CTS2 Introduction and Overview

Introduction to the Common Terminology Services 2 Standard

Scott Bauer, Cory Endle

HL7 Working Group Meeting
May 7th, 2013

Purpose and Outline

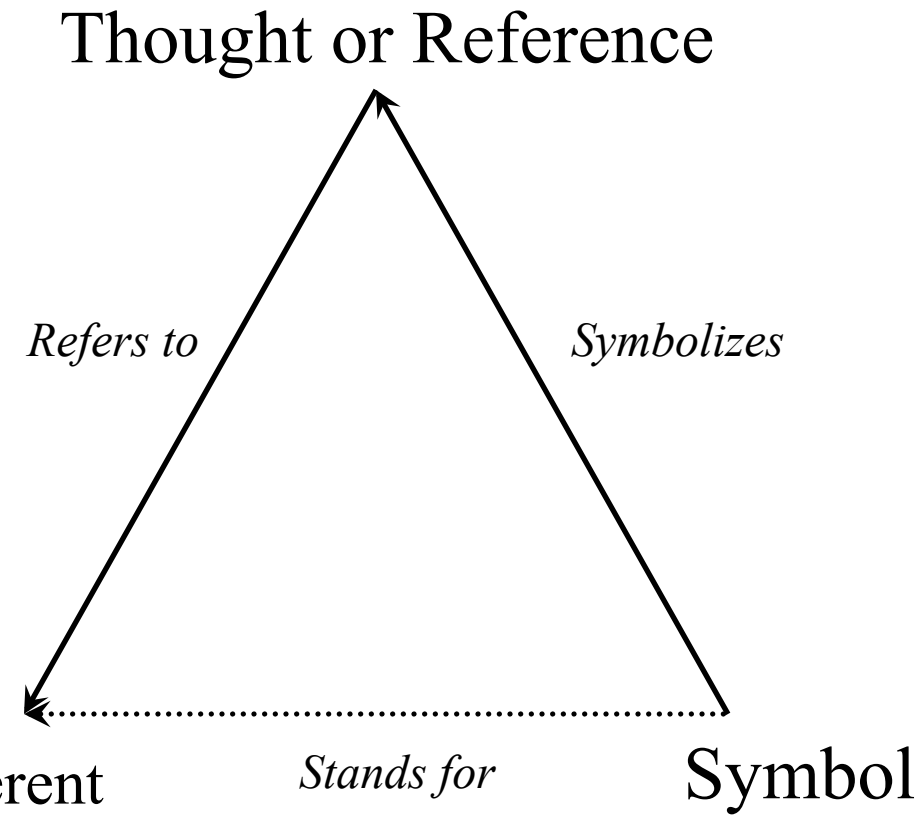
- Provide a background to Terminology Services
- Outline the process and history culminating in the completion of the specification
- Describe the artifacts of this process and their purpose
- Discuss the architecture of the API and how it can be implemented
- Report on and demonstrate current implementations

Communication is about Language

Language - a "specification" that enables communication

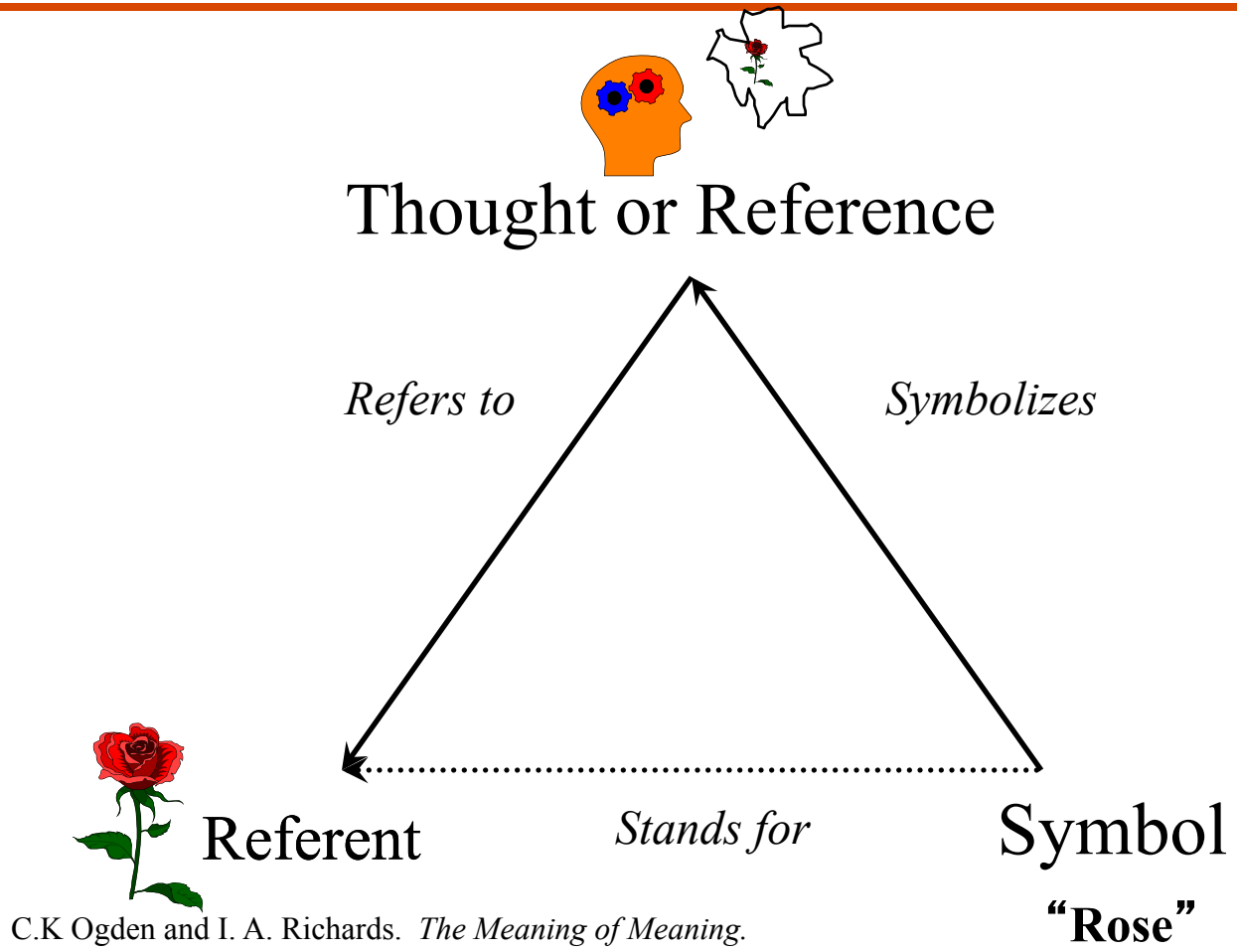
- Semantics - the association between signs or symbols and their intended "meaning"
- Syntax - the rules for ordering and structuring the signs into phrases and sentences
- Pragmatics - the relationship between signs and symbols and the recipient. Broadly, the shared *context*.

The Semiotic Triangle



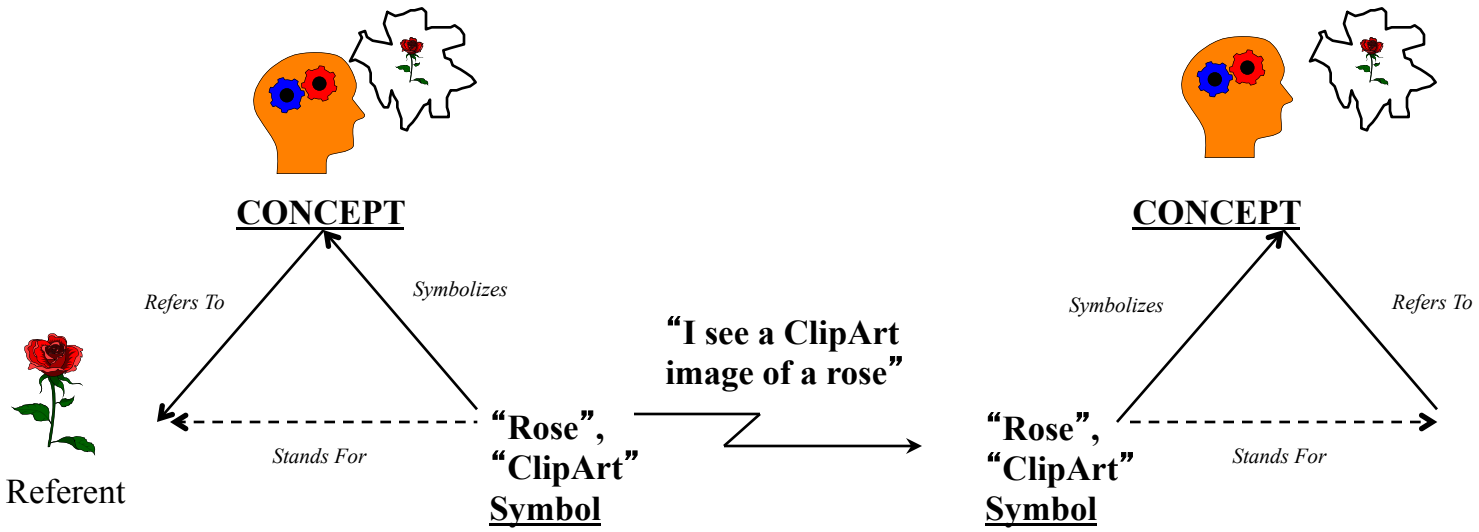
C.K Ogden and I. A. Richards. *The Meaning of Meaning.*

The Semiotic Triangle

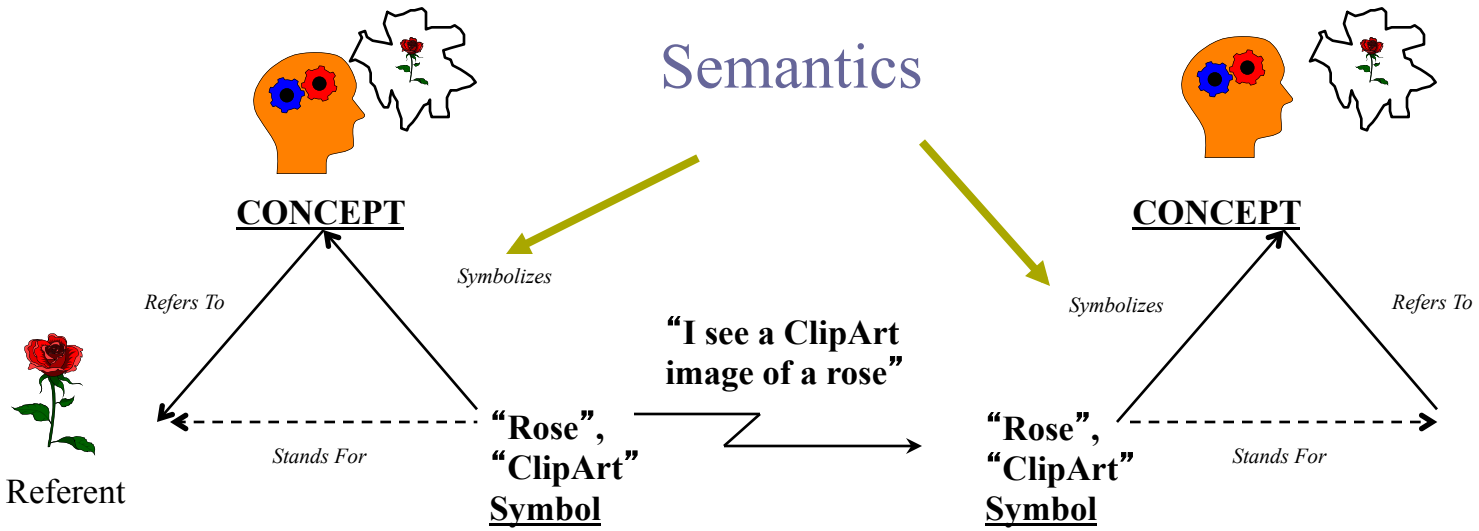


C.K Ogden and I. A. Richards. *The Meaning of Meaning.*

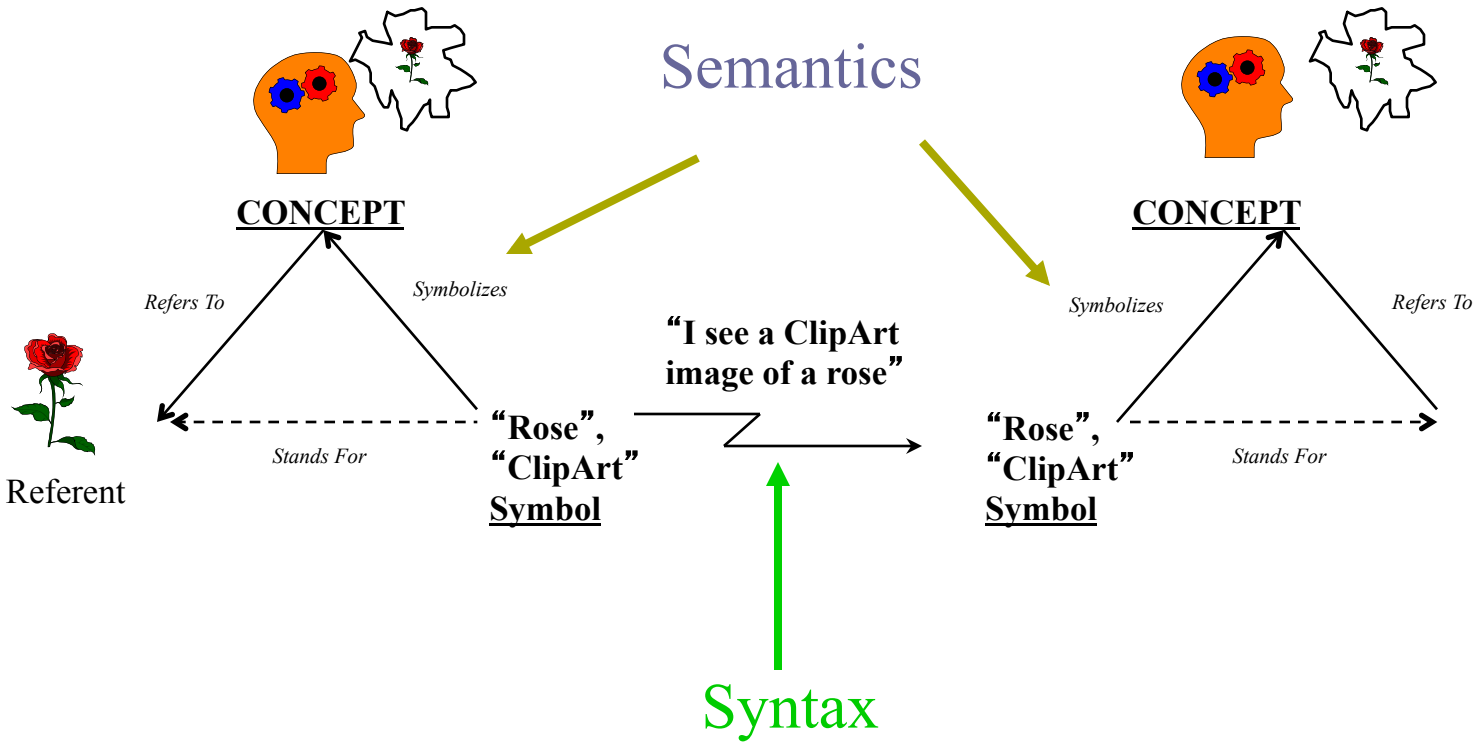
The Communication Process



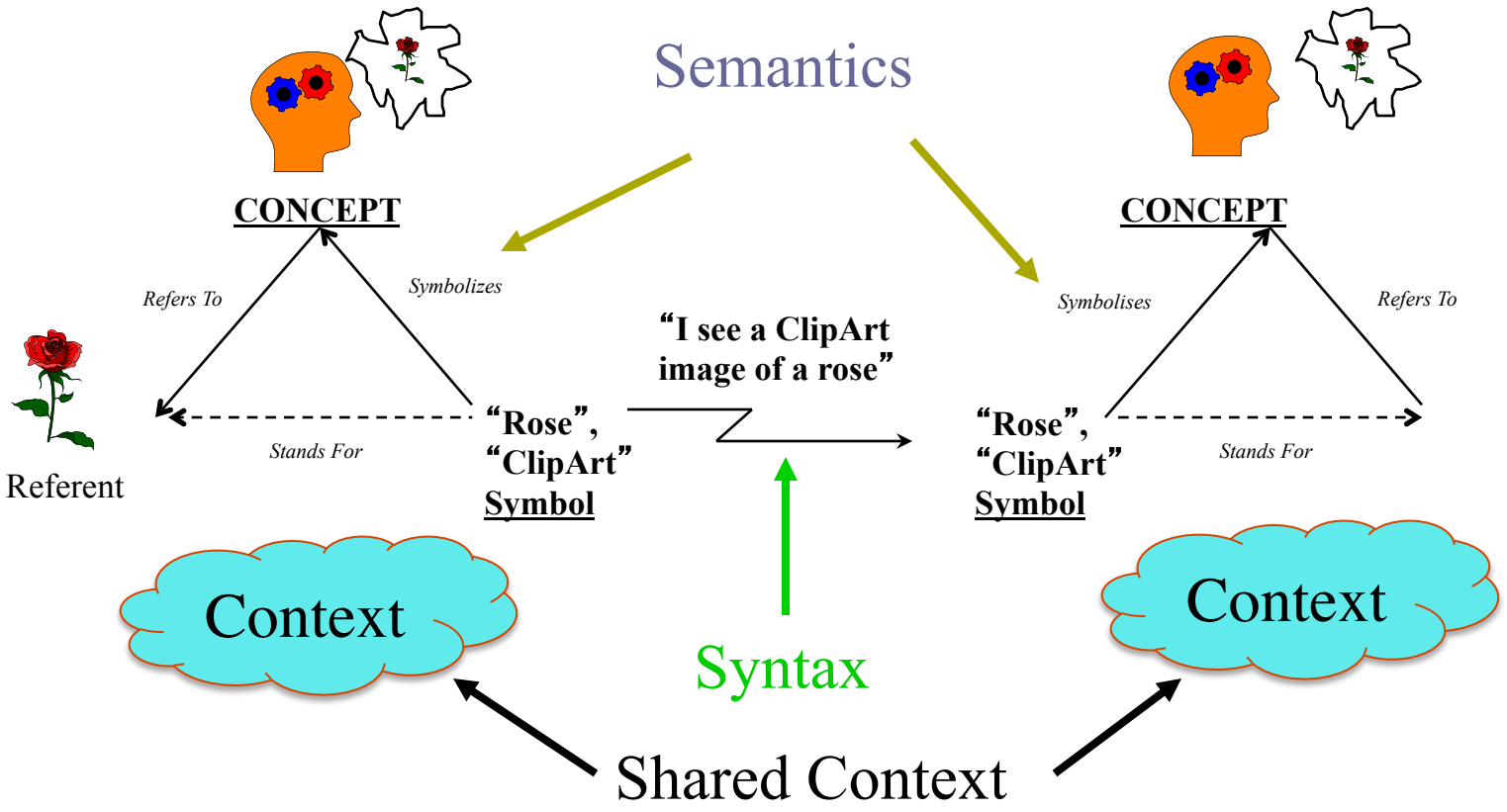
The Communication Process



The Communication Process



The Communication Process



Common Forms of Contextual Formalism

- Dictionaries
- Thesauri
- Textbooks, college courses, etc.
- Operations manuals
- Data dictionaries
- … Ontologies (recent)

AKA… Terminology

- Dictionaries
- Thesauri
- Textbooks, college courses, etc.
- Operations manuals
- Data dictionaries
- … Ontologies (new)

"Terminology"

A generic "term" that covers a broad spectrum:

- Code / Value pairs – Country codes, airline codes, fare codes, ...
- Controlled vocabularies – when United Airlines uses "confirmed reservation", they mean "..."
- Thesauri – if you're are looking for information on the Tiger Swallowtail, you might also look at books with butterfly pictures, books about insect migration

“Terminology” (continued)

- Classifications – “buckets” for various statistical and information gathering purposes. Example: ICD-9-CM – 250.01 “Diabetes Mellitus with complication, type 1, not stated as uncontrolled”
- Formal Logic Systems – Web Ontology Language (OWL), OBO, Common Logic, ... Classes, individuals relationships. SNOMED-CT, Gene Ontology, FMA, CYC, SUMO, ...

Why Terminology Services?

Start with a simple example – the ISO Country Codes:

- Where can you get them?
- What formats are they available in?
- How are they organized? How do I list just EU countries? Countries in Africa?
- Oh – they change???? How do I know when a change occurs? How do I know what happened?

Why Terminology Services?

- Ok – I've downloaded them...
- I create a set of SQL tables...
- ... construct some queries to create a dropdown list for a form
- ... write an Ajax application for my web page
- ... write some more queries to print the full names on a report

Why Terminology Services?

- Discover the accounting package needs three character codes
- Repeat the download and write a conversion
- Discover the accounting package hasn't updated their tables since 1990... - what do we do with "YU"
- Discover ISO *reuses* codes (!!!)

Why Terminology Services?

- We've just been bought... our new owner doesn't use ISO codes
- ... we've got to start translating between ISO codes and 01-299 codes
- ... and they've got "YU"...
- ... and they now need to know what currencies are (and were) used in a given country...

Why Terminology Services?

- Take the scenario above and figure that a reasonably complex business such as a medical practice may have to deal with
- .... *hundreds* of tables / classifications / relationships each with its own formats, labels, release cycles, rules
- ... *many* applications that define, create and consume these terminologies

Why Terminology Services?

- ... other businesses with the same information but different codes and formats
- .... An ever increasing load of reporting / analysis / billing requirements ...

Why Terminology Services?
Terminology Service Requirements

At some point, it begins to make sense to try to come up with:

- A catalog of terminologies
- Who maintains them, what they are for, how often they are updated, what they cost, ...
- A simple way to download, deploy and update their content

Why Terminology Services?
Terminology Service Requirements

A way to ask similar queries in the same way:

- By textual content – “All currencies that use the ‘$’ symbol”
- By organization – “All EU countries”
- By relationship – “All countries that share a border with France”
- By secondary set – “All countries that are HL7 members”

Why Terminology Services?
Terminology Service Requirements

- Reasoning
- Maps between terminologies
- Incremental updates
- Local extensions
- “Binding” to data elements
- Different output formats

Why Terminology Services?
Summary

- SOA and information exchange require shared semantics (terminology)
- Shared semantics, in turn, requires shared terminology
- No one can afford to do their own research, write their own importers, access methods, Ajax widgets, update mechanisms, etc.

Why Terminology Services?
Summary

Terminology Service *Standards* provide:

- Semantics for terminology
- Shared exchange models
- Complete and robust requirements
- Common API's

TERMINOLOGY SERVICES
A SHORT HISTORY

Common Terminology Services
A Short History: OMG and CTS

CTS2 Derives from:

- OMG Lexical Query Service (LQS) Standard (1999)
  - OO Model, read only, but laid most of the foundation for subsequent efforts
- HL7 CTS Specification (2004)
  - ANSI and ISO Standard
  - SOA Model, read only, reduced scope from LQS

Common Terminology Services
A Short History: CTS

Developed to meet HL7 requirements:

- LQS was deemed too big to implement
- Designed to be read only
- Versioning and revisions not included in the specification

Common Terminology Services
A Short History: CTS

Formal Specification in:

- UML model
- OMG IDL language

This was released prior to a complete Model Driven Architecture standard

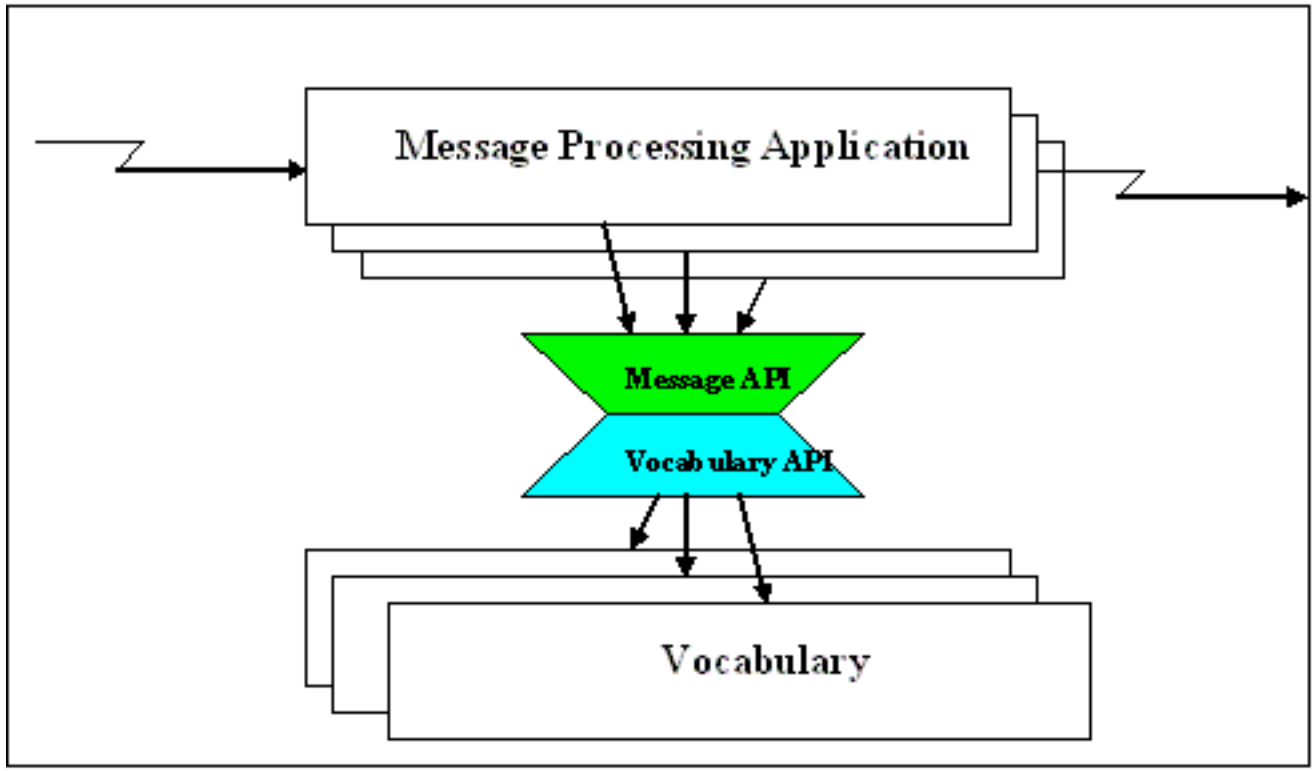
- Needed to get from UML to Java
- Apache Axis produced a WSDL and Java Beans were derived from that WSDL

Common Terminology Services
A Short History: CTS

The specification was designed in two distinct layers

- A message API (Application Programming Interface)
- A vocabulary API

Common Terminology Services
A Short History: CTS



Common Terminology Services
A Short History: CTS

Vocabulary Translation Layer

- Code map (minimal one-to-one code system level map)

Common Terminology Services 2
A Short History: CTS2

CTS2 Derives from (continued):

- Mayo LexGrid information model (2002 – present and also derived from the Lexical Query Service)
- National Cancer Institute NCI LexEVS (2005 – present)
- HL7 Service Functional Model (SFM)
- Work and insights of *many* individuals and organizations...

Common Terminology Services 2
Approach

Developed via the Healthcare Services Specification Program or HSSP process:

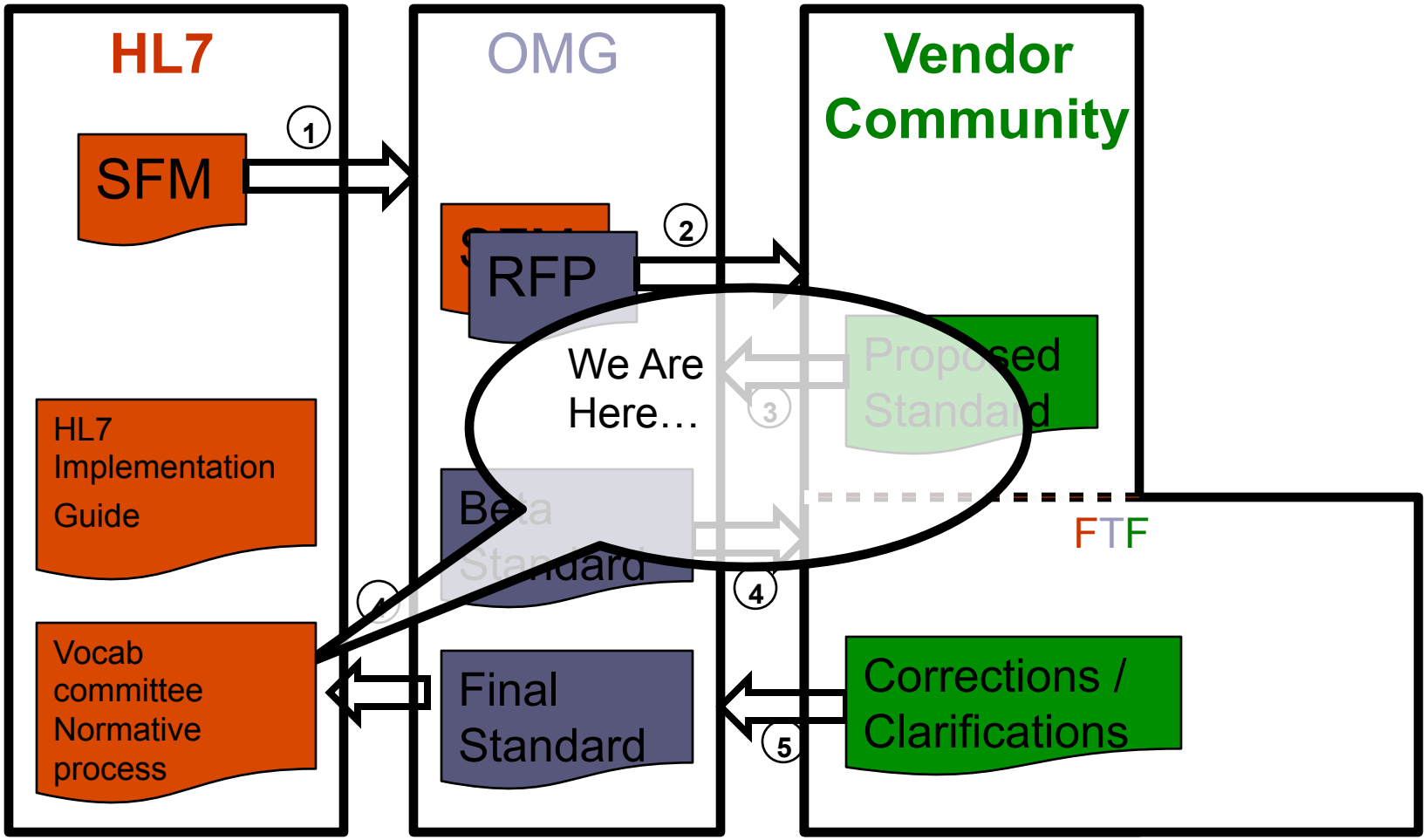
- HL7 issued the starting requirements document (HL7 SFM)
- Requirements transferred to Object Management Group (OMG), which issued a Request for Proposal through the Ontology Platform Committee
- Participants drawn from W3C, Healthcare, XML, Space, Finance and other communities

Common Terminology Services 2
Approach

Developed via the HSSP process (continued)

- Specification submitted to OMG and balloted June 2011. Issued as a “Beta” spec
- Implementations developed, issues submitted to CTS2 Finalization Task Force
- Accepted for adoption by the OMG in August 2012 – available on OMG web site

Healthcare Services Specification Project Approach: (HSSP) Workflow



CTS2 Service Functional Model (SFM)
Approach: SFM

- SFM insured that the implementation was complete
- Describes use cases and functionality for the specification
- Balloted as a draft standard for trial use (DSTU)
- Turned over to OMG where it was used to create a Platform Independent Model or PIM and a Platform Specific Model or PSM.
- Now moving toward a normative specification in the vocabulary committee

CTS2 PIM, Platform Independent Model
Approach: The role of the PIM

- A Unified Modeling Language (UML) representation:
- Defines names, purpose and how used
- No physical rendering (i.e. Java Beans, C++ classes, XML schema, etc.)
- The PIM can be transformed into multiple different Platform Specific Models
- This is where you go when you have detailed questions about the specification

CTS2 PIM, Platform Independent Model
Approach: The role of the PIM

Comprised of an:

- Information Model
  - Core
  - CTS2 Resources
- Computational Model
  - Core
  - CTS2 Operations

CTS2 PSM: Platform Specific Model
Approach: The Role of the PSM

- CTS2 representations in
  - XML schema, allowing content to be created in XML (A PSM of the Information Model)
  - WADL: a REST interface to the the computational and information model
  - WSDL: a SOAP interface to the same computational and information model

CTS2 Specification

A model and specification for discovering, accessing, distributing and updating *terminological resources* on the internet.

CTS2 Specification

The CTS2 specification provides:

- A model and set of symbols for discussing terminology:
  - Formal in UML
  - Practical in XML Schema
- A *RESTful* Architecture for querying, exchanging and updating terminological content

CTS2 Specification
Resource-Oriented Architecture

It's just four concepts:

- Resources
- Their names (URIs)
- Their representations
- The links between them

And four properties:

- Addressability
- Statelessness
- Connectedness
- A “uniform interface”

RESTFul Web Services – Richardson and Ruby, 2007

CTS2 Specification
Why REST?

- Enables “Linear Value Proposition”
  - Easy things should be easy
  - Cost of adding complexity should be linear

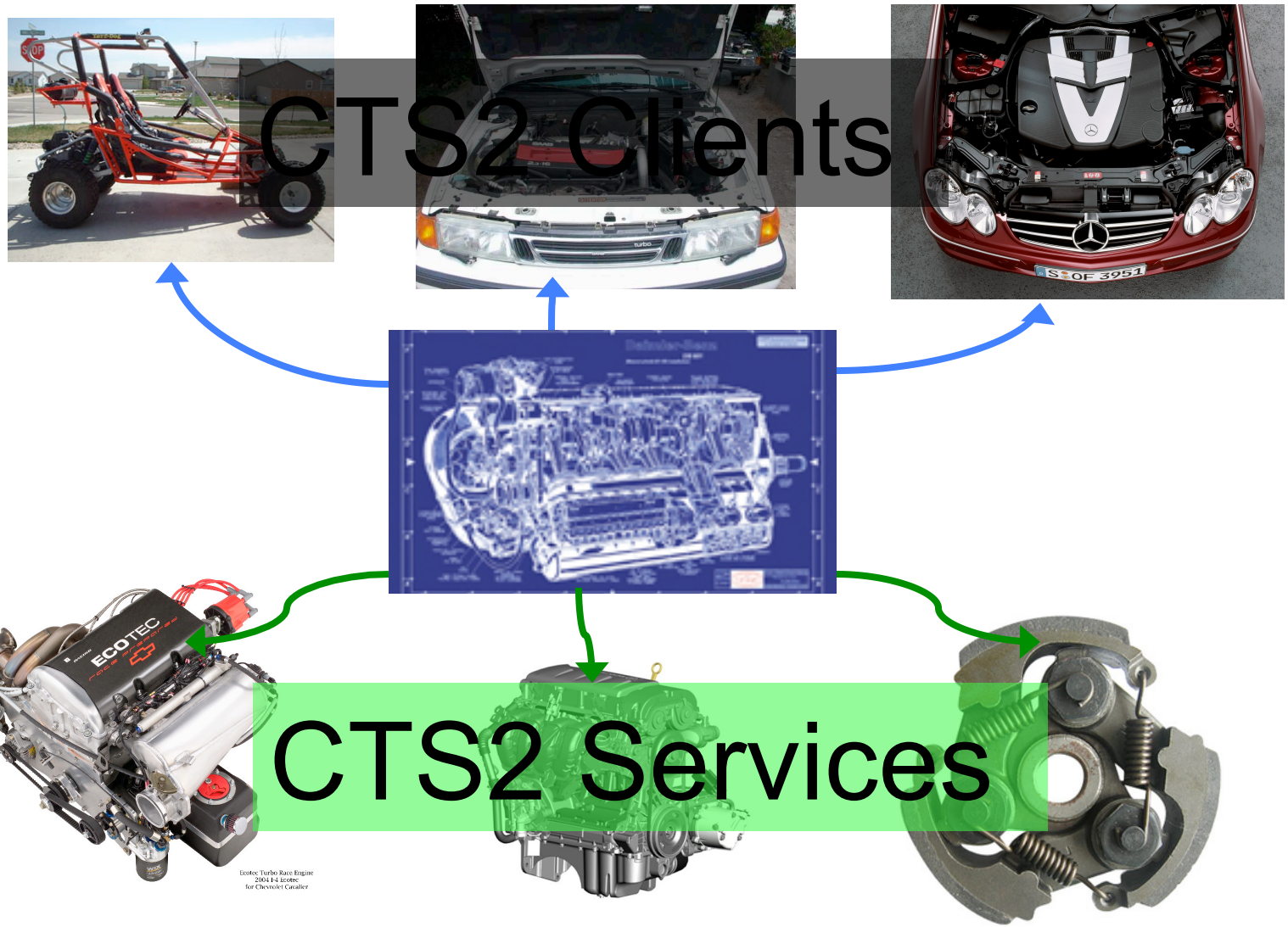
  (Charlie Mead, 2011)
- Proven Technology
  - Architecture of the World Wide Web
  - Can leverage existing WWW tools (HTTP / XML / XSLT / …)
- Enables modular architecture

CTS2 Specification
API

CTS2 is an *Application Programming Interface* (API) specification.

- It defines the semantics, syntax and valid interactions that can occur
- It is a “blueprint” for software
- If everyone follows the blueprint (and the blueprint is sufficiently precise) then CTS2 clients and services can interoperate

CTS2 Standard as a Blueprint



Modular Specification

Implement / use what you need

- Code System Catalog Entry and Code System Version Catalog Entry
- Entity Description and Association
- Value Set Catalog Entry, Value Set Definition and Resolved Value Set
- Map, Map Version
- Concept Domain and Concept Domain Binding
- Statement Model

Modular Functionality

Implement / use what you need

- Read
- Query
- Import
- Export
- Update
- Maintenance
- History
- Temporal

Each Module (or Resource) Has

- Formal Information Model in UML
  - Defines the resource, how it is identified and what information it contains
- Formal Computational Model in UML
  - Defines operations that are available for each functional (read, query, import, …) profile


class CodeSystemCatalogEntry
AbstractResourceDescription
CodeSystemCatalogEntry
codeSystemCategory: CodeSystemCategoryReference [0..1]
codeSystemName: CodeSystemName {readOnly}
currentVersion: CodeSystemVersionReference [0..1] {readOnly}
designedForOntologyTask: OntologyTaskReference [0..*]
hasOntologyLanguage: OntologyLanguageReference [0..1]
includes: CodeSystemReference [0..*]
ontologyDomain: OntologyDomainReference [0..*]
ontologyType: OntologyTypeReference [0..1]
usedOntologyEngineeringTool: OntologyEngineeringToolReference [0..*]
versions: CodeSystemVersionCatalogEntryDirectoryURI [0..1] {readOnly}
::AbstractResourceDescription
releaseDocumentation: OpaqueData [0..1]
releaseFormat: SourceAndNotation [0..*]
::ResourceDescription
about: ExternalURI {readOnly}
additionalDocumentation: PersistentURI [0..*]
alternateID: ExternalURI [0..*]
describedResourceType: CTS2ResourceType {readOnly}
formalName: String [0..1]
keyword: String [0..*]
note: Comment [0..*]
property: Property [0..*]
resourceID: LocalIdentifier {readOnly}
resourceSynopsis: EntryDescription [0..1]
resourceType: URIAndEntityName [1..*]
rights: OpaqueData [0..1]
sourceAndRole: SourceAndRoleReference [0..*]
sourceStatements: StatementDirectoryURI [0..1] {readOnly}
::Changeable
entryID: PersistentURI {readOnly}
entryState: EntryState
status: StatusReference [0..1]
codeSystemName: CodeSystemName {readOnly}
about: ExternalURI {readOnly}
Formal
Information
UML
resourceID: LocalIdentifier {readOnly}
resourceType: URIAndEntityName[1..*]
entryID: PersistenURI {readOnly}
entryState: EntryState

A code system is a resource that is maintained by individuals and/or organizations, typically has a specific goal or purpose and is published and/or updated at periodic intervals. Its purpose is to declare a collection of codes or identifiers that represent classes, categories, or individuals that are used for reporting, organizing and/or reasoning about knowledge in some discipline, specialty or domain. The `CodeSystemCatalogEntry` model carries metadata about the code system itself, while `CodeSystemVersionCatalogEntry` carries information about the *content* of a code system.

Class CodeSystemCatalogEntry

Metadata and access information about a code system.

Superclasses:

- Every instance of `CodeSystemCatalogEntry` is also `AbstractResourceDescription`.

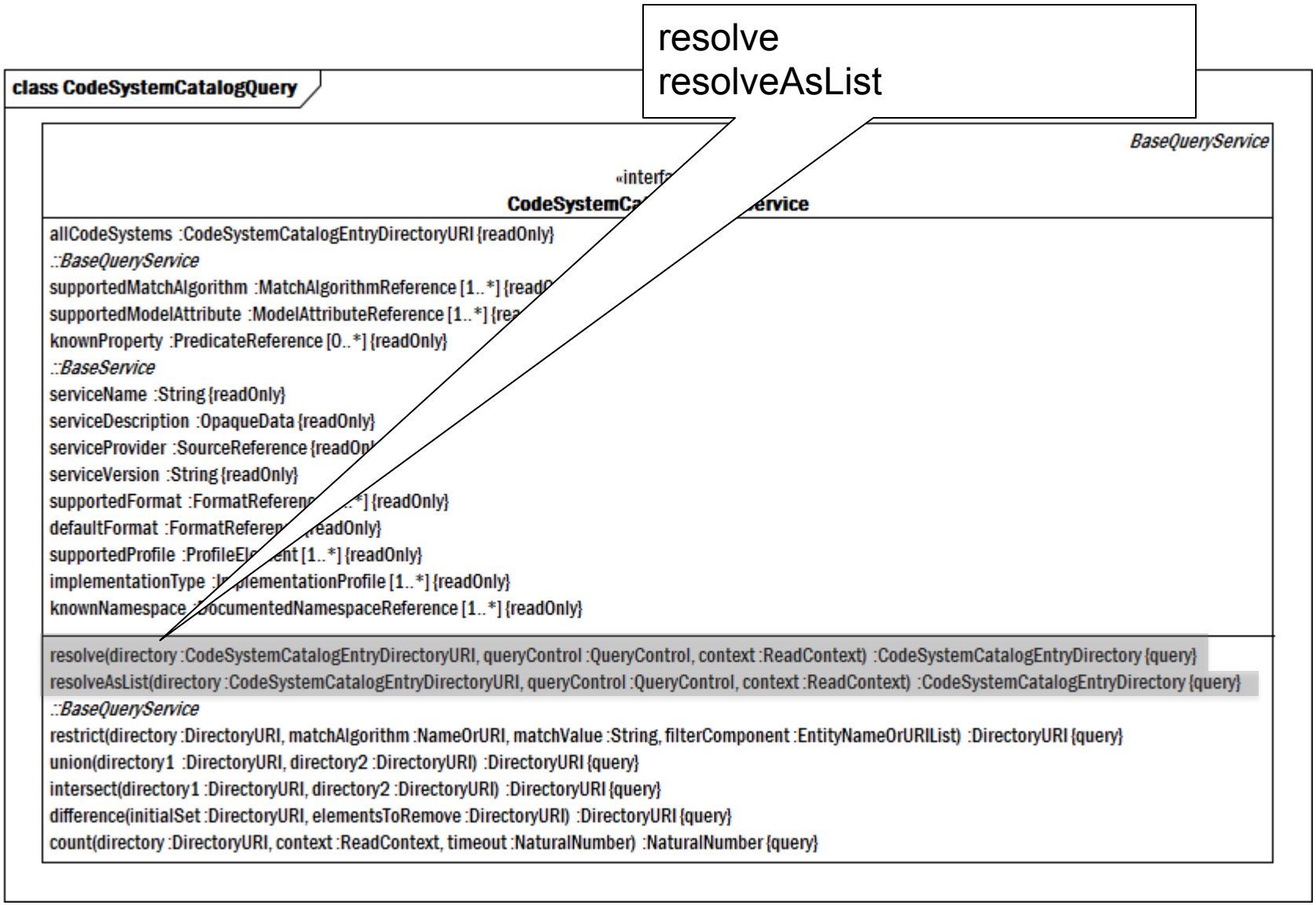
Attributes:

- **codeSystemName** - the local identifier that uniquely identifies the code system within the context of the implementing service. Note that the `about` URI is the globally unique identifier.
- **codeSystemCategory** - the category or type of resource
- **ontologyDomain** - the subject domain of the ontology
- **ontologyType** - the nature of the content of the ontology
- **designedForOntologyTask** - the purpose for which the ontology was originally designed
- **hasOntologyLanguage** - the formal ontology language
- **includes** - a reference to an external code system that is included as a component of the referenced code system. The Wine Ontology, as an example, includes the Food Ontology as one of its components. Similarly, the US Edition of SNOMED-CT includes the international edition.
- **versions** - a `DirectoryURI` that references the known versions of this code system.
- **currentVersion** - a reference to the code system version marked as `CURRENT` in the service instance.
- **usedOntologyEngineeringTool** - information about a tool used to create the ontology

Invariants:

1. The `resourceID` of a `CodeSystem` is the `codeSystemName`.
2. The `describedResourceType` of a `CodeSystem` is `CODE_SYSTEM`.
3. One of the resource types must be owl:Ontology or skos:ConceptScheme.
4. `CodeSystems` cannot have workflow status.

Formal Computational Model



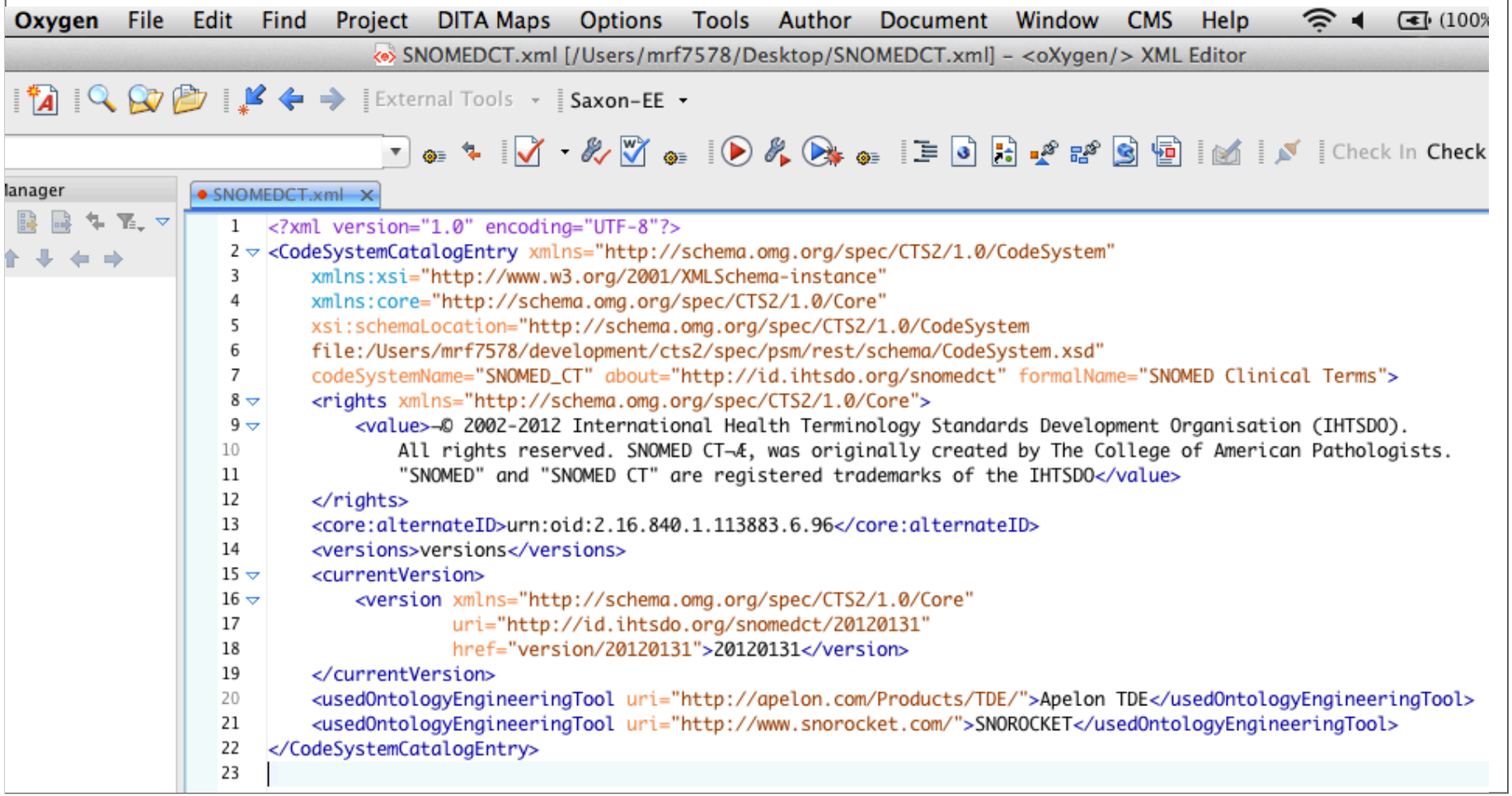
Each Module Also Has

- XML Schema representation of the Information Model
- WADL (Web Application Description Language) a rendering of Computational Profiles
  - Defines the URLs + operations (GET, PUT, POST, DELETE) that implement the computations

XML Schema Sample

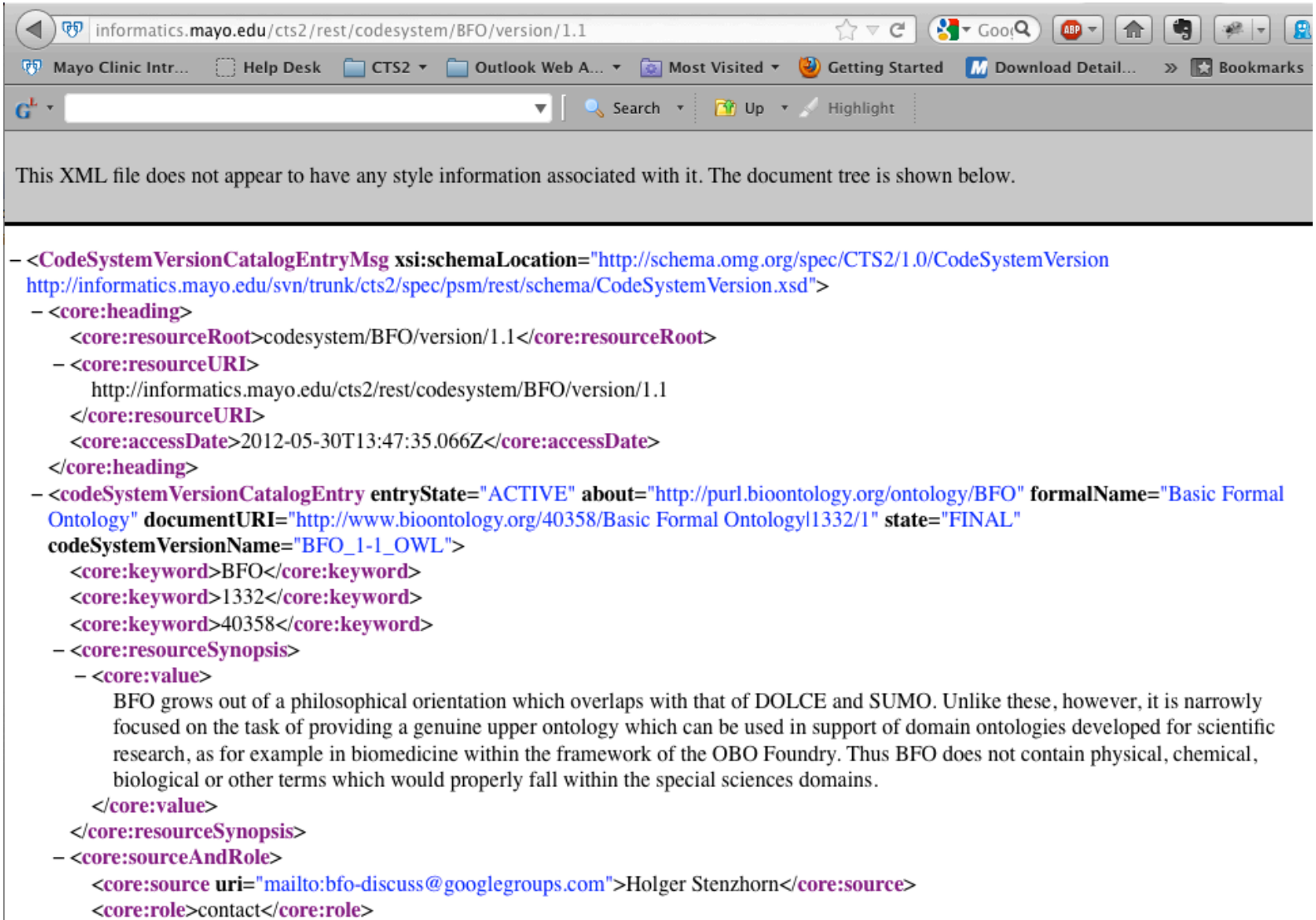
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:core="http://schema.omg.org/spec/CTS2/1.0/Core"
    xmlns="http://schema.omg.org/spec/CTS2/1.0/CodeSystem"
    targetNamespace="http://schema.omg.org/spec/CTS2/1.0/CodeSystem"
    elementFormDefault="qualified">
    <xs:import namespace="http://schema.omg.org/spec/CTS2/1.0/Core" schemaLocation="../core/Core.xsd"/>
    <!-- ========================================================
        CodeSystemCatalogEntry
        ========================================================-->
    <xs:element name="CodeSystemCatalogEntry" type="CodeSystemCatalogEntry">
        <xs:annotation>
            <xs:documentation>Metadata and access information about a code system.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:complexType name="CodeSystemCatalogEntry">
        <xs:annotation>
            <xs:documentation>Metadata and access information about a code system.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="core:AbstractResourceDescription">
                <xs:sequence>
                    <xs:element name="codeSystemCategory" type="core:CodeSystemCategoryReference" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>the category or type of resource that the code system represents.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="ontologyDomain" type="core:OntologyDomainReference" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>the subject domain of the ontology</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="ontologyType" type="core:OntologyTypeReference" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>the nature of the content of the ontology</xs:documentation>
                        </xs:annotation>
                    </xs:element>
```

Content Can be Created with Any XML Compliant tool



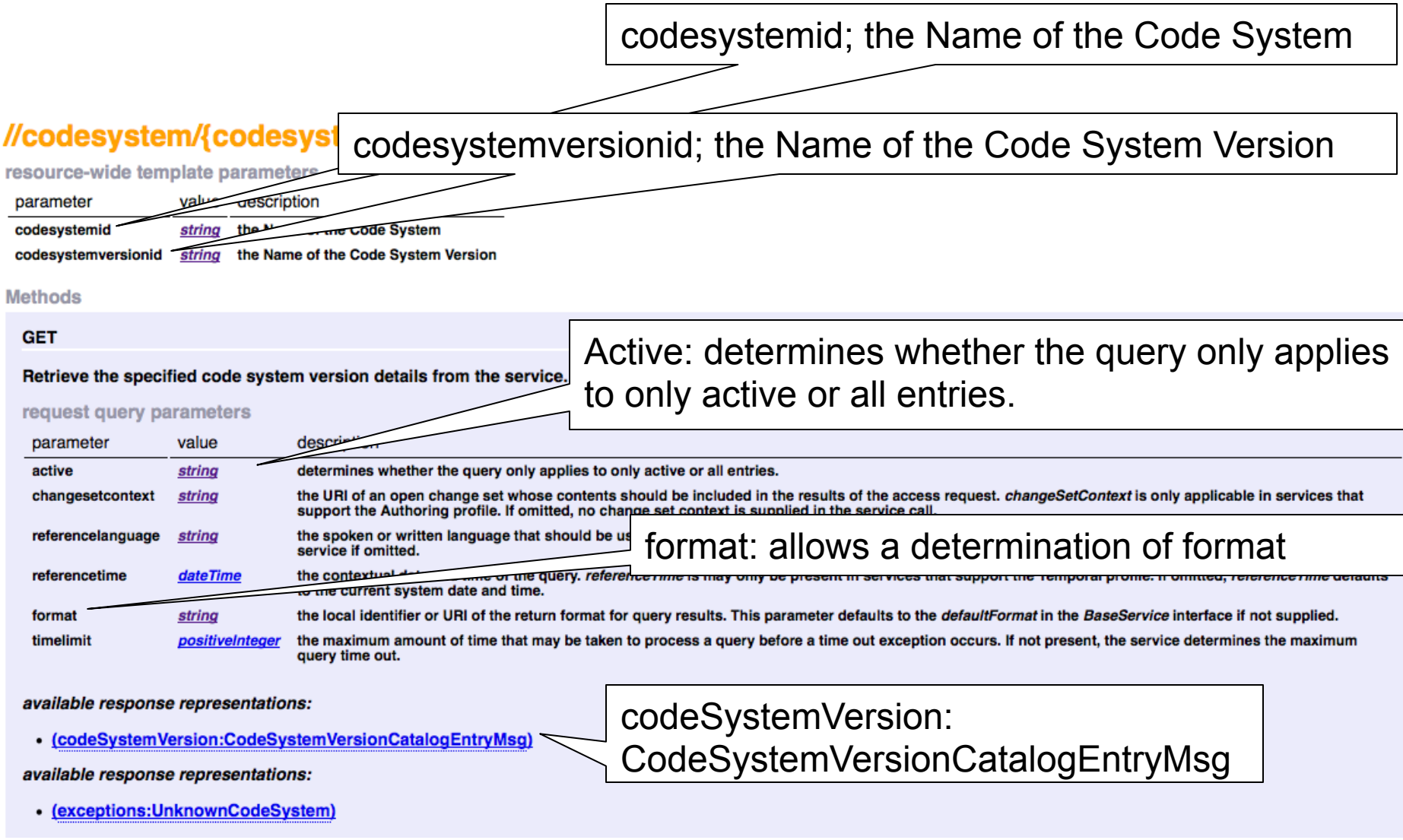
And published as http…

http://informatics.mayo.edu/cts2/rest/codesystem/BFO/version/1.1



WADL Sample

http://informatics.mayo.edu/svn/trunk/cts2/spec/psm/rest/html/cts2.html



Each Module Has

- WSDL Rendering of Computational model
  - Allows SOAP (RPC) access
- JSON rendering of Information Model
  - Not “formal” standard
  - Enables Ajax (browser as a service client)

Combined they provide a set of rules that:

- Describes (some of) what can be said about various aspects of terminologies
- Provides a straight-forward format for structuring the information
- Provides rules for publishing, querying, exchanging and updating information in a distributed, federated environment

… Or …

- Web Services / SOAP Publish Discover…
- JAVA Objects
- JSON and other (standardization still underway)

JSON


```
{
 - codeSystemVersionCatalogEntry: {
      codeSystemVersionName: "BFO_1-1_OWL",
    - versionOf: {
         content: "BFO",
         uri: http://purl.bioontology.org/ontology/BFO,
         href: http://informatics.mayo.edu/cts2/rest/codesystem/BFO
      },
      importsList: [ ],
      supportedLanguageList: [ ],
      entityDescriptions: http://informatics.mayo.edu/cts2/rest/codesystem/BFO/version/1.1/entities,
      associations: http://informatics.mayo.edu/cts2/rest/codesystem/BFO/version/1.1/associations,
      documentURI: "http://www.bioontology.org/40358/Basic Formal Ontology|1332/1",
      state: "FINAL",
    - sourceAndNotation: {
        - sourceDocumentSyntax: {
             content: "OWL"
          }
      },
      officialResourceVersionId: "1.1",
      officialReleaseDate: "Jan 1, 2009 8:00:00 AM",
      about: http://purl.bioontology.org/ontology/BFO,
      formalName: "Basic Formal Ontology",
    - keywordList: [
         "BFO",
         "1332",
         "40358"
      ],
      resourceTypeList: [ ],
    - resourceSynopsis: {
        - value: {
             content: "BFO grows out of a philosophical orientation which overlaps with that of DOLCE and SUMO. Unlike these, however, it is narrowly
             focused on the task of providing a genuine upper ontology which can be used in support of domain ontologies developed for scientific
             research, as for example in biomedicine within the framework of the OBO Foundry. Thus BFO does not contain physical, chemical, biological or
             other terms which would properly fall within the special sciences domains.",
             anyObject: [ ]
          }
      },
```

CTS2

The CTS2 standard is actually a *collection* of relatively small standards…

… and a set of assembly rules.

The goal of the CTS2 standard is *distribution* and *federation:*

- **Distribution** means that different organizations can publish different aspects
- **Federation** means that organizations can share

Key is that implementations can be shared…

… no one has to do it all!

CTS2 Design Philosophy

- Simple things should be simple
- Representational State Transfer architectural style but will support SOAP
- Modular components allow implementation of a subset of a larger specification (Component Resources)

CTS2 Component Resources

- Code System Catalog
- Code System Version Catalog
- Entity Description
- Association
- Map Catalog
- Map Version
- Concept Domain Catalog
- Concept Domain Binding
- Value Set Catalog
- Value Set Definition
- Resolved Value Set

CTS2 Component Resources
Code System Catalog Module

Metadata about code systems (ontologies, code sets, thesauri, classification systems, etc.)

- A service would implement this section to publish information about:
  - What sort of terminological resources are available
  - Who maintains them, what they are intended to be used for
  - How often they are released
  - What copyrights pertain to them, etc.

CTS2 COMPONENT RESOURCES

Code System Version Catalog Module

CTS2 Component Resources
Code System Version Catalog Module

Metadata about specific versions of code systems.

- A service would implement this section to provide information about specific versions of code systems:
  - When they were released
  - What format they are in
  - When they were intended to become active
  - Which version they replace

CTS2 COMPONENT RESOURCES

ENTITY DESCRIPTION MODULE

CTS2 Component Resources
Entity Description Module

A set of entity identifiers known to the service

- These might also be thought of as:
  - Class
  - Category
  - Concept
  - Predicate
  - Property
  - Term
  - Individual

CTS2 Component Resources
Entity Description Module

- Code system version information:
  - Which code system versions make assertions about these identifiers
  - What the codes systems say about them
- Their intended meanings
- What elements in the code system are their children, parents

CTS2 Component Resources
Entity Description Module

The entity description service focuses on the lexical aspect of entity identifiers:

- Providing access to:
  - Designations
  - Definitions
  - Descriptions
  - Examples
  - Usage notes and other artifacts used to represent the meaning of the entity to human consumers

CTS2 Component Resources
Entity Description Module

Why would services implement this module?

- To publish information about the codes described in various versions of code systems

CTS2 COMPONENT RESOURCES

ASSOCIATION MODULE

CTS2 Component Resources
Association Module

- Sets of “semantic” assertions about entity identifiers
- The entity identifier may play the role of
  - Subject
  - Predicate (verb) or
  - Object
  - … in the assertions

CTS2 Component Resources
Association Module

This area represents the formal machine interpretable aspects of code systems

- A service would implement this section to support:
  - Classification
  - Reasoning or other computational logic systems built on terminological content

CTS2 COMPONENT RESOURCES

VALUE SET CATALOG MODULE

CTS2 Component Resources
Value Set Catalog Module

Metadata about sets of entity identifiers (value sets) that have been grouped for some purpose

- A service would implement this section if it wanted to publish information about these collections such as:
  - Who created them
  - Who maintains them
  - What is their purpose
  - What code systems do they depend on
  - How they are updated, etc.

CTS2 COMPONENT RESOURCES

VALUE SET DEFINITION MODULE

CTS2 Component Resources
Value Set Definition Module

Information about how value sets are constructed.

- A value set definition can:
  - Be a simple enumeration of its elements
  - Or can contain instructions about how the elements are assembled from aspects of entity descriptions and associations.

CTS2 Component Resources
Value Set Definition Module

- Value set definitions may be coupled to specific versions of code system,
- May be defined in a way that they could be applied to different versions, potentially with different results.

CTS2 Component Resources
Value Set Definition Module

- Services would implement this section to:
  - Publish the rules on the construction of value sets.
  - The value set definition section includes an optional sub-section (Resolved Value Set) that enables:
    - Publication
    - Loading
    - Use of the results of applying value set definitions

CTS2 Component Resources
Value Set Definition Module

Services would implement the Resolved Value Set section when they needed to consume and use value sets without having access to the underlying code systems.

CTS2 COMPONENT RESOURCES

CONCEPT DOMAIN CATALOG MODULE

CTS2 Component Resources
Concept Domain Catalog Module

- A catalog of abstract “concept domains” that represent a collection of possible meanings
- Concept domains are intended to represent the intended meaning of :
  - A field on a form
  - Column in a database
  - Field in a message, etc.

CTS2 Component Resources
Concept Domain Catalog Module

- The CTS2 specification focuses on enumerated concept domains:
  - Fields that represent discrete collections of “meanings”
  - Each of which is represented by a permissible value

CTS2 Component Resources
Concept Domain Catalog Module

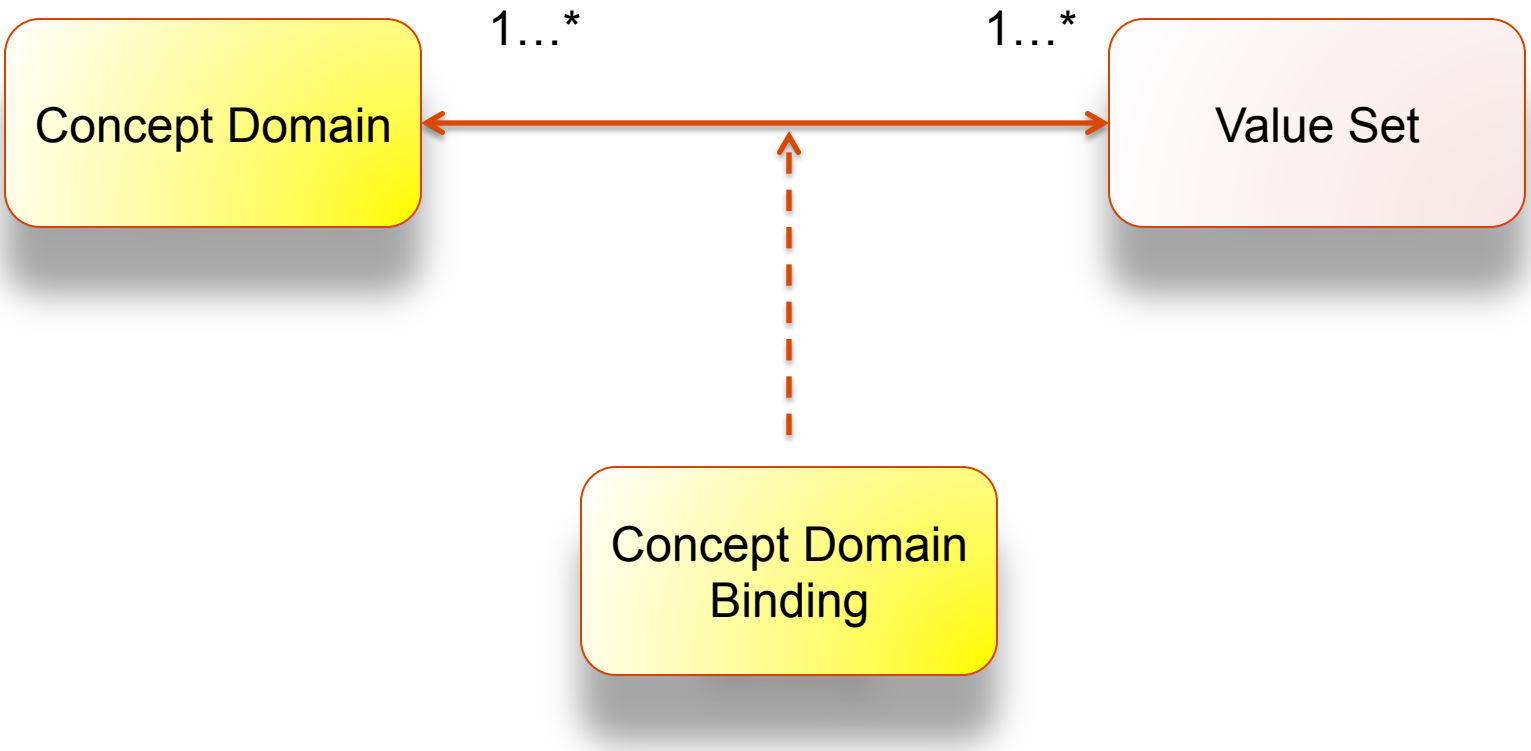
A service would implement this module to provide a list of generic fields that would be used in data interchange.

CTS2 COMPONENT RESOURCES

CONCEPT DOMAIN BINDING MODULE

CTS2 Component Resources
Concept Domain Binding Module

The coupling of a concept domain with a value set, where the value set provides a list of possible meanings that can be used in a concept domain in a particular use case or context.



CTS2 COMPONENT RESOURCES

MAP CATALOG MODULE

CTS2 Component Resources
Map Catalog Module

A catalog of “maps” - collection of rules that allow human or machine assisted transformation between the codes in one value set or code system and those in a second.

CTS2 Component Resources
Map Catalog Module

A service would implement this section to publish information about

- Which rule sets are available
- Which code systems or value sets they map between
- Their intended purpose
- How often they are published
- What formats they are in, etc.

CTS2 COMPONENT RESOURCES

MAP VERSION MODULE

CTS2 Component Resources
Map Version Module

An instance of a map, including the

- Specific value set definitions
- Code systems that they are based on and the actual rules

This module includes an optional sub-section (Map Resolution) that provides access to the machine aided interpretation of map versions – a service that does the actual map transformation

CTS2 Component Resources
Map Version Module

- A service would implement this section if it wished to publish the content of maps.

CTS2 COMPONENT RESOURCES

STATEMENT MODULE

CTS2 Component Resources
Statement Module

- A bridge between the information contained in the various sections described above and the actual assertions made by the information providers
- Statement is also intended to act as a bridge between the structured CTS2 model and simple subject/predicate/target systems such as represented by OWL and RDF
- Contains direct support for bnode and reification

CTS2 Component Resources
Statement Module

- A service would implement this section when it needed to provide additional provenance about assertions made in:
  - Catalogs or resource versions including what was actually said
  - How it mapped to the CTS2 service representation
  - The provenance of the assertion, etc.

CTS2 Functional Profiles

Computational Model for each Module

- Read
- Query
- Import
- Export
- Update
- History
- Maintenance
- Temporal
- (specialized)

CTS2 FUNCTIONAL PROFILES

READ FUNCTION

CTS2 Module Functions
Read Function

Direct access to the contents of a resource via:

- URI

http://informatics.mayo.edu/cts2/services/py4cts2/cts2/entitybyuri?uri=http://snomed.info/id/74400008

- Local identifier

http://informatics.mayo.edu/cts2/rest/codesystem/SNOMEDCT/version/2011_07_31/entity/1000004

CTS2 Module Functions
Read Function

Where applicable, a combination of an abstract resource identifier and version tag (e.g. SNOMED-CT / Current version)

- http://informatics.mayo.edu/cts2/rest/codesystem/EMO/version/1.1

CTS2 FUNCTIONAL PROFILES

QUERY FUNCTION

CTS2 Module Functions
Query Function

- The ability to access, combine and filter lists of resources based on the resource content and user context

http://informatics.mayo.edu/cts2/rest/codesystemversions?matchvalue=anatomy&filtercomponent=resourceSynopsis

CTS2 MODULE FUNCTIONS
IMPORT AND EXPORT

CTS2 Module Functions
Import and Export Functions

The ability to import external content into the service and/or export the contents of the service in different formats.

CTS2 FUNCTIONAL PROFILES

UPDATE FUNCTIONS

CTS2 Module Functions
Update Functions

- The ability to validate load sets of changes into the service that updates its content
- Update is a one step process validated by loading a change log

CTS2 FUNCTIONAL PROFILES

HISTORY FUNCTION

CTS2 Module Functions
History Function

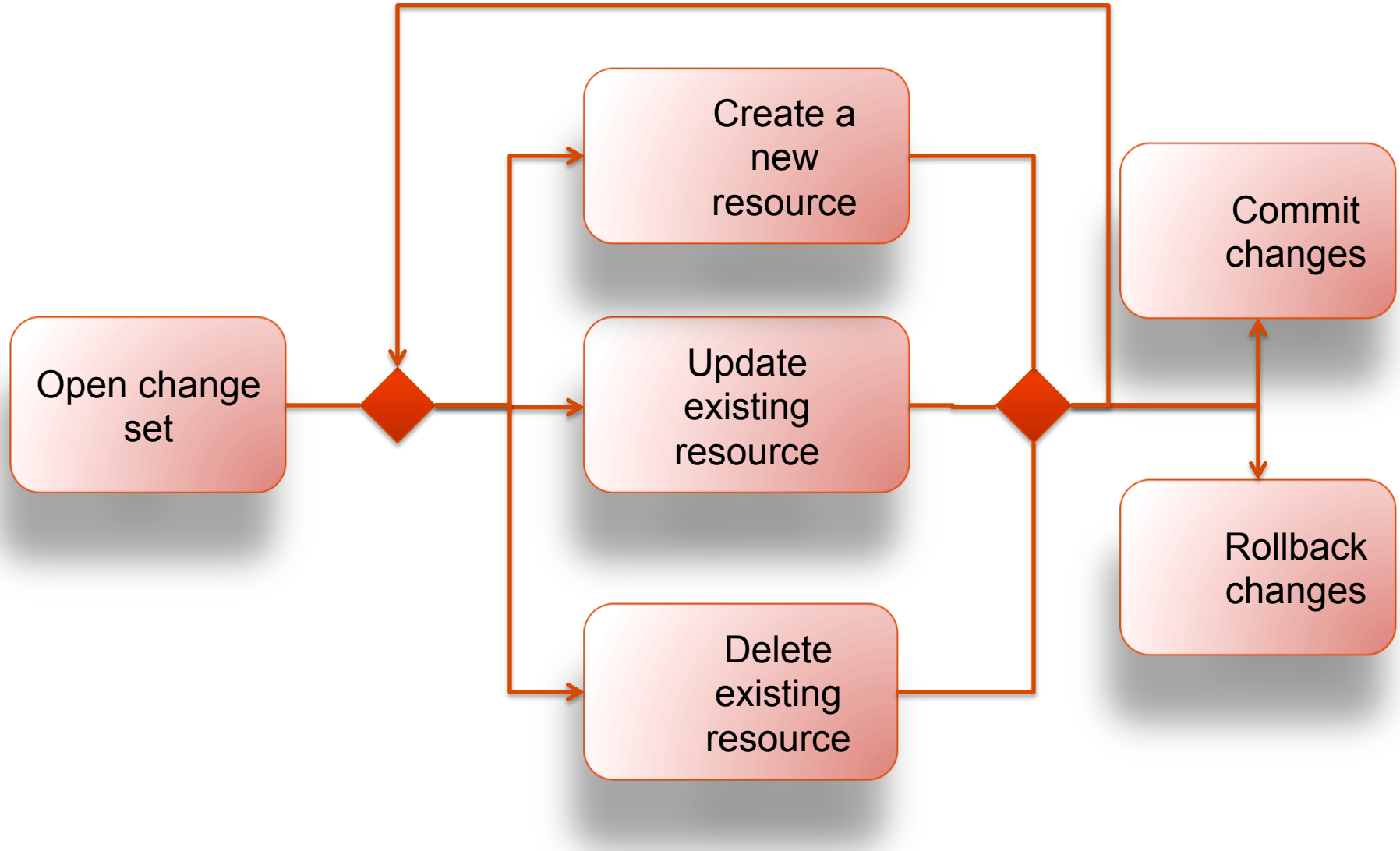
The ability to determine what changes have occurred over stated periods of time.

CTS2 FUNCTIONAL PROFILES

MAINTENANCE FUNCTION

CTS2 Module Functions
Maintenance Function

The ability to create and commit sets of changes


Open change set
Create a new resource
Update existing resource
Delete existing resource
Commit changes
Rollback changes

CTS2 FUNCTIONAL PROFILES

TEMPORAL FUNCTIONS

CTS2 Module Functions
Temporal Function

The ability to ask questions about the state of the service at a given point in the past (or future).

CTS2 FUNCTIONAL PROFILES

SPECIALIZED FUNCTION

CTS2 Module Functions
Specialized Function

- Service specific functions such as:
  - the association reasoning services
  - the map entry services
  - the resolved value set services
- Functions that don't fall into the categories of other functions

CTS2 Module Functions

- The Import, Export and Temporal functions are generic
- There are no resource specific aspects to these services and, as such, they are defined once in the Core Service Elements module
- The remaining components have different signatures depending on the structural domain to which they are applied

CTS2 Module and Functions Summary

At this point you should:

- Have enough information to determine what modules you should might want to implement or consume
- Have some idea what functions you would like to use to consume content

Next let's examine how a project leader might coordinate such an implementation

CTS2 Implementation
Project and Resource Planning

Basic Implementations can be very simple:

Content and Service:

- Content can be created in an XML editor
- Loaded as a page in web server or XML based database and it automatically becomes CTS2 compliant content available to users

CTS2 Implementation
Project and Resource Planning

Content Consumption:

Python

```
conn = Connection("http://informatics.mayo.edu/cts2/rest")
print conn.request_get("/valuesets?matchvalue='Sequence'")['body']
```

Command Line Linux

```
curl http://informatics.mayo.edu/cts2/rest/valuesets > valuesets.xml
```

This can be very easy.

CTS2 Implementation
Project and Resource Planning

High level decisions need to be made about:

- Whether you want to create a service
- Consume services
- Or both

CTS2 Implementation
Project and Resource Planning

High level decisions continued:

- If creating a service:
  - Which modules do you want to implement
  - Which module functions do you want to implement.

CTS2 Implementation
Project and Resource Planning

Service creation requires:

- Ontology and CTS2 expertise to create a map from source to CTS2 model elements
- Development staff with experience in Spring, REST and Java
- Server and system administration resources

CTS2 Implementation
Project and Resource Planning

Service consumption:

- Identify relevant terminology sources
- Establish the range of services implemented at each service
- Satisfy licensing and authentication issues for any restricted sources

CTS2 Implementation
Project and Resource Planning

Service consumption requires:

- Development staff with experience in REST, SOAP (limited), JSON or XML
- Implementation of some kind of http or https connection and parsers for JSON, XML or SOAP messaging

CTS2 Implementation
Project and Resource Planning

Service consumption:

- Development opportunities:
  - Examples available for JavaScript, Java, Scala, and Python
- Platform and implementation strategies:
  - Scope of the implementation
  - Capability of the platform to connect to REST service
  - Consume XML, SOAP, JSON

CTS2 Implementation
Project and Resource Planning

At this point you should be able to:

- Make a decision as to whether you want to provide a service or a client
- Understand what resources are required
- Have a sense of which modules and which module functions you want to implement

CTS2 Summary

- We've had an introduction to the purpose and scope of CTS2
- Described its history and provenance
- Discussed its technical representations as both Platform Independent Model and Platform Specific Model
- Given an overview of all specification modules and their functions
- We'll now proceed to discuss and demonstrate current CTS2 Implementations

CTS2 DEVELOPMENT FRAMEWORK AND IMPLEMENTATIONS

Cory Endle

Outline

- CTS2 Development Framework Overview
- CTS2 Development Framework Implementations
  - eXist DB
  - MU Quality Measure Value Set Service
  - NCBO BioPortal REST
  - NCBO BioPortal RDF

Outline(continued)

- CTS2 Implementation (not created with the Development Framework)
  - SNOMED CT RF2 (py4cts2)
- GUI Implementations
  - CTS2 Value Set Viewer
  - High-Throughput Phenotyping (HTP)
  - CTS2 Maintenance Editor
- UI Widgets

CTS2 DEVELOPMENT FRAMEWORK OVERVIEW

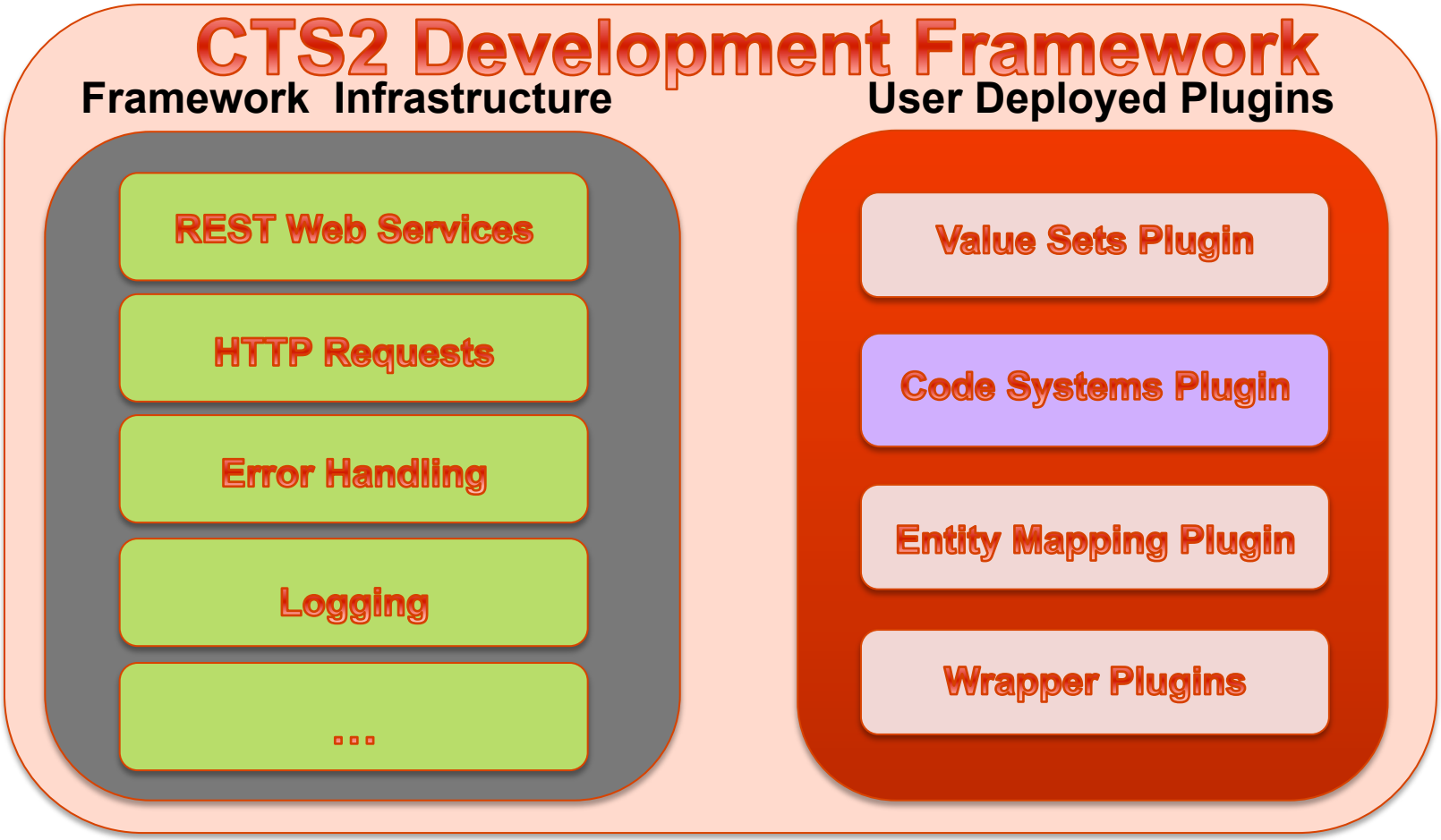
CTS2 Development Framework Overview

- A framework for rapidly creating CTS2 compliant applications
- Allows users to:
  - Create plugins for specific resources (ex. Value Sets)
  - Deploy the plugins into the Development Framework

CTS2 Development Framework Overview

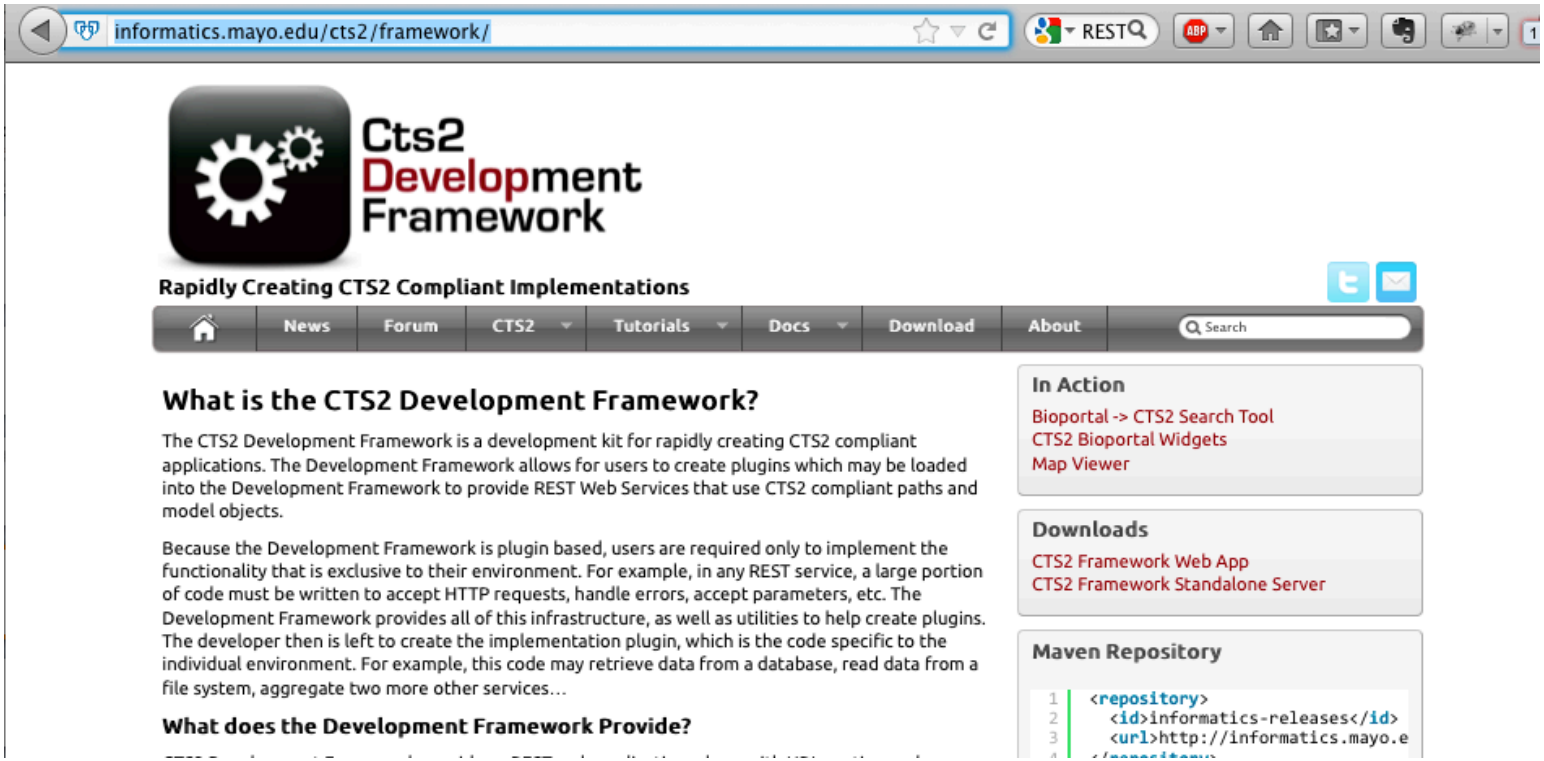
- Developers are required only to implement the functionality that is exclusive to their environment
  - This code may retrieve data from a database, read data from a file system, aggregate two more other services

CTS2 Development Framework Overview



CTS2 Development Framework Overview

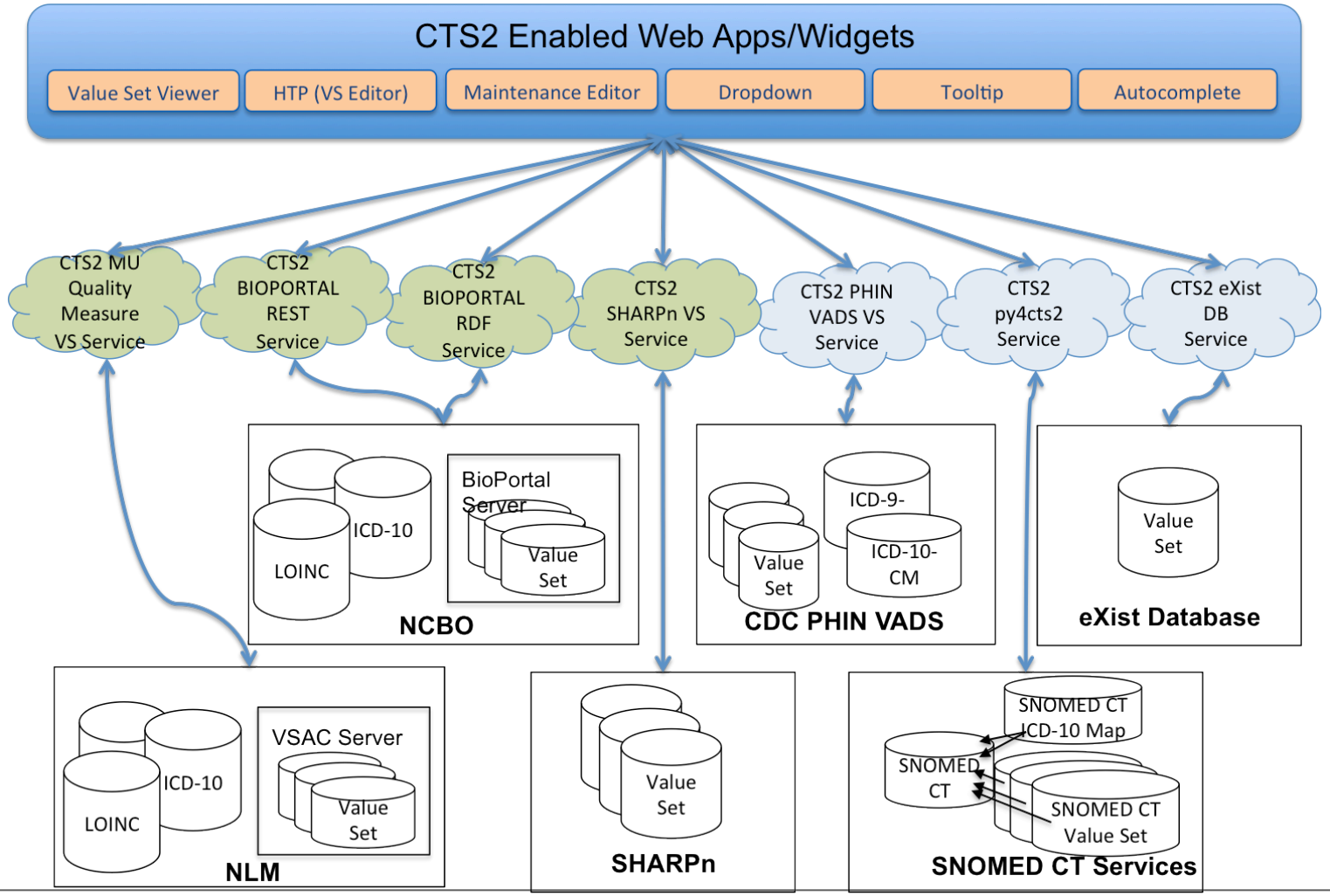
- CTS2 Development Framework
  - http://informatics.mayo.edu/cts2/framework/



CTS2 IMPLEMENTATIONS

CTS2 Implementations

Overview



CTS2 Development Framework Implementations
eXist DB

- eXist XML Database Plugin:
  - A CTS2 implementation based on the eXist Database (Open Source Native XML Database)
- Functions supported:
  - READ
  - QUERY
  - MAINTENANCE

CTS2 Development Framework Implementations
eXist DB

- Supports the most resources of any CTS2 implementation
- Status - Development
- Example of how to implement CTS2 using a NO SQL (XML database) backend

CTS2 Development Framework Implementations
MU Quality Measure Value Set Service

- A CTS2 Query implementation of Value Sets based on Meaningful Use 2 Standards for NQF eMeasures
- Value Sets provided by US National Library of Medicine (NLM)
- This Implementation is a wrapper around NLM's REST interface

CTS2 Development Framework Implementations

MU Quality Measure Value Set Service

- Functions supported:
  - READ
  - QUERY
- Resources supported:
  - Value set
  - Value set definition
  - Resolved value set

CTS2 Development Framework Implementations
MU Quality Measure Value Set Service

- Status – Production
- Important service for users that need Meaningful Use value sets in a CTS2 service

CTS2 Development Framework Implementations
NCBO BioPortal

Component of the National Center for Biomedical Ontologies (NCBO)

Mission:

Create software and support services for the application of principled ontologies in biomedical science and clinical care, ranging from tools for application developers to software for end-users.

CTS2 Development Framework Implementations
NCBO BioPortal

- Web-based application for accessing and sharing biomedical ontologies
  - Search for terms
  - Submit a new ontology to BioPortal library
  - Views on large ontologies
  - Explore mappings between ontologies
- REST based API
- Many useful viewing, browsing and access widgets

CTS2 Development Framework Implementations
NCBO BioPortal REST

- NCBO BioPortal Wrapper
  - A CTS2 implementation based on the NCBO BioPortal REST service
- Functions supported:
  - READ
  - QUERY

CTS2 Development Framework Implementations
NCBO BioPortal REST

- Resources supported:
  - Code System
  - Code System Version
  - Value Set
  - Resolved Value Set
  - Entity
  - Association
- Status - Production

CTS2 Development Framework Implementations
NCBO BioPortal REST

- Example of how CTS2 can be implemented as a wrapper over an existing service
- Augments (doesn't replace) existing solutions
- Enables users full access to all of the NCBO content through the CTS2 service

CTS2 Development Framework Implementations
NCBO BioPortal REST

- http://www.bioontology.org/wiki/index.php/CTS2_BioPortal_wrapper
- http://informatics.mayo.edu/cts2/rest/valuesets

CTS2 Framework Implementations
NCBO BioPortal RDF

- RDF Triplestore Wrapper
  - A CTS2 implementation based on the NCBO BioPortal SPARQL endpoint
- Functions supported:
  - READ
  - QUERY

CTS2 Framework Implementations
NCBO BioPortal RDF

- Resources supported:
  - Code System
  - Code System Version
  - Value Set
  - Resolved Value Set
  - Entity
  - Association
- Status - Production

CTS2 Framework Implementations
NCBO BioPortal RDF

- Example of how to implement CTS2 as a wrapper over an RDF backend using SPARQL

CTS2 Framework Implementations
NCBO BioPortal RDF

- http://www.bioontology.org/wiki/index.php/CTS2_RDF_Plugin
- http://informatics.mayo.edu/cts2/services/bioportal-rdf/valuesets

CTS2 Implementation (using py4cts2)

- SNOMED CT RF2 (py4cts2)
  - A Python implementation of CTS2 to expose the contents of the SNOMED CT RF2 distribution
  - This service provides REST based access to the SNOMED tables and their equivalent CTS2 rendering
  - Functions supported:
    - READ
    - QUERY

CTS2 Implementation (using py4cts2)

- Resources supported:
  - Code System
  - Code System Version
  - Value Set
  - Resolved Value Set
  - Entity
  - Association
  - Advanced Association
- Status - Development

CTS2 Implementation (using py4cts2)

- Most full featured SNOMEDCT implementation
- Has RDF output capability
- Only CTS2 implementation (to date) that supports the Advanced Association (Graph) CTS2 profile
- Implementation of CTS2 that showcases how the CTS2 spec may be implemented in a different language

CTS2 Implementation (using py4cts2)

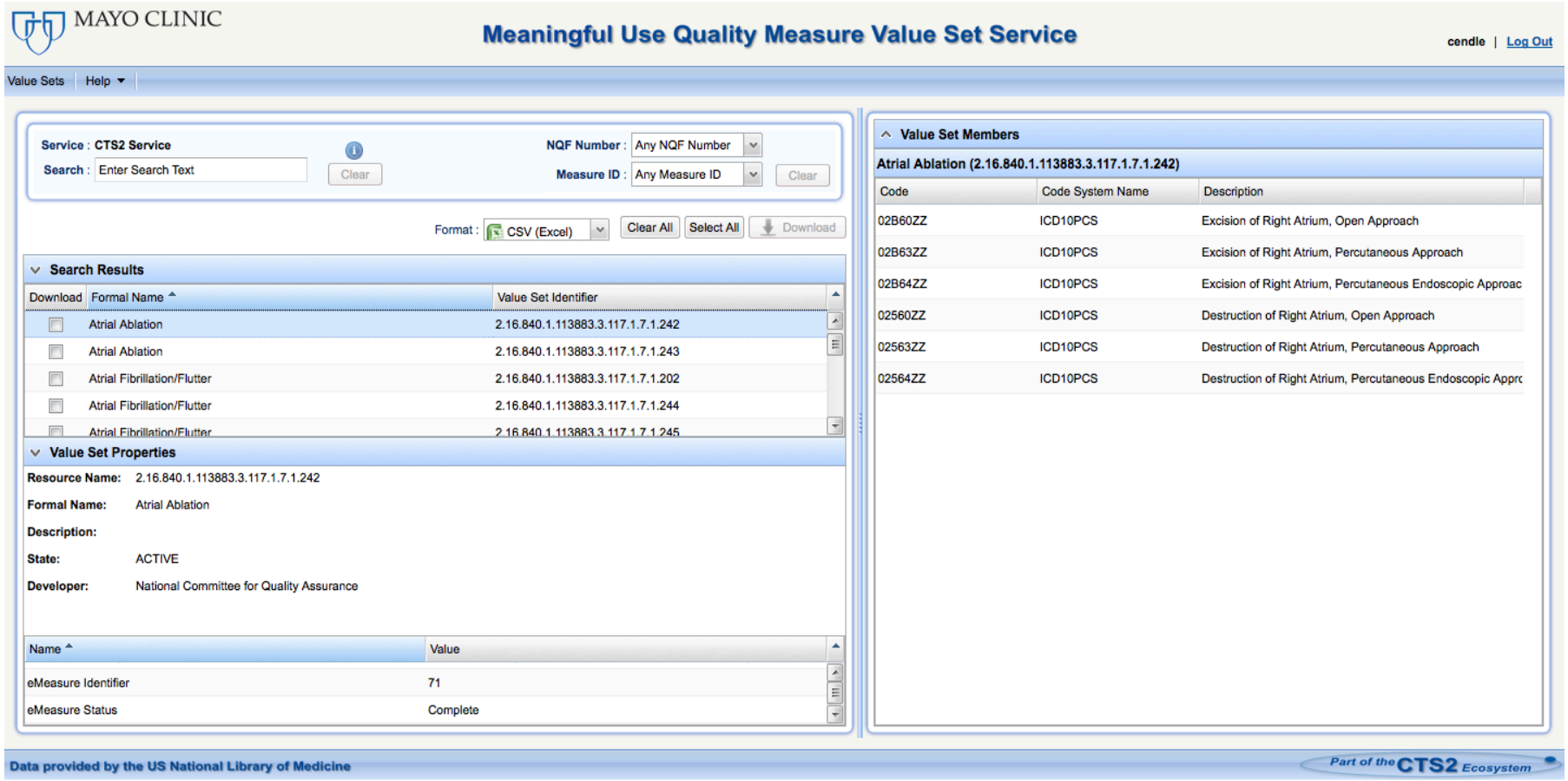
- Examples
  - CTS2 Viewer – Entity Details Window
    - https://informatics.mayo.edu/vsmc/
- http://informatics.mayo.edu/cts2/services/py4cts2

WEB APPLICATIONS USING THE CTS2 SERVICE

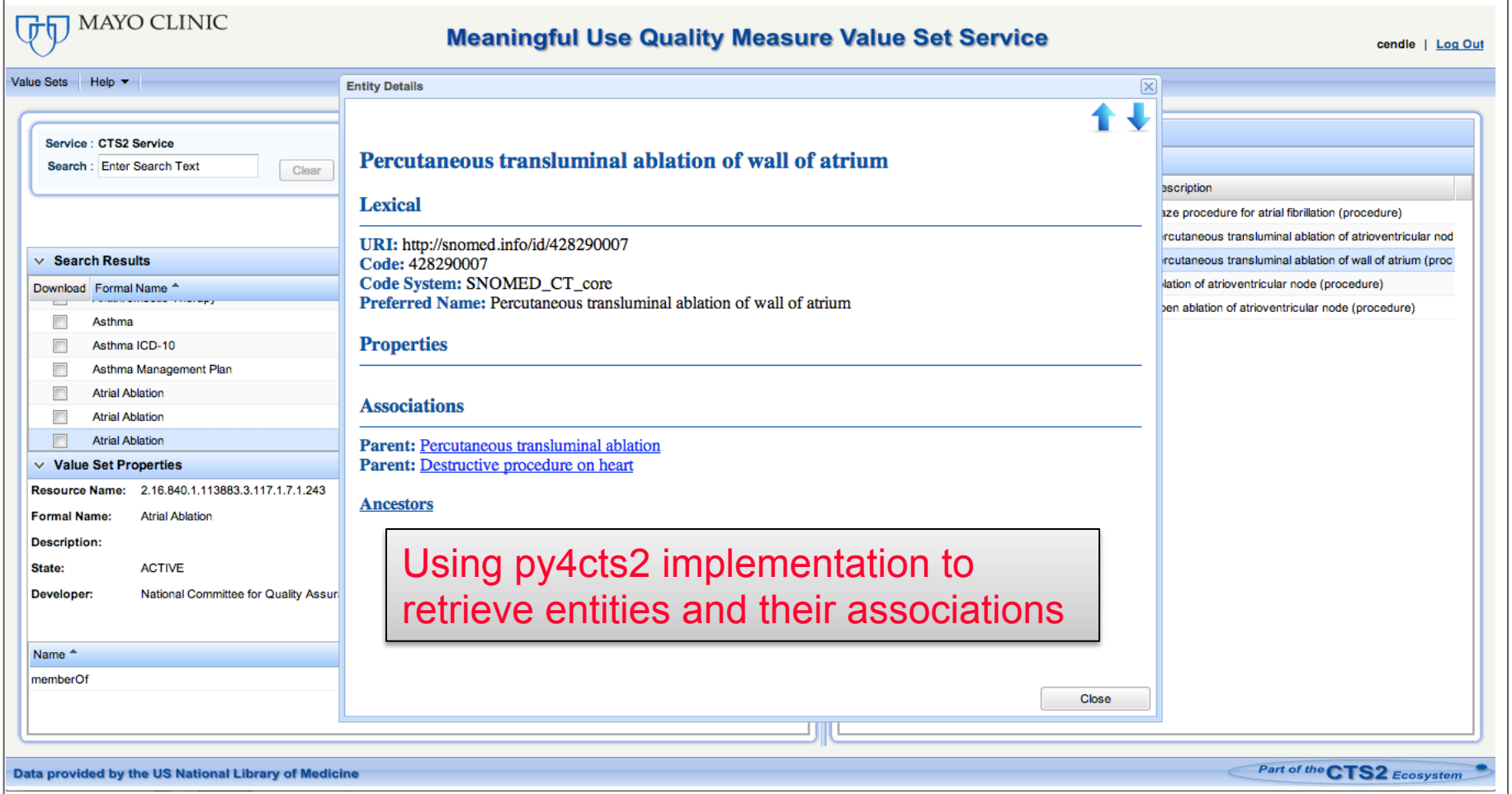
Web Applications
CTS2 Value Set Viewer

- Generic Value Set Viewer to:
  - Browse value sets and its entities
  - Search for value sets
  - Export value sets
- Initially created for MU Quality Measure Value Set Service to serve NLM Value Sets
  - Can view any CTS2 compliant Value Set service (NCBO,PHINVADS CDC,VSAC)
- https://informatics.mayo.edu/vsmc/?showAll=true

Web Applications
CTS2 Value Set Viewer



Web Applications
CTS2 Value Set Viewer



Web Applications
High-Throughput Phenotyping (HTP)

- A tool to help identify patient cohorts using electronic health record (EHR) data by leveraging informatics-based phenotyping processes
- Rules from the National Quality Forum (NQF) Quality Data Model (QDM) files can be uploaded to the Phenotype Portal

Web Applications
High-Throughput Phenotyping (HTP)

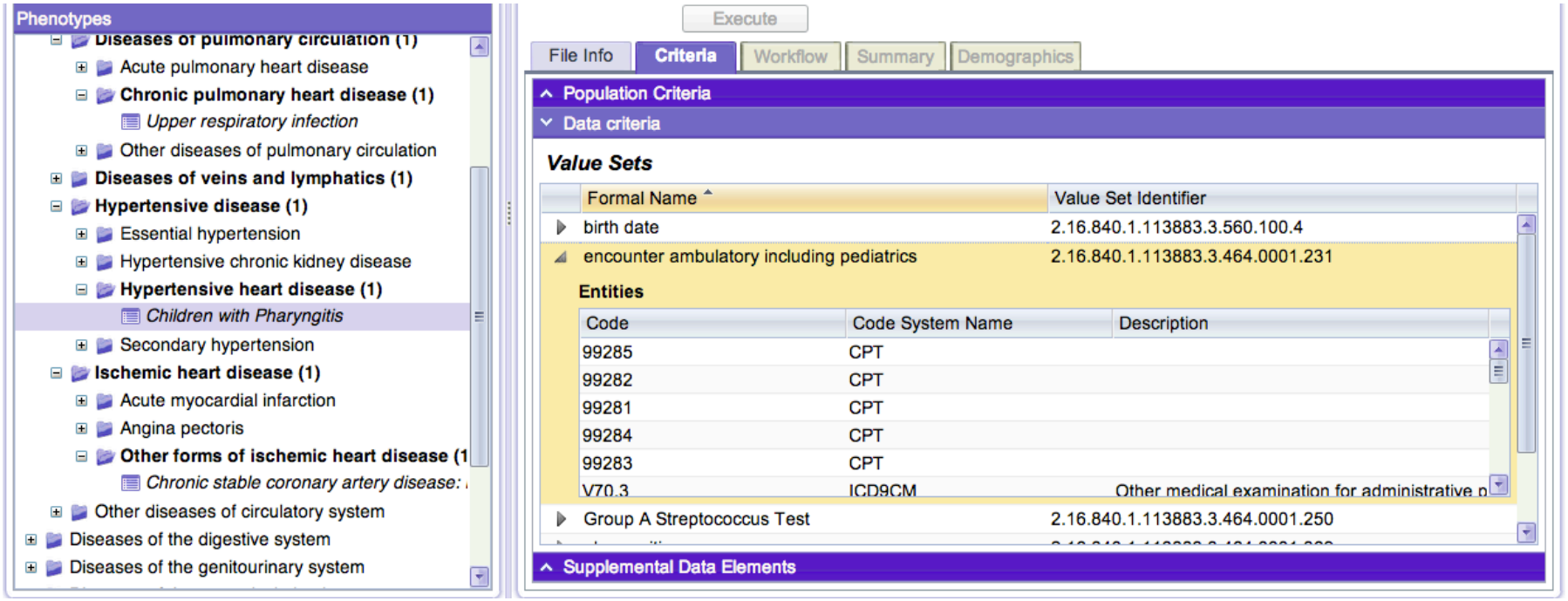
- Convert structured phenotype criteria into executable queries using JBoss Drools Rules
- Runs against Clinical Element Model (CEM) DB and EHRs to find cohorts that match the given criteria

Web Applications
High-Throughput Phenotyping (HTP)

- Users can currently view Value Sets and their entities via the CTS2 service
- Planned availability to add/remove entries from Value Sets and save them. This will allow the user to execute the phenotypes with modified criteria to see how it affects the results

Web Applications
High-Throughput Phenotyping (HTP)

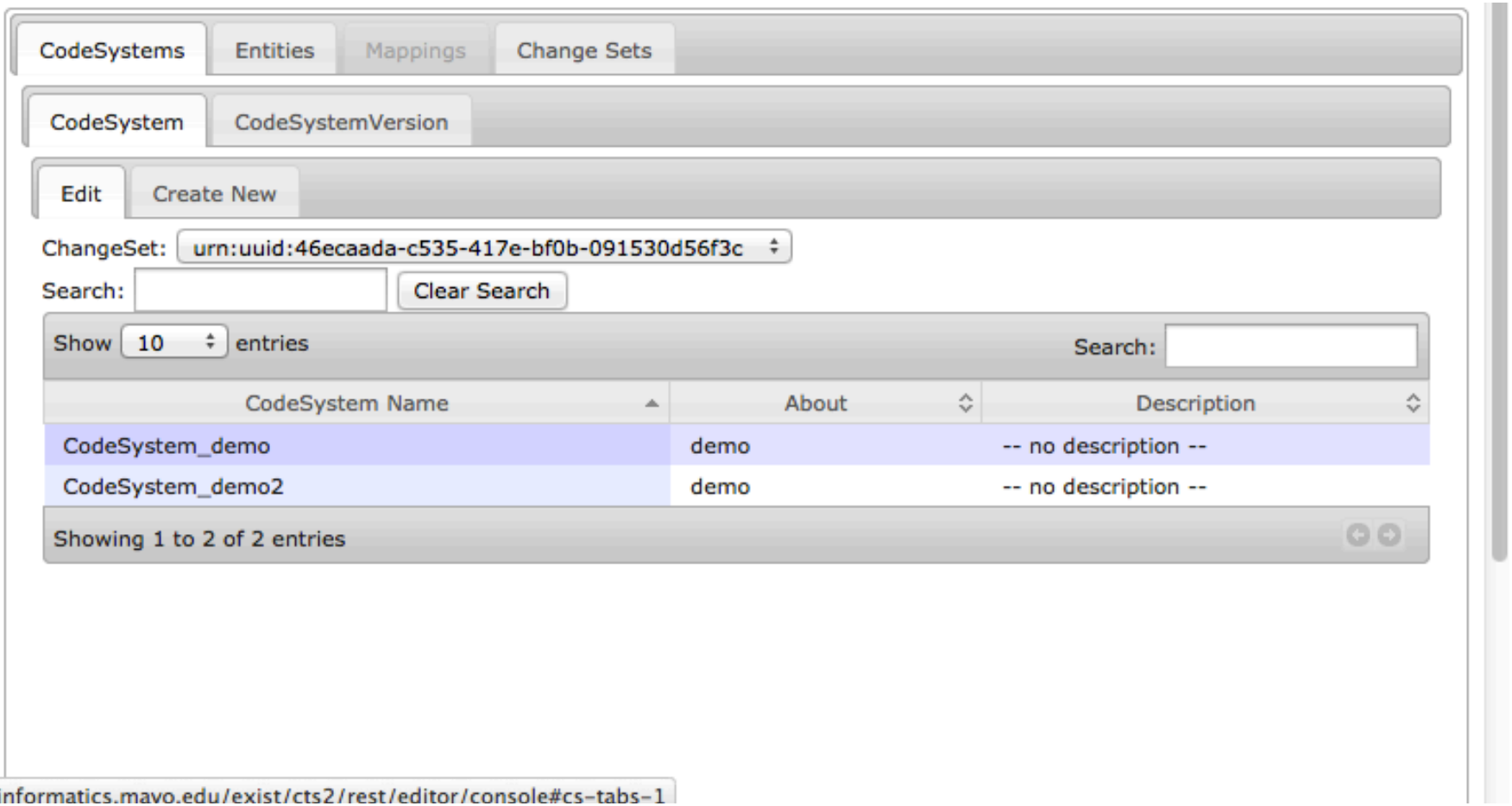
- http://phenotypeportal.org/



Web Applications
CTS2 Maintenance Editor

- Application for managing CodeSystems, Entities, and Change Sets
- Demonstration of CTS2 maintenance functionality
- Currently in development
- http://informatics.mayo.edu/exist/cts2/rest/editor/console

Web Applications
CTS2 Maintenance Editor



CTS2 ENABLED UI WIDGETS

CTS2 Enabled UI Widgets
Overview

- Tooltips
  - jQuery tooltip widget used to display Value Sets and Code Systems when you mouse over text
- Autocomplete
  - jQuery autocomplete widget used to suggest Value Sets as you type

UI Widgets
Overview

- Dropdown
  - jQuery example to fill a select (dropdown) with specific resolved value sets

CTS2 Enabled UI Widgets
Demonstration

- http://informatics.mayo.edu/cts2widgets/widgets.html
- Features and customization points of each widget:
  - Ability to set the Value Set ID/Code System ID to retrieve different values
  - Cascading Style Sheets (CSS) to change the look and feel
  - CTS2 parameterized values to customize the query

CTS2 Enabled UI Widgets
Future

- Started out with some simple UI widgets from Mayo
- Envision a whole library of CTS2 enabled widgets (including more sophisticated widgets) will be developed over time from:
  - Mayo
  - CTS2 community

CTS2 WEB APPLICATIONS/UI WIDGET REVIEW

CTS2 Web Applications/UI Widget Review

- Several examples of web applications and widget to show the power of the CTS2 compliant services
- CTS2 compliant web apps and UI widgets can be used to interact with any CTS2 compliant service

QUESTIONS

References

- CTS
  - http://informatics.mayo.edu/LexGrid/downloads/CTS/specification/ctsspec/cts.htm
- LexEVS
  - https://cabig-kc.nci.nih.gov/Vocab/KC/index.php/LexEVS
- BioPortal
  - http://bioportal.bioontology.org/
- BioPortal REST Services
  - http://www.bioontology.org/wiki/index.php/BioPortal_REST_services
- CTS2
  - http://informatics.mayo.edu/cts2

References

- CTS2 Development Framework
  - http://informatics.mayo.edu/cts2/framework/
- Healthcare Services Specification Program
  - http://hssp.wikispaces.com/
- BioPortal REST Wrapper
  - http://informatics.mayo.edu/cts2/rest/valuesets
- BioPortal RDF Wrapper
  - http://informatics.mayo.edu/cts2/services/bioportal-rdf/valuesets
- BioPortal SPARQL endpoint:
  - http://sparql.bioontology.org/

References

- MU Quality Measure Value Set Service
  - https://informatics.mayo.edu/cts2/services/mat/valuesets
- GUIs that display CTS2 Content:
  - https://informatics.mayo.edu/vsmc/?showAll=true
  - http://www.phenotypeportal.org/
  - http://informatics.mayo.edu/exist/cts2/rest/editor/console
- UI Widgets Example:
  - http://informatics.mayo.edu/cts2widgets/widgets.html
- UI Widgets in GitHub:
  - https://github.com/cts2/
  - https://github.com/cts2/cts2autocomplete
  - https://github.com/cts2/cts2tooltip
  - https://github.com/cts2/cts2dropdown

Questions & Discussion