I. Using OpenCDS Concepts

Contents

Change Log

Date	Author	Notes
12-17-2011	David Shields	Initial document
12-20-2011	Ken Kawamoto	Minor edits; added more details on use of Apelon.
12-20-2011	David Shields	Add supporting graphics and examples, separate "OpenCDS Concepts" from "concept instances."
12-21-2011	David Shields	More examples, improved graphics, explain both OpenCDS and user maintained concepts better.
4-6-2012	David Shields	Reworded definitions, improved graphics, worked on better linkage from ideas about OpenCDS Concepts to the actual setup files used for OpenCDS.

Introduction

OpenCDS is a Clinical Decision Support system that is built around the notion of "clinical concepts." There are many medical terminology systems and medical information exchange systems that refer to "concepts," "concept descriptors (CDs)," or "concept unique identifiers (CUIs)."

When we refer to an OpenCDS Concept we are referring to specific implementation techniques within OpenCDS which have a strong dependence on most of these ideas, but in a concrete implementation.

The purpose of this document is to explain how we use all of these terms in OpenCDS, and how they relate to the general clinical understanding of "concepts."

A. What is an "OpenCDS Concept?"

An OpenCDS Concept is an implementation technique in OpenCDS. As a big picture idea, OpenCDS has a structure and methods that are designed to allow the clinical user to develop decision support rules that are written using clinical concepts. We call those particular concepts "OpenCDS Concepts", and they provide an interface to the detailed data that represents an instance of the clinical concept.

This means that the clinical rule writer can work using clinical terminology that clinicians understand. OpenCDS is designed to support rules written for the open-source Drools inferencing engine, using a domain specific language (DSL). The DSL makes it possible to write the rules so that they read just like the way a clinician would describe them, using terms (OpenCDS Concepts) that the clinician uses every day.

A medical informaticist, terminologist, or vocabulary expert then produces mappings of those concepts to values in a code system which is used in the actual clinical data. In many cases this will involve the use of terminology management systems with large databases, such as Apelon, First DataBank, and UMLS which support potentially large and internationally supported terminologies. However, it is possible to also create simple XML files that map proprietary or special-case codes to OpenCDS Concepts.

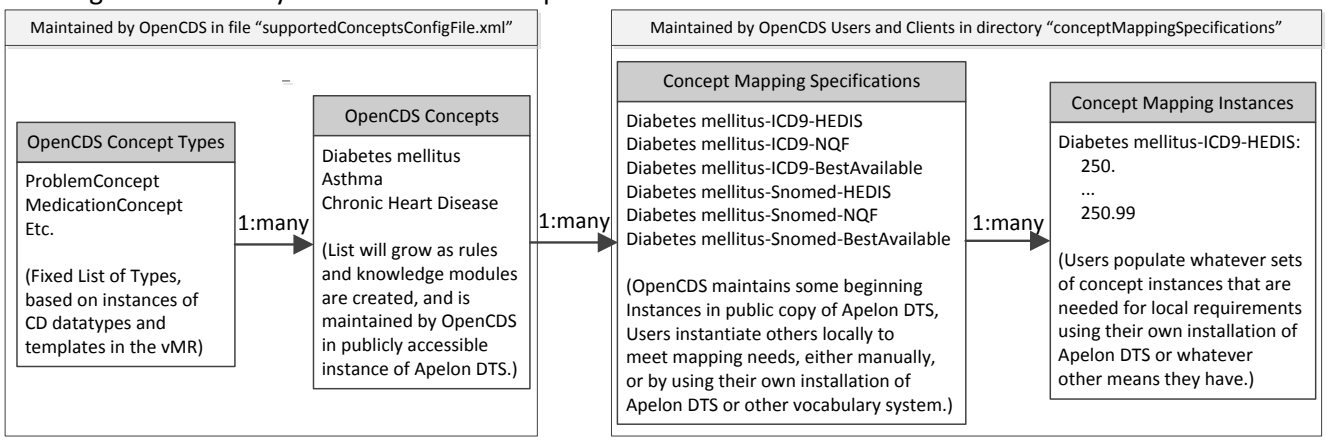
The clinical rules use OpenCDS Concepts in preference to references to the raw data, and the terminology mappings provide implementations of those concepts as lists of codes from one or more code systems. This separates the logic of the rules from the details of the data which the rules work on.

Therefore, **an OpenCDS Concept is the interface between the clinical ideas and the data details** that represent instantiations of the clinical concepts.

This section of this document will explain the following items, and discuss how they are used in OpenCDS:

- Concept Types
- OpenCDS Concepts
- Concept Instances
- Concept Mappings and Enumerations

The diagram below may make this relationship easier to visualize:



Following sections will discuss the techniques used to relate the patient data to the clinical concepts or ideas in the following software:

- JBoss Drools Guvnor (our supported authoring environment for KnowledgeModules, aka "rules" or generically as "knowledge")
- OpenCDS Decision Support Service (our software service to do clinical decision support)

1. Concept Types

a) Definition:

Concept Types in OpenCDS is the term we use to refer to Java classes in OpenCDS that have been created for every "concept descriptor" (aka CD) and "template" found in the input data structured as a virtual medical record (vMR).

A Concept Type represents an entire category of information that can be found at several places in a clinical statement. Clinical statements are the basic building blocks of the vMR. The vMR is more fully discussed in a separate document named "**Notes on OpenCDS internal Data Structure**." In addition, there are also a few additional Concept Types that deal with "templates" and some structural elements of the vMR.

In this document, I will always capitalize Concept Type when I am referring to those Java classes, and use lower-case when I am referring to the more common or generic notions of "concept."

In some cases, we have also post-coordinated more than one CD in the input data (such as "body site" and "laterality"), because there may be a need to make decisions based on the combination of the two, or common coding systems for a procedure or problem may not provide adequate specificity without being associated with both values as a set. For example, we need to be able to say that a rash was present both on the right arm and the left leg.

Examples of Concept Types are a "ProcedureConcept," an "AdverseEventAgentConcept," a "ProblemConcept", or a "MedicationClassConcept."

Because they are based on specific data elements in the vMR, OpenCDS supports a fixed list of these Concept Types, including a few that are broad enough to fill most needs for an "other" category. While we can add more Concept Types if the need arises, we feel that the current list is useable. Supporting additional Concept Types is non-trivial, so it will not be done casually.

The authoritative list of supported Concept Types can always be found in the source code as separate files in the **module opencds-vmr-1_0-internal** as separate Java classes in the package **org.opencds.vmr.v1_0.internal.concepts**. These files have the name of the Concept Types as you would see them in a Drools Rule. They are also listed for a different purpose (building enumerations) in a Java class named "**org.opencds.common.terminology.OpenCDSConceptTypes**.java".

The current list of Concept Types as of the time this document was written is shown on the next page:

AdverseEventAffectedBodySiteConcept
AdverseEventAffectedBodySiteLateralityConcept
AdverseEventAgentConcept
AdverseEventConcept
AdverseEventCriticalityConcept
AdverseEventSeverityConcept
AdverseEventStatusConcept
BrandedMedicationConcept
CDSInputTemplateConcept
CDSOutputTemplateConcept
ClinicalStatementEntityInRoleRelationshipConcept
ClinicalStatementRelationshipConcept
ClinicalStatementTemplateConcept
DataSourceTypeConcept
DoseTypeConcept
DosingSigConcept
EncounterCriticalityConcept
EncounterTypeConcept
EntityRelationshipConcept
EntityTemplateConcept
EntityTypeConcept
EthnicityConcept
EvaluatedPersonRelationshipConcept
GenderConcept
GenericMediccationConcept
GoalCodedValueConcept
GoalCriticalityConcept
GoalFocusConcept
GoalStatusConcept
GoalTargetBodySiteConcept
GoalTargetBodySiteLateralityConcept
ImmunizationConcept
InformationAttestationTypeConcept
InformationRecipientPreferredLanguageConcept
InformationRecipientTypeConcept
ManufacturