertility in female patients. Ovarian transposition may be considered if pelvic radiotherapy is planned, but results of this procedure are variable.
Rector’s Model of models

Diagram modified from Alan Rector’s “Model of models” found at:
// most recent platelets observation as a physical quantity:

let plateletsCode: CodedValue = Factory.CodedValue('777-3','LN')
let platelets_obs = observation->select(code = plateletsCode) ->sortby(absolutetime) ->last()

platelets_obs.value
Rector’s Model of models

Diagram modified from Alan Rector’s “Model of models” found at:
Let \( o: \) observation = parameter[1]

Let OpenAbdominal:CodedValue = Factory.CodedValue('108189003', 'SNOMED-CT')

Let SnomedTerm:CodedValue =
  if o.isdefined() then
    o.find_observation('1.1.1').Value.asType(CodedValue)
  else
    Factory.CodedValue('64572001', 'SNOMED-CT')
  endif

Let CholeIsAbdoOperation: Boolean = SnomedTerm.implies(OpenAbdominal)

CholeIsAbdoOperation
<table>
<thead>
<tr>
<th></th>
<th>Units</th>
<th>Reference Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sodium</td>
<td>mmol/L</td>
<td>135-145</td>
</tr>
<tr>
<td>Potassium</td>
<td>mmol/L</td>
<td>3.5-5.0</td>
</tr>
<tr>
<td>Chloride</td>
<td>mmol/L</td>
<td>95-110</td>
</tr>
<tr>
<td>Bicarbonate</td>
<td>mmol/L</td>
<td>22-32</td>
</tr>
<tr>
<td>Glucose</td>
<td>mmol/L</td>
<td>3.0-7.8</td>
</tr>
<tr>
<td>Glucose - fasting</td>
<td>mmol/L</td>
<td>3.0-5.5</td>
</tr>
<tr>
<td>Urea</td>
<td>mmol/L</td>
<td>1.0-3.0</td>
</tr>
<tr>
<td>Creatinine</td>
<td>mmol/L</td>
<td>21-91</td>
</tr>
<tr>
<td>Urate</td>
<td>mmol/L</td>
<td>0.15-0.40</td>
</tr>
<tr>
<td>Protein Total</td>
<td>g/L</td>
<td>50-75</td>
</tr>
<tr>
<td>Albumin</td>
<td>g/L</td>
<td>20-44</td>
</tr>
<tr>
<td>Bilirubin Total</td>
<td>umol/L</td>
<td>&lt;200</td>
</tr>
<tr>
<td>Bilirubin Direct</td>
<td>umol/L</td>
<td>&lt;7</td>
</tr>
<tr>
<td>Alkaline Phosphatase</td>
<td>U/L</td>
<td>50-300</td>
</tr>
<tr>
<td>Gamma GT</td>
<td>U/L</td>
<td>&lt;45</td>
</tr>
<tr>
<td>ALT</td>
<td>U/L</td>
<td>&lt;60</td>
</tr>
<tr>
<td>AST</td>
<td>U/L</td>
<td>&lt;60</td>
</tr>
<tr>
<td>LD</td>
<td>U/L</td>
<td>150-600</td>
</tr>
<tr>
<td>Calcium</td>
<td>mmol/L</td>
<td>2.10-2.50</td>
</tr>
<tr>
<td>Corrected Calcium</td>
<td>mmol/L</td>
<td>2.10-2.50</td>
</tr>
<tr>
<td>Phosphate</td>
<td>mmol/L</td>
<td>0.8-1.5</td>
</tr>
<tr>
<td>Iron</td>
<td>umol/L</td>
<td>10-30</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>mmol/L</td>
<td>&lt;5.5</td>
</tr>
<tr>
<td>Triglycerides</td>
<td>mmol/L</td>
<td>&lt;2.0</td>
</tr>
</tbody>
</table>
### Term Definitions

<table>
<thead>
<tr>
<th>Local Code</th>
<th>Lang.</th>
<th>Text</th>
<th>Description</th>
<th>Quality Control</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>at0017</td>
<td>en</td>
<td>AST</td>
<td>AST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0018</td>
<td>en</td>
<td>LD</td>
<td>LD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0019</td>
<td>en</td>
<td>Calcium</td>
<td>Calcium</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0020</td>
<td>en</td>
<td>Corrected Calcium</td>
<td>Corrected Calcium</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0021</td>
<td>en</td>
<td>Phosphate</td>
<td>Phosphate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0022</td>
<td>en</td>
<td>Protein Total</td>
<td>Protein Total</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0023</td>
<td>en</td>
<td>Albumin</td>
<td>Albumin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0024</td>
<td>en</td>
<td>Globulins</td>
<td>Globulins</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0025</td>
<td>en</td>
<td>Iron</td>
<td>Iron</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0026</td>
<td>en</td>
<td>Cholesterol</td>
<td>Cholesterol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0027</td>
<td>en</td>
<td>Triglycerides</td>
<td>Triglycerides</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0028</td>
<td>en</td>
<td>eGFR</td>
<td>eGFR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0029</td>
<td>en</td>
<td>Comments</td>
<td>Comments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0030</td>
<td>en</td>
<td>Osmolality</td>
<td>Osmolality</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0034</td>
<td>en</td>
<td>Non reportable items</td>
<td>Non reportable items</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0035</td>
<td>en</td>
<td>Working notes</td>
<td>Working notes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0036</td>
<td>en</td>
<td>Glucose - fasting</td>
<td>Glucose - fasting</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at0037</td>
<td>en</td>
<td>Backup glucose</td>
<td>Backup glucose</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Term Binding

<table>
<thead>
<tr>
<th>Local Code</th>
<th>Terminology</th>
<th>Code</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>at0019</td>
<td>AUSPATH</td>
<td>CA</td>
<td>Calcium</td>
</tr>
<tr>
<td>at0019</td>
<td>Integra400Plus</td>
<td>0-012</td>
<td>Calcium</td>
</tr>
<tr>
<td>at0019</td>
<td>Integra400Plus</td>
<td>12</td>
<td>Calcium</td>
</tr>
<tr>
<td>at0019</td>
<td>LN</td>
<td>2000-8</td>
<td>Calcium</td>
</tr>
<tr>
<td>at0019</td>
<td>SNOMED-CT</td>
<td>271240001</td>
<td>Calcium</td>
</tr>
</tbody>
</table>
let o:observation = parameter[1]
let Gender = Patient.gender
let Age: Integer = patient.age_at(factory.absolutetime("0"), now())
let Cr: Real =
    if o.isdefined() then
        o.find_observation('1.1.9').value_asFQ().value
    else
        0.0
    endif
let x: real = Cr^0.0113
let y: real = Age
let MaleResult: Real =
    if y > 17.0 then 186*x.power(-1.154)*y.power(-0.293)
    else
        0.0
    endif
let FemaleResult: real = MaleResult*0.742

let MaleResultText: String =
    if o.isdefined() then
        MaleResult.format(2,0)
    else
        '0.0'
    endif

let FemaleResultText: String =
    if o.isdefined() then
        FemaleResult.format(2,0)
    else
        '0.0'
    endif
//Try and determine if any open abdominal surgery

//setup a dummy surgical history
let Chole:CodedValue = Factory.SnomedCV("50228007|Cholecystectomy, exploration of duct and cholangiogram|")
let Appendix: CodedValue = Factory.SnomedCV("6225007|Laparoscopic Appendectomy|")
let KneeArthroscopy: CodedValue = Factory.SnomedCV("38969008|Knee Arthroscopy and Menisectomy|")

let PastSurgery: set(CodedValue) = set(Chole, Appendix, KneeArthroscopy)

//We use this as the test for a laparoscopic procedure
let Laparoscopic: CodedValue = Factory.SnomedCV("51315003|Laparoscopic Procedure|")
let AbdoSurgery: CodedValue = Factory.SnomedCV("385671003|Abdominal Cavity Surgery|")

//this is the test we are doing in the loop, done here as example only
let isLaparoscopic: Boolean = Appendix.implies(Laparoscopic)
let KneeLaparoscopic: Boolean = KneeArthroscopy.implies(Laparoscopic)
let KneeAndHo: Boolean = KneeArthroscopy.implies(AbdoSurgery)
let CholeIsHooperation: Boolean = Chole.implies(AbdoSurgery)

//see if cholangiogram was done as example
let Cholangiogramdone: Boolean = Chole.implies(Factory.SnomedCV("28367004|Cholangiogram|"))

//Iterate over past history, seeing if any laparoscopic surgery
let OpenSurgeryCount: Integer = PastSurgery->iterate(SX, 1:integer = 0 | if |SX.implies(AbdoSurgery) and not SX.implies(Laparoscopic)) then 1+1 else 1 endif

//see if there has been any open abdo surgery
if OpenSurgeryCount > 0 then
true
else
false
endif
```gello
Let o : observation = parameter[1]
Let Asthma : CodedValue = Factory.CodedValue('195967001', 'SNOMED-CT')
Let SnomedTerm1 : CodedValue =
  if o.isDefined() then
    o.find_observation('1.1.2').Value.asType(CodedValue)
  else
    Factory.CodedValue('64572001', 'SNOMED-CT')
  endif
let IsAsthma : Boolean = SnomedTerm1.implies(Asthma)
// return the result
IsIsthma
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Undefined</td>
<td>&lt;undefined&gt;</td>
</tr>
<tr>
<td>ASTHMA</td>
<td>CodedValue</td>
<td>195967001^^SNOMED-CT</td>
</tr>
<tr>
<td>SNOMEDTERM1</td>
<td>CodedValue</td>
<td>64572001^^SNOMED-CT</td>
</tr>
<tr>
<td>ISASTHMA</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td>[Result]</td>
<td>Boolean</td>
<td>False</td>
</tr>
</tbody>
</table>