LexGrid
Philosophy, Model and Interfaces

Harold R Solbrig
Division of Biomedical Statistics and Informatics
Mayo Clinic
Outline

• Why the LexGrid model was created
• LexGrid approach and principles
• Key aspects of the LexGrid model
Why LexGrid?

The situation in the late 1990’s:

- Multiple “terminologies” available
  - SNOMED-3 and SNOMED-RT
  - READ Codes
  - HCDA (ICD-8 w/ Mayo Extensions)
  - ICD-9-CM
  - ...
Why LexGrid?

The situation in the late 1990’s:

• DL was on the horizon
  • SNOMED-RT
  • GALEN
  • DAML+OIL beginning to emerge
Why LexGrid?

Mayo Health Sciences Research

• Multiple experiments and projects involving NLP, semi-automated record coding and classification, terminology-driven record retrieval, coded medical records, etc.
Why LexGrid?

Mayo recognized the need for re-use

- Terminologies have common characteristics
- Software should be reusable
  - Search and indexing
  - Query
  - Tree traversal
  - ...

Why LexGrid?

Part of the solution was the service oriented model:

• Aka “Breadboard”

• API specifications (OMG’s LQS was primary example)
Why LexGrid?

Service Oriented Model:

Application

Find designations matching “Myocard Infarct”

Client

Server

API Specification

Interface Specification

SNOMED-RT

ICD-9-CM
Why LexGrid?
API/Interface Specification

Provides a common semantics

• What is a “definition”, “designation”, “relationship”, ...
• Provides a common interface

Allows implementation to be specific to the terminology...
Why LexGrid API/Interface Specification

Server

SNOMED-RT

Semantic Mapping

Import

ICD-9-CM

Semantic Mapping

Import

...
Why LexGrid?
Harmonization on the model level
Why LexGrid?

LexGrid:
A Common Terminology Data Model

Descriptive *not* Prescriptive
LexGrid
Design Principles
Must span spectrum of “terminology”

- Code/value lists
- Thesauri (BT/NT)
- Classification Schemes
- Ontology & DL
LexGrid
Design Principles

Must provide common semantics for elements that are used in service API:

• (Textual) Definitions
• Designations
• Comments
  • Language / context / character set
• Hierarchies
• Relationships
LexGrid
Design Principles

Must support non-API components as tag/value pairs.

Must map ALL internal semantics to external (terminological) definitions.

• A property is useless if you don’t know the meaning of the tag

• A relation is useless if you don’t know its definition
LexGrid
Design Principles

Focus should be in information model vs. implementation:

- Originally implemented in LDAP
- XML Schema Model
- (Multiple) SQL Renderings to meet different user requirements
- Both Castor and Eclipse EMF renderings
LexGrid Model
Service Layer becomes secondary!

LexGrid - Philosophy and Model

05/01/2009
LexGrid
Key Components

Coding Scheme

Code

Definitions

Designations

Comments

Properties

Instructions

Associations

Relation

Source

Target

1..*

1..*

1..*
LexGrid

Key Components

Mappings

- supportedCodingScheme
- supportedSource
- supportedProperty
- supportedAssociation
- supportedPropertyQualifier
- ....

Transform a “local name” to a URI

- supportedAssociation localId="hasPart"
  URI="http://www.obofoundry.org/ro/ro.owl#part_of"
LexGrid
Future and Next Steps

Many loaders, interfaces available today

- OBO, OWL, RDF, UMLS, CSV, Ontylog, custom...

Several service API’s and implementations

- CTS, LexEVS (core of caBIG), LexWiki (sort of “implementation”)

LexGrid
Future and Next Steps

LexRDF

- OWL (2.0), DC, FOAF, SKOS (2008), RDF, RDFS, RO (to an extent) together now provide a reasonable overlay to LexGrid semantics

- Next step is to absorb and integrate
  - Mappings can now reference these
  - RDF import/export form that maintains model while using appropriate tags
More Information

http://LexGrid.org/

Solbrig.harold@mayo.edu