Application of Information Technology ■

LexGrid: A Framework for Representing, Storing, and Querying Biomedical Terminologies from Simple to Sublime

JYOTISHMAN PATHAK, PhD, HAROLD R. SOLBRIG, JAMES D. BUNTROCK, MS, THOMAS M. JOHNSON, CHRISTOPHER G. CHUTE, MD, DRPH

Abstract Many biomedical terminologies, classifications, and ontological resources such as the NCI Thesaurus (NCIT), International Classification of Diseases (ICD), Systematized Nomenclature of Medicine (SNOMED), Current Procedural Terminology (CPT), and Gene Ontology (GO) have been developed and used to build a variety of IT applications in biology, biomedicine, and health care settings. However, virtually all these resources involve incompatible formats, are based on different modeling languages, and lack appropriate tooling and programming interfaces (APIs) that hinder their wide-scale adoption and usage in a variety of application contexts. The Lexical Grid (LexGrid) project introduced in this paper is an ongoing community-driven initiative, coordinated by the Mayo Clinic Division of Biomedical Statistics and Informatics, designed to bridge this gap using a common terminology model called the LexGrid model. The key aspect of the model is to accommodate multiple vocabulary and ontology distribution formats and support of multiple data stores for federated vocabulary distribution. The model provides a foundation for building consistent and standardized APIs to access multiple vocabularies that support lexical search queries, hierarchy navigation, and a rich set of features such as recursive subsumption (e.g., get all the children of the concept *penicillin*). Existing LexGrid implementations include the LexBIG API as well as a reference implementation of the HL7 Common Terminology Services (CTS) specification providing programmatic access via Java, Web, and Grid services.

■ J Am Med Inform Assoc. 2009;16:305–315. DOI 10.1197/jamia.M3006.

Introduction

Semantic interoperability among health information systems is a longstanding aspiration of the health care community. Consensus-based information models and biomedical ontologies can address many interoperability issues. Consequently, the evolution of coding schemes, ontologies, and vocabularies, as well as other terminological resources, across the spectrum of detailed nomenclatures and sophisticated classifications, has accelerated dramatically.^{1–3,*} The complexity of patient data in electronic medical records (EMR), coupled with expectations that these data facilitate clinical decision-making, health care cost-effectiveness, medical error reduction, and evidence-based medicine,

makes obvious the role of ontologies as a foundation for comparable and consistent representation of patient information. However, in practice, health care providers and EMR system vendors alike confront the difficulties of incorporating elaborate ontologies and vocabularies into clinical workstations and record system clients while delivering an intuitive, friendly, and responsive interface that preserves the expressive power of these terminology resources. This can mainly be attributed to the following three reasons:

- *Incompatible Representational Formats:* Terminological resources, in general, are developed by an individual or a group of individuals, and are stored using different representations (e.g., text files, XML files, relational databases) depending on various factors ranging from application requirements to ease of use, or just as a matter of preference. Even though standards for a consistent distribution and representational format for such resources have been proposed (ISO TC 215, WG3 on Health Concept Representation [<http://www.tc215wg3.nhs.uk/pages/default.asp>] has assumed this task), standards will not address the reality that using a particular ontology often remains a formidable burden for the end user or vendor. Consequently, tools that leverage multiple, heterogeneous terminologies or ontologies are required to implement custom techniques for parsing and uniformly interpreting the ontological knowledge—a process that can be highly cumbersome and labor-intensive.
- *Multiple Ontology Modeling Languages:* As ontologies enjoy increasing adoption within a variety of biomedical

Affiliation of the authors: Division of Biomedical Statistics and Informatics, Mayo Clinic, Rochester, MN.

The authors thank our colleagues in the HL7 Vocabulary and the ISO TC215 Semantic Content Working Groups, particularly those associated with the evolution of CTS and CTS 2, for their input and critique. The Vocabulary and Common Data Elements Working Group of the caBIG community have contributed importantly to the evolution and deployment of LexBIG. Support for this work has derived in part from NIH grants and contracts, including LM07319 and multiple caBIG funding instruments.

Correspondence: Jyotishman Pathak, PhD, Division of Biomedical Statistics and Informatics, Mayo Clinic, 200 1st Street SW, Rochester, MN 55905; e-mail: <pathak.jyotishman@mayo.edu>.

Received for review: 09/19/08; accepted for publication: 02/09/09.

*(In this paper, for the sake of simplicity, we use the terms “ontologies”, “vocabularies”, “terminologies”, “classifications” and “coding schemes” interchangeably.)

application contexts, their users drive requirements that result in the building or modification of ontology modeling languages (e.g., Open Biomedical Ontologies—OBO,⁴ Web Ontology Language—OWL⁵). For example, to facilitate ontology reuse and scalable ontology reasoning and querying, various modular ontology languages such as Distributed Description Logics,⁶ E-connections,⁷ and Package-based Description Logics⁸ are being developed.^{9,10} Although useful, the development of multiple ontology languages necessitates that applications comprehend them, which in itself can be a non-trivial task. In practice, this leads to redundant, inefficient, often expensive, and frequently semantically distorting efforts to create interfaces to ontologies which proliferate across a multitude of vendors, applications, provider institutions, and even departments or user groups.

- *Lack of Common Tools and Application Programming Interfaces*: Along with multiple ontology modeling languages and formats, multiple and unique tools and application programming interfaces get developed. For example, to edit and query OBO and OWL ontologies, OBO-edit¹¹ and Protege¹² environments, respectively, are widely used within the biomedical ontology community. Similarly, before the development of LexGrid, there was no single tool or programming interface that could uniformly access common terminologies such as International Classification of Diseases (ICD) or Current Procedural Terminology (CPT). Consequently, applications using such ontologies must build separate appropriate tooling and programming interfaces (APIs) implementations. Although few tools exist that enable conversion of ontologies developed in one language to another,¹³ this does not mitigate the requirement to develop software that exposes a “common and uniform API” for developing and using ontologies modeled in different languages and formats. Access to a robust, scalable, consistent, and available suite of ontology services would eliminate this barrier to using ontologies and reduce the variation in clinical description attributable to interface dependencies.

To address these requirements, there was a need to design intermediary software between ontologies and applications that allowed managing the navigational and querying challenges implicit within sophisticated ontologies and vocabularies. Historically, this challenge had been left as an exercise to the end user or vendors. Most efforts to develop terminology interface software have dramatically underestimated the challenge. Barring a few,^{12,14,15} virtually all efforts at interface development had been proprietary or integrally bound with application environments. The consequence was that redundant, insular, noncomparable, nonscalable, and nonincremental terminology-access solutions prevailed. Neither practitioners nor patients were well served by this status quo; nor could the promise of individualized medicine and care to improve decisions, avoid errors, support research and new knowledge discovery, and continuously improve quality be realized. Specifically, some of the desirable characteristics of such software include the following^{16,17}:

- *Functional Characteristics*:
 - *Direct concept retrieval*: the ability to retrieve various concept attributes such as definitions or context syn-

onyms when the concept identifier is known. This may be a simple concept look-up by the identifier, or the specification for the concept in a particular language.

- *Associated text-based concept retrieval*: the ability to retrieve concept attributes when one knows only terms or phrases that are similar to the concept sought. This is a “smart” term look-up. More sophisticated implementations may tolerate spelling errors and support synonym and near-synonym resolution.
- *Hierarchy traversal*: the ability to navigate relationships between concepts (e.g., finding sub- or super-concepts of a given concept), or to recursively return all the “children” of a given concept (e.g., all drugs in the penicillin family).
- *Metadata access*: the ability to retrieve information about versions, allowed relationships, languages, and other capabilities of a particular ontology.
- *Complex queries*: the ability to access information combinatorially, e.g., list the immediate parent concepts of all concepts that have a term that contains the word “infarction”.
- *Performance Characteristics*:
 - *Static performance*: how much time does it take to respond to a single request, absent competing user system load? How to include optimizations such as local caching or code parallelization for improving query response time?
 - *Dynamic performance*: how a system responds to different loads? Does the system behave in a predictable fashion when subjected to different loads? Can it be configured to guarantee a minimum average and worst case response time?
- *Architectural Characteristics*:
 - *Fault resilience*: can services deployed in a data-grid fashion sustain hardware failure and recover from software failure without crashing? Mission-critical clinical applications would seek high availability.
 - *Security*: can the services prevent unauthorized alteration or disruption of content? How to ensure restriction of access to authorized users?
 - *Distributed*: how rapidly can the changes introduced into a master server be synchronized to slave/local copies along a chain of distribution? Updates must not destroy unique local content.
 - *Federated*: is it possible to maintain cross linkages between and among components of a single, large terminology or related terminologies with cross-referenced content? Similarly, is it possible to cross terminology boundaries to create composite terms from different sources, e.g., disease and anatomy?
 - *Platform and language independence*: can the services be operated independent of the programming language used for implementation, operating system and hardware platform?

Toward this end, in this paper we introduce the Lexical Grid (LexGrid) project which is an on-going community-driven initiative, coordinated by the Mayo Clinic Division of Biomedical Statistics and Informatics, that builds upon a set of common tools, data formats, and read/update mechanisms for storing, representing and querying biomedical ontologies and vocabularies.

In particular, the terminology and ontology resources in LexGrid are intended to be:

- represented using a common terminology model
- accessible through a set of common API's
- joined through shared indexes
- accessible online
- downloadable
- cross-linked (i.e., have interontology mappings)
- loosely coupled
- locally extensible
- globally revised, and
- available on the "grid" anytime or anywhere

The key principle of LexGrid is to accommodate multiple vocabulary and ontology distribution formats and support of multiple data stores for federated vocabulary and ontology access. This is facilitated via a common terminology model that defines how ontologies are formatted and represented programmatically, and is intended to be flexible enough to accurately represent a wide variety of ontologies developed in different languages and other lexically based knowledge resources. The LexGrid model provides the semantic basis for the development of consistent and rich APIs to access multiple vocabularies that support lexical search queries, hierarchy navigation, and various other features. Existing implementations include the LexBIG API as well as a reference implementation of the HL7 Common Terminology Services (CTS) (<http://informatics.mayo.edu/LexGrid/downloads/CTS/specification/ctsspec/cts.htm>) specification that provides programmatic access via Java, Web, and Grid services. These open-source API middleware are deployed in production by different projects and initiatives, both internal and external to Mayo Clinic, including NCI's Cancer Biomedical Informatics Grid (caBIG) (<https://cabig.nci.nih.gov/>), the National Center for Biomedical Ontology (NCBO) (<http://www.bioontology.org>), the Biomedical Grid Terminology project (BiomedGT) (<http://www.biomedgt.org>), and the World Health Organization International Classification of Diseases (ICD-11) (<http://www.who.int/classifications/icd/en/>) development process.

Related Work

Mayo Clinic's experience in using and contributing to terminology services and navigators dates to early versions of Lexical Technology, Inc's (later known as Apelon, Inc) (<http://www.apelon.com>) HyperCard-based Meta-Card, the Macintosh HyperCard-based browser¹⁸ of the first UMLS releases. We developed several ad hoc browsers and matching software in that same era,^{19,20} in addition to a long series of statistically based approaches to concept mapping.²¹ We will limit the remainder of this discussion to developments (by ourselves and others) that are germane to this manuscript.

Yet Another Terminology Navigator/Metaphrase

Yet Another Terminology Navigator (YATN) was originally developed in close collaboration between Mayo Clinic and Lexical Technology (LTI) in the mid-90s. It provided some key services and features for term normalization, term completion, semantic locality, and lexical matching. It later evolved into the development by LTI of a product called MetaPhrase²² which was a scalable, middleware component designed to provide access to terms and concepts from the

UMLS Metathesaurus (<http://www.nlm.nih.gov/research/umls/>). In addition to the features present in YATN, MetaPhrase provided intuitive and user-friendly ways for semantically navigating knowledge resources.

OpenGALEN

OpenGALEN (<http://opengalen.org>) provides access to the GALEN terminological content which is a computer-based multilingual coding system for medicine. This terminological content is known as the GALEN Common Reference Model, and the representation scheme used to build this model is known as GRAIL—the GALEN Representation and Integration Language. The GALEN Common Reference Model is delivered and used via a software device called a GALEN Terminology Server developed by multiple software vendors associated with OpenGALEN.²³ An important role of OpenGALEN is to maintain a specification of the GALEN Terminology Server which is essentially concerned with the definition and behavior of the GRAIL language implemented within the server, and with describing the kinds of services that must be available to client applications.

Lexicon Query Services

The Lexicon Query Service (LQS),²⁴ published by the Object Management Group (OMG), specifies a set of Interface Description Language (IDL) interfaces to be used in querying and accessing computerized medical terminology services. The IDL is an ISO standard method of defining abstract signatures in an object-based, distributed computing environment. The complete OMG specification defines how methods would be invoked from a wide variety of programming languages, and describes the expected behavior of the methods. In particular, the specification attempts to address a broad-spectrum of clinical terminologies, from simple code/value tables to sophisticated description-logic based terminologies, such as GALEN or SNOMED-RT. The object-based approach enabled the LQS to add functionality as needed, in incremental layers.

Health Level 7 (HL7)

Health Level 7 (HL7) (<http://www.hl7.org>) is an ANSI-accredited standards organization that is focused on creating standards for the exchange, management and integration of data that support clinical patient care and the management, delivery and evaluation of health care services. Historically, HL7 has concentrated on interoperability between health care information systems and has worked with a message-based information passing architecture. Early HL7 public releases (e.g., version 2.1) made only limited use of terminologies, and those that it did use were mostly simple code/text tables. The current Version 3 of the standard makes extensive use of terminological resources, some of which third parties supply and some of which have been internally developed.

Health Level 7 recognized that the added complexity multiple terminology resources, required for V3, necessitated a software abstraction layer, which the Vocabulary Technical Committee is refining under the rubric of Common Terminology Services 2 (CTS 2) (<http://hssp.wikispaces.com/cts2>). This in turn expands on the original functionality developed in HL7's Common Terminology Service specification (CTS). CTS 2 is an Application Programming Interface

(API) specification that is intended to describe the basic functionality that will be needed by HL7 Version 3 software implementations to query and access terminological content. It is specified as an API rather than a set of data structures to enable a wide variety of terminological content to be integrated within the HL7 Version 3 messaging framework without the need for significant migration or rewrite.

The Mayo reference implementation of CTS is built upon LexGrid repositories and technologies. More information is available at: <http://informatics.mayo.edu/LexGrid/index.php?page=ctsspec>.

Apelon Distributed Terminology System

The Apelon Distributed Terminology System (DTS)²⁵ is an integrated set of open-source components that provides comprehensive terminology services in distributed application environments. It assists in the maintenance and deployment of vocabularies within health care applications such as interface engines, order-entry systems, and decision support systems. The DTS enables data normalization (i.e., matching of text input to standardized terms and concepts via lexical analysis), code translation (i.e., mapping of clinical data to standard coding systems such as ICD-9), semantic navigation (i.e., browsing of a rich set of hierarchical and nonhierarchical relationships between concepts), and a host of additional features. The DTS APIs and management applications are available for both Java and Microsoft.NET environments. Furthermore, DTS is bundled with an extensible editor that enables the enhancement of DTS Knowledge Base by adding new content and localizing it for specific purposes.

The UMLS Knowledge Source Server

The UMLS project develops "Knowledge Sources", namely the UMLS Metathesaurus, the Semantic Network, and the Specialist Lexicon, that can be used by a wide variety of applications to overcome retrieval problems caused by differences in terminology and the scattering of relevant information across many databases. All these sources can be accessed via an array of software such as the NLM Gateway and the UMLS Knowledge Source Server (KSS).²⁶

From a terminology service perspective, the KSS component is the most relevant to our work. As described in the UMLS web site, KSS is "an application that provides internet access to the Knowledge Sources. Its purpose is to make UMLS data more accessible to users and in particular to system developers. The system architecture allows remote site users (individuals as well as computer programs) to send requests to a server at the National Library of Medicine. Access to the system is provided through the world wide web and through an Application Programming Interface (API)." While KSS has been implemented specifically for UMLS, it influenced various aspects of the LexGrid implementation including specifying the functionality and signature of the terminology services.

Semantic Web Technologies

The Semantic Web is an extension of the existing World Wide Web where semantics of the information and services on the Web are defined, making it possible for the Web to understand and satisfy the requests of people and machines to use Web content.²⁷ This is facilitated by a host of Semantic Web specifications/languages for description and character-

ization of data representation models and the data itself (e.g., OWL,⁵ RDF²⁸), for querying the data (e.g., SPARQL²⁹), and to extract and bind this information to traditional data sources to ensure interchange with data from other sources (e.g., GRDDL³⁰). These specifications are in turn implemented by various tools and APIs that provide functionality akin to terminology services. We mention two of them, relevant to LexGrid.

- *Protégé Ontology Development Environment*: Protege^{12,31} is an extensible, platform-independent environment for creating and editing ontologies and knowledge bases. It supports two main ways of modeling ontologies via the Protege frames and the Protege OWL editor plugins, and is being widely used by members of different communities ranging from biomedicine to intelligence gathering to manufacturing. At its core, Protege implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Various tools, including the frames and OWL editors, leverage protege's plug in architecture and its Java-based API.
- *The OWL API*: the OWL API,^{14,15} formerly known as the WonderWeb API, is a Java interface and implementation for the W3C web ontology language (OWL), and provides features similar to the Protege OWL API. However, unlike the Protege OWL API, the current development of the OWL API is geared towards OWL 2.0³² which encompasses OWL-lite, OWL-DL, as well as some elements of OWL-full. Additionally, this API ships with a selection of parsers, renderers, and interfaces to various state-of-the-art reasoners. The key advantage having the OWL API based on OWL 2.0 is that, unlike OWL 1.0 which does not make any strong suggestions about what an OWL 1.0 implementation should look like, the OWL 2.0 documentation precisely states what a "standard" OWL 2.0 implementation should look like. Consequently, there is less ambiguity with respect to how ontology constructs should be translated into a programming interface.

The LexGrid Terminology Model

The LexGrid Model is a community-driven proposal for the standard storage of controlled vocabularies and ontologies whose current development is being coordinated by the Mayo Clinic. The model defines how ontologies are formatted and represented programmatically, and is intended to be flexible enough to accurately represent a wide variety of ontologies developed in different languages and other lexically based knowledge resources. The model has been applied in the context of several different server storage mechanisms (e.g., relational databases, Lightweight Data Access Protocol—LDAP) and an XML format. It provides the core representation for all data managed and retrieved through the API, and can represent vocabularies provided in numerous source formats including the (i) Open Biomedical Ontologies (OBO), e.g., Gene Ontology,³³ (ii) Web Ontology Language (OWL), e.g., NCI Thesaurus,³⁴ (iii) Unified Medical Language System (UMLS) Rich Release Format, e.g., NCI Metathesaurus which comprises of extensive clinical terminologies such as SNOMED CT,³⁵ and (iv) Classification

Markup Language (clam liters), e.g., International Classification of Diseases version 10 (ICD-10).

As mentioned earlier, this common terminology model is a critical component of the LexGrid project. Once disparate vocabulary information can be represented in a standard model, it becomes possible to build common repositories to store vocabulary content and common programming interfaces and tools to access and manipulate that content. This is evidenced by the LexBIG API, developed for the Cancer Biomedical Informatics Grid (caBIG) and the National Center for Biomedical Ontology (NCBO) initiatives. In what follows, we discuss some of the key elements of our LexGrid model.

Key Elements of the LexGrid Model

LexGrid is based on a model-driven architecture. The master LexGrid model is maintained in XML Schema which is then transformed into UML and other representational forms for publication. The model is comprised of multiple high-level objects that form the core in uniformly representing ontologies and vocabularies. These include the following:

Service: A service represents an access point for a collection of coding schemes and/or value domains and their accompanying history and provenance information (see Fig 1). Each coding scheme in the coding schemes collection represents a snap shot of a terminological resource (code list, classification scheme, thesaurus, ontology, etc) at a particular point. Similarly, each value domain represents a snap shot of a collection of codes and pick-lists. The history collection describes release “packages”, which may consist of multiple coding schemes and value domains that have been published as a unit entity. Not all the coding schemes or value domains named in a given system release are necessarily available from the containing service. As an example, a given service may include the 2007/10/31 release of SNOMED-CT and version 2007_05E of the NCI Thesaurus, while the corresponding system release would describe the UMLS 2008 AA Release, and describe all 140 plus source vocabularies in the thesaurus itself. System releases are intended to document version dependencies between terminology resources and/or value domains.

Coding Scheme: A coding scheme represents particular version of a lexicon, code set, classification system, thesaurus, ontology, mapping or other terminological resource in a uniform and consistent fashion (see Fig 2). Each `coding-`

`scheme` carries information that uniquely identifies the terminological resource along with the relevant metadata. Such metadata include information such as the default language used in the coding scheme, the source(s) of the coding scheme, any copyright information associated with the coding scheme, and an official URI for the coding scheme (`registeredName`).

Additionally, each `codingScheme` is associated with a collection of mappings, which assign meaning to all identifiers that are used within the description of the terminological resource and its content (see Fig 5, available as an online data supplement at <http://www.jamia.org>). These mappings associate the local identifiers that are used in individual terminological resources with their intended meaning, which is represented by a Universal Resource Identifier (URI). The associated URI may represent another `codedEntry` within the current coding scheme or may reference an external terminological resource. As an example (see Table 1), the Foundational Model of Anatomy (<http://sig.biostr.washington.edu/projects/fm/>) may use “ENG” to represent text in the English language, while the NCI Thesaurus (<http://nciterns.nci.nih.gov/NCIBrowser/Dictionary.do>) might use “en” to represent the same thing. Both of these entries can be anchored to a common “meaning” via `supportedLanguage` entries:

Similarly, different resources may use different identifiers to reference coding schemes. As an example, the UMLS Metathesaurus uses the abbreviation “ICD9CM” to reference ICD-9-CM, while Health Level 7 uses “ICD9”. The `supportedCodingScheme` (see Table 2) entries show that both of these resources are referencing the same entity. Obviously, these mapping entries depend on an outside authority that can register and assign URN’s. While no such authority has yet emerged on a global scale, Health Level 7 (HL7) manages a URI registry at <http://www.hl7.org/oid/index.cfm> for health care related terminologies, the BioPortal (<http://bioontology.org>) supports a URI assignment scheme for Biomedical related ontologies, the W3C assigns URI’s to its resources, the Dublin Core (<http://dublincore.org/>) uses PURL (<http://purl.org/>) based resources, etc.

Properties: An important aspect of the LexGrid model is its notion of a “property” (see Fig 6, available as an online data supplement at <http://www.jamia.org>). The LexGrid model was designed with two primary purposes—(1) to provide a

Figure 1. LexGrid Service.

Figure 2. LexGrid Model Element: codingScheme.

common representation and semantics for terminological entities and attributes that are commonly used across many terminological resources, and (2) to faithfully represent the remainder of the terminological content in a way that remains faithful to the original resource.

As an example (see Table 3), the UMLS has a field in the MRSAB table called "SON", which carries the official name of the resource. The Dublin Core (<http://dublincore.org/>) by contrast, uses the property, `dcterms:title`, to identify exactly the same entity. The official name of the terminological resource is something that is considered semantically significant to the LexGrid model, so it is recorded in both the `formalName` attribute of the coding scheme and the prop-

erties collection. The SKOS (<http://www.w3.org/2004/02/skos/>) representation of FAO Agrovoc (http://www.fao.org/aims/ag_intro.htm), by contrast, has a property, `dcterms:issued`. The LexGrid model has no slot for this so it is only stored in the properties collection.

Note that "SON", "title" and "issued" might appear in the supportedProperty table as (see Table 4) where, the UMLS SON has arbitrarily been assigned to the Dublin Core "title" category.

Table 1 ■ Supported Language Entries

Coding Scheme	Localized	Urn
Foundational Model of Anatomy	Eng	Urn: oid: http://2.16.840.1.113883.6.84:en
NCI Thesaurus	En	Urn: oid: http://2.16.840.1.113883.6.84:en

NCI = National Cancer Institute.

Table 2 ■ SupportedCodingScheme Entries

Containing Coding Scheme	Localized	Urn
UMLS Metathesaurus	ICD9CM	Urn: oid: http://2.16.840.1.113883.6.2
HL7 version 2 tables	ICD9*	Urn: oid: http://2.16.840.1.113883.6.2

UMLS = Unified Medical Language System; HL7 = Health Level 7.
 *The authors are aware that the WHO published the original ICD9, distinct from the American CM modification. This is a legacy designation within the HL7 V2 table namespace.

Table 3 ■ LexGrid Property Examples

Resource	Value	CodingScheme full Name	Property propertyName	Property propertyValue	Property lexGridName
MRSAB SON	International Health Terminology Standards Development Organisation, SNOMED Clinical Terms.	International Health Terminology Standards Development Organisation, SNOMED Clinical Terms.	SON	International Health Terminology Standards Development Organisation, SNOMED Clinical Terms.	full name
dc: title	SKOS Core Vocabulary 2006–04– 18 ^{2nd} W3C Public working draft (amended) edition	SKOS Core Vocabulary 2006–04– 18 ^{2nd} W3C Public working draft (amended) edition	title	SKOS Core Vocabulary 2006– 04–18 ^{2nd} W3C Public working draft (amended) edition	full name
dct: issued	2006/4/18		issued	2006/4/18	

SNOMED = systematized nomenclature of medicine; SKOS = simple knowledge organization systems.

CodedEntry. A typical `codingScheme` defines a collection of coded entries (see Fig 7, available as an online data supplement at <http://www.jamia.org>). A `codedEntry` has the following characteristics:

1. Every `codedEntry` is associated with a code or an identifier that is unique within the namespace of the containing coding scheme.
2. Each `codedEntry` code is intended to represent a class, category, individual or association within the context of the containing coding scheme.

In addition to defining coded entries, a coding scheme may also add information to coded entries drawn from other coding schemes.

Every coded entry must have an `entryCode` that is unique within the containing coding scheme. The containing coding scheme defaults to the coding scheme in the model unless the `entryCodingScheme` property is specified. This property allows coding schemes to refer to or further define codes drawn from outside schemes. Every coded entry must also have at least one presentation, which is usually a word or phrase that conveys the intended meaning of the code. In addition, coded entries may have multiple definitions, comments and/or instructions about the when, why and how the coded entry is intended to be used. A `codedEntry` is associated with two status fields: `isActive`, which states whether this entry should be returned during textual or relation based searches (vs. by `entryCode`) and `entryStatus`, which identifies the status of the entry according to the original provider. One of the steps involved in mapping a terminology into the LexGrid model is determining which `entryStatus` are considered active.

Coded entries can be further categorized depending on the use case. Terminologies represented in OWL DL 1.0 partition coded entries into the categories “Class” (any subclass of `rdfs:Class` or any type of `owl:Class`), “Association” (any subclass of `rdf:Property`) and “Individual” (any type of `owl:Class`).

For historical reasons, the elements in this partition are labeled as `concept`, `association` and `instance`, respectively (see Fig. 3). The `concept` and `association` sub-

classes of coded entry identify additional semantics that are specific to that type.

Concepts: In the context of the LexGrid model, a `concept` defines a group of individuals that belong together because they share some properties or characteristics (see Fig 8, available as an online data supplement at <http://www.jamia.org>). Each `concept` must have an `entryCode` in the model, although if the `entryCode` was internally assigned and was not explicitly asserted in the ontology itself, the `isAnonymous` flag in the `concept` is set to `true`.

Instances: In addition to the `concepts`, each `codingScheme` may define their members (see Fig 9, available as an online data supplement at <http://www.jamia.org>). We normally think of these as individuals in our universe of things, and represent them by defining one or more `instances` containers. Each `instance` is unique within the code system that defines it, and is a member of one, and only one, `concept`. Similar to a `concept`, each `instance` of the `concept` must have an `entryCode` in the model. For example, Bill Clinton and George W. Bush are both members of the `concept Person`.

Additionally, it is possible to specify relations between `concepts` and `instances` via `associations`, as described next.

Relations: Each code system may define one or more containers to encapsulate relationships between `concepts` and `instances`. Each named relationship (e.g., “`subClassOf`”, “`Part of`”, “`Same as`”) is represented as an `association` within the LexGrid model (see Figs 10 and 11, available as an online data supplement at <http://www.jamia.org>). Each `relations` container must define one or more `association`. The `association` definition may also further define the nature of the relationship in terms of transitivity, symmetry, reflexivity, forward and inverse names, etc. Multiple in-

Table 4 ■ supportedProperty Entries

Localized	Urn
SON	http://purl.org/dc/elements/1.1/title
Title	http://purl.org/dc/elements/1.1/title
Issued	http://purl.org/dc/terms/issued

Figure 3. LexGrid codedEntry partition.

stances of each association can be defined, each of which provide a directed relationship between one source and one or more target concepts or instances. Additionally, the model allows specifying relationships between the associations themselves (e.g., `rdfs:subPropertyOf`).

The source (domain) and target (range) of each association may be contained in the same code system as the association or another if explicitly identified, and can be set to a concept, or an instance, or an association, thus in essence, providing the capability to specify relationships between two concepts, instances, associations, or any combination of these entities. For handling OWL data type ranges (e.g., `xsd:string`) which also includes an attribute for storing the data values (e.g., AIDS). Finally, the model also allows to specify qualifiers such as `owl:someValuesFrom` or `owl:allValuesFrom` on the associations via the `associationQualifier`.

Available Representations

The master representation of the LexGrid model is provided in XML Schema Definition (XSD) format and can be accessed from <http://informatics.mayo.edu/schema/2008/01/LexGrid/>.

Conversions to other formal representations are available, including XML Metadata Interchange (XMI) and Unified Modeling Language (UML). Implementation or technology-specific renderings of the model also exist. These include relational database schema (MySQL, PostgreSQL, DB2, Oracle, etc) and Lightweight Directory Access Protocol (LDAP)

schema. Programming interfaces generated from the formal representation include Java bean interfaces based on the Eclipse Modeling Framework (EMF) and Castor frameworks.

LexGrid Architecture and Implementation

This section provides a basic description of the building blocks, interfaces, and tools comprising the LexGrid project (see Fig 4).

A *LexGrid Node* represents both software and a backing data store that provide a logical persistence layer for accessing,

Figure 4. LexGrid Architectural Elements.

storing and managing vocabulary content according to a formalized information model described in the previous section. LexGrid nodes typically use relational database management systems for management of data and indexing functions. Implementations include but are not limited to MySQL, PostgreSQL, DB2, Oracle, Hypersonic, and LDAP/BDB. As part of on-going work, we are exploring the possibility of having a RDF-based triple store as a backend for LexGrid.

The *Import Toolkit* provides an API and a set of administration tools to load, index, publish, and manage vocabulary content for the vocabulary server. Various standard formats and models that are supported by the loaders of the toolkit include Rich Release Format (RRF), Web Ontology Language (OWL), Protege frames, LexGrid XML, Text Delimited, Open Biomedical Ontology (OBO), UMLS Semantic Net, and Classification Markup Language (ClAML). Specifically, a loader for a particular format “maps” various elements of the format to the elements of the LexGrid model. For example, in the case of the OWL loader, elements such as `owl:Class` maps to the LexGrid element `concept`, `owl:ObjectProperty` maps to `association`, and so on. Additionally, during ontology loading, various optimizations such as content caching are done to improve performance. Once the ontologies are loaded, indexes (based on Apache Lucene) are created to enhance performance and scalability of operations to search terms and concepts defined in the ontologies. Indexes are built by extracting indexable tokens out of the ontology elements (e.g., concept names, concept definitions), and applying traditional analytics such as skipping stop words (i.e., frequently used words that do not help distinguish one document from the other, such as “a”, “an”, “the”, “in”, “on”, etc), converting all tokens to lowercase letters, so that searches are not case-sensitive, and so on. Once an index is created for a `codingScheme`, it is stored in a single directory in the file system.

The *Export Toolkit* provides an API and set of administration tools to export content in a standard format from a LexGrid node. Standard formats provided for export include LexGrid XML, Open Biomedical Ontology (OBO), Protege frames, and Web Ontology Language (OWL). Obviously, in some scenarios, it is feasible to leverage external translators for doing format conversions.¹³

LexGrid also provides a *Vocabulary Browser* that gives the ability to connect with and search LexGrid repositories via the API, whereas the *LexGrid Editor* is a basic editor for creating, modifying, and changing vocabulary content according to the LexGrid information model. The editor is an Eclipse-based application that supports multivocabulary query and browsing, interactive views, logging and auditing. Note that both the LexGrid browser and editor were created as proof-of-concept efforts and are not a focus of active development.

- *LexGrid services* provide an access layer to search and access vocabulary content. Specific software packages such as LexBIG have been implemented to access LexGrid content, which we discuss in the following.

LexGrid Deployment Projects

In this section, we mention a few projects that leverage the LexGrid model and infrastructure.

The LexBIG Project

LexBIG (<https://cabig-kc.nci.nih.gov/Vocab/KC/index.php/LexBIG>) is a project that applies the LexGrid vision and technologies to the requirements of the Cancer Biomedical Informatics Grid (caBIG) Community. The goal of the project is to build a vocabulary server accessed through a well-structured application programming interface (API) capable of accessing and distributing vocabularies as commodity resources. The major components of LexBIG include:

- A set of service management programs to load, index, publish, and manage content for the vocabulary server.
- An Application Programming Interface providing Java interfaces to various functions including lexical queries, graph representation and traversal, and NCI change event history.
- A graphical user interface providing access to administrative and API functions.
- Documentation consisting of API JavaDocs, administrator and programmer guides.
- Numerous examples providing sample source code for common vocabulary queries.
- A test suite to validate the software package installation.

Cancer Centers can use the LexBIG package to install the NCI Thesaurus and NCI Metathesaurus, and other content and perform queries via a rich API. LexBIG services can also be incorporated into numerous applications wherever vocabulary content is needed.

The LexBIO Project

LexBIO (<http://informatics.mayo.edu/LexGrid/index.php?page=lexbio>) provides development and support of LexGrid-based terminology software (services, persistence, tooling) to the National Center for Biomedical Ontology (NCBO; <http://bioontology.org>). Deliverables focus on providing administration and access to vocabularies required by NCBO driving biological projects and associated use cases. Implementation shares a technology base with the LexBIG project, including Java API, repositories, and incorporation of the LexGrid model architecture. This base is then enhanced and extended to meet specific needs of the Center.

The LexWiki Project

LexWiki (<https://cabig-kc.nci.nih.gov/Vocab/KC/index.php/LexWiki>) is a project that is being introduced and applied within the biomedical informatics community for collaborative terminology authoring and curation tasks. The system comprises of three main modules: (i) a lightweight editor for proposal creation and consensus building based on the Semantic MediaWiki (SMW) platform,³⁶ (ii) A formal ontology editor for change commitment, formalism rendering and consistency checking based on Collaborative Protege,³¹ and (iii) a terminology service module based on the LexGrid terminology model and the LexBIG API. The key aspect of LexWiki is to provide a common mechanism for representation of the ontological content, and this is achieved via “templates” which model the core semantics of the LexWiki contents. The data elements are derived from the LexGrid terminology model, defining the semantic core of LexWiki,

so that different templates can be used as long as they are mapped to the LexGrid model.

At this writing, the LexWiki infrastructure is adopted for two large-scale projects: World Health Organization (WHO) International Classification of Diseases (ICD) version 11 revision project, and the National Cancer Institute (NCI) Biomedical Grid Terminology (BiomedGT). In the WHO ICD-11 revision project, developing a structured, Wiki-like tool to create the drafts of ICD-11 will allow broader participation of multiple stakeholders. BiomedGT is an open, collaboratively developed terminology for translational research, sponsored by NCI. Initial content is based on the NCI Thesaurus, but the goal is to evolve BiomedGT into a set of subterminologies that can be federated, with content maintained by experts in the relevant research communities.

Dissemination

The tools and technologies described in this manuscript are open-source and available for download. The following are some of the relevant Web sites:

- LexBIG: <https://cabig-kc.nci.nih.gov/Vocab/KC/index.php/LexBIG>
 - i. Downloadable software distributions: https://gforge.nci.nih.gov/frs/?group_id=491
 - ii. Administrator guide: <https://gforge.nci.nih.gov/frs/download.php/4509/LexBIG-install-admin-guide-2.3.0rc1.pdf>
 - iii. Programmer guide: <https://gforge.nci.nih.gov/frs/download.php/4508/LexBIG-programmer-guide-2.3.0rc1.pdf>
 - iv. EVS technical Guide (documents LexBIG use within the context of NCI Enterprise Vocabulary Services): http://gforge.nci.nih.gov/docman/view.php/491/12758/EVS_API_4-1_TechnicalGuide.pdf
- LexBIO: <http://informatics.mayo.edu/LexGrid/index.php?page=lexbio>
- LexWiki: <https://cabig-kc.nci.nih.gov/Vocab/KC/index.php/LexWiki>

Information about any of these projects, and other relevant information can be obtained via: informatics@mayo.edu.

Furthermore, Mayo Clinic has been designated to be one of the caBIG Vocabulary Knowledge Centers. This center is expected to serve as the repositories and resources for the knowledge associated within their designated area of focus (e.g., biomedical informatics), and as the steward of the tools and documentation, providing access and support to those individuals and institutions interested in making use of or extending caBIG tools. More information is available at: https://cabig-kc.nci.nih.gov/Vocab/KC/index.php/Main_Page.

Discussion

In the previous sections, we have illustrated some of the important aspects of the LexGrid terminology model and services provided by the APIs. An important feature that we plan to incorporate within the LexGrid infrastructure is the ability to reason with ontologies. Generically speaking, reasoning is considered as the cognitive process of looking for reasons to belief, conclusions, actions or feelings. In the

context of ontology reasoning, this implies using a software tool, called a reasoner (e.g., Pellet,³⁷ Racer,³⁸ FaCT++³⁹), to retrieve answers to certain queries, identify logical inconsistencies, integrate and align multiple ontologies and vocabularies, among others. Thus, to provide reasoning services within LexBIG, our objective is to enhance the existing LexBIG API such that it can leverage existing off-the-shelf open-source reasoners such as Pellet.

At present, the LexGrid model and infrastructure enables representation of relationships (and querying) across multiple-ontologies, however such relationships are required to be prespecified when the ontologies are loaded. Consequently, another important requirement we intend to address is to provide semi-automatic support for alignment between ontologies. Past research has developed tools and techniques that leverage an “mediating ontology” for defining the mappings between “source” and “target” ontologies.^{40–42} Although useful, the requirement to define mappings to mediating ontology from the source/target ontologies is a labor-intensive and cumbersome process. To address this limitation, we plan to investigate a combination of lexical and semantic based semi-automatic techniques for identifying interontology mappings.^{43–45}

Finally, some of the features under investigation or active development include the following:

- Federated vocabulary access, registration, and discovery (actively developed as part of NCI Enterprise Vocabulary Services for caBIG as a grid-level implementation of the LexBIG API).
- API extensions to support local vocabulary extensions and provider suggestions.
- API extensions to support HL7/CTS 2 API (being defined).
- API extensions to support submission of vocabulary change requests.
- Integration of the LexGrid model with the ISO 11179 specification.

Conclusion

LexGrid is an evolving set of software for supporting vocabulary distribution and access. The overarching goal of the project is to accommodate multiple vocabulary and ontology distribution formats and support of multiple data stores for federated vocabulary distribution, and provide a consistent and standardized rich API to access multiple vocabularies that support lexical search queries as well as traversal of the ontology itself. The current LexGrid implementation is open-source and is actively deployed in the development of Cancer Biomedical Informatics Grid (caBIG) and the National Center for Biomedical Ontology (NCBO).

References ■

1. Bodenreider O. Biomedical ontologies in action: Role in knowledge management, data integration and decision support. *Yearb Med Inform.* 2008;67–79.
2. Chute CG. The Copernican era of health care terminology: A Re-centering of health information systems. *Proc AMIA Symp.* 1998;68–73.
3. Chute CG. Clinical Classification and terminology: Some history and current observations. *J Am Med Inform Assoc* 2000;7(3): 298–303.

4. OBO. Open biomedical ontologies. Available at: <http://obo.sourceforge.net>. Accessed 3/29/09.
5. Antoniou G, van Harmelen F. Web ontology language: Owl. In: Handbook on Ontologies in Information Systems. Edited by Staab S, Studer R: Springer-Verlag, 2003.
6. Borgida A, Serafini L. Distributed description logics: Assimilating information from peer sources. *J Data Semant* 2003; LNCS 2800:153–184.
7. Grau B, Parsia B, Sirin E. Combining OWL ontologies using E-connections. *J Web Semant* 2006;4(1):40–59.
8. Bao J, Caragea D, Honavar V. On the semantics of linking and importing in modular ontologies. In: International Semantic Web Conference. LNCS;4273, Springer-Verlag; 2006.
9. Wang Y, Bao J, Haase P, Qi G. Evaluating formalisms for modular ontologies in distributed information systems. In: 1st International Conference on Web Reasoning and Rule Systems. LNCS;4524, Springer-Verlag; 2007.
10. Grau B, Horrocks I, Kazakov Y, Sattler U. Modular reuse of ontologies: Theory and practice. *J Artif Intell* 2008;31:273–318.
11. Day-Richter J, Harris M, Haendel M, Lewis S. OBO-Edit An ontology. In: Editor for Biologists, BioInform, 2007;23(16):2198–200.
12. Knublauch H, Ferguson R, Noy N, Musen M. The protege: OWL plugging: An open development environment for semantic web applications. In: International Semantic Web Conference. LNCS;3298, Springer-Verlag; 2004.
13. Moreira D, Musen M. OBO to Owl: A Protege OWL Tab to Read/Save OBO Ontologies, BioInform 2007;23(14):1868–70.
14. Bechhofer S, Volz R, Lord P. Cooking the semantic web with the OWL API. In: International Semantic Web Conference. LNCS; 2870, Springer-Verlag; 2003.
15. Horridge M, Bechhofer S, Noppens S. Igniting the OWL 1.1 touch paper: The OWL API. In: 3rd OWL Experienced and Directions Workshop, 2007.
16. Chute CG, Elkin P, Sherertz D, Tuttle M. Desiderata for a clinical terminology server. In: AMIA Annual Symposium, 1999.
17. Solbrig H, Chute CG. Terminology access methods leveraging LDAP resources. In: 11th World Congress on Medical Informatics (Medinfo), IOS Press, 2004.
18. Nelson S, Sherertz D, Tuttle M, Erlbaum M. Using MetaCard: A HyperCard browser for biomedical knowledge sources. In: 14th Annual Symposium on Computer Applications in Medical Care. IEEE CS Press, 1990.
19. Chute CG, Yang Y, Tuttle M, et al. A preliminary evaluation of the UMLS metathesaurus for patient Record classification. In: 14th Annual Symposium on Computer Applications in Medical Care. IEEE CS Press, 1990.
20. Chute CG, Atkin G, Ihrke D. An empirical evaluation of concept capture by clinical classifications. In: 7th World Congress on Medical Informatics (Medinfo), IOS Press, 1992.
21. Chute CG, Yang Y. An overview of statistical methods for the classification and retrieval of patient events. *Methods Inf Med* 1995;34(1–2):104–10.
22. Tuttle M, Olson N, Keck K, et al. Metaphrase: An aid to the clinical conceptualization and formalization of patient problems in health care enterprises. *Methods Inf Med* 1998; 37(4–5):373–83.
23. Rector A, Rogers J, Zanstra P, vander hE. OpenGALEN: Open source medical terminology and tools. In: AMIA Annual Symposium, 2003.
24. Lexicon Query Services. Available at: http://www.omg.org/technology/documents/formal/lexicon_query_service.htm. Accessed 3/23/09.
25. Apelon. Distributed terminology server (DTS). Available at: <http://apelon-dts.sourceforge.net/>. Accessed 3/23/09.
26. UMLS Knowledge Source Server (UMLSKS). Available at <http://umlsks.nlm.nih.gov/>. Accessed 3/23/09.
27. Berners-Lee T, Hendler J, Lassila O. The semantic web. *Scientific American*. 2001. Available at <http://www.sciam.com/article.cfm?id=the=semantic-web/>. Accessed 3/23/09.
28. Resource description framework, RDF. Available at: <http://www.w3.org/RDF/>. Accessed 3/23/09.
29. SPARQL Query Language for RDF. Available at: <http://www.w3.org/TR/rdf-sparql-query/>. Accessed 3/23/09.
30. Gleaning Resource Descriptions from Dialects of Languages (GRDDL). Available at: <http://www.w3.org/TR/grddl/>. Accessed 3/23/09.
31. Noy N, Chugh A, Liu W, Musen M. A framework for ontology evolution in collaborative environments. In: 5th International Semantic Web Conference. LNCS;4273, Springer-Verlag; 2006.
32. OWL. 2 Web ontology language. Available at: <http://www.w3.org/2007/OWL/>. Accessed 3/23/09.
33. Bada M, Stevens R, Goble C, et al. A short study on the success of gene ontology. *J Web Semantics* 2003;1(2):235–40.
34. Noy N, de Cornado S, Solbrig H, et al. Representing the NCI thesaurus in OWL DL: Modeling tools that help modeling languages. *Appl Ontology* 2008.
35. Systematized Nomenclature of Medicine-Clinical Terms (SNOMED CT). Available at: <http://www.snomed.org>. Accessed 3/23/09.
36. Krotzsch M, Vrandečić D, Volkel M, Haller H, Studer R. Semantic Wikipedia. *J Web Semantics* 2007;5(4):251–61.
37. Sirin E, Parsia B, Grau B, Kalyanpur A, Katz A. Pellet: A practical OWL-DL reasoner. *J Web Semantics* 2007;5(2):51–3.
38. Haarslev V, Maoller R. 1st International Joint Conference on Automated Reasoning, ed. RACER System Description, LNAI, p 2083, Springer-Verlag; 2001.
39. Tsarkov D, Horrocks I. FaCT++ Description Logic Reasoner: System description. In: International Joint Conference on Automated Reasoning LNAI 4130, Springer-Verlag; 2006.
40. Fung K, Bodenreider O. Utilizing the UMLS for semantic mapping between terminologies. In: AMIA Annual Symposium, 2005.
41. Fung K, Bodenreider O, Aronson A, Hole W, Srinivasan S. Combining lexical and semantic methods of inter-terminology mapping using the UMLS. In: 12th World Congress on Health (Medical) Informatics (Medinfo), IOS Press, 2007.
42. Barrows R, Cimino J, Clayton P. Mapping clinically useful terminology to a controlled medical vocabulary. *Annu Symposium on Comput Appl in Med Care*. 1995.
43. Lambrix P, Tan H. Sambo: A system for aligning and merging biomedical ontologies. *Web Semantics Sci Serv Agents World Wide Web* 2006;4(3):196–206.
44. Wiseman F, Roos N. Domain independent learning of ontology mappings. In: International Joint Conference on Autonomous Agents and Multiagent Systems, 2004.
45. Wang Y, Patrick J, Miller G, O'Halloran J. Linguistic mapping of terminologies to SNOMED-CT. In: Semantic Mining Conference on SNOMED CT, 2006.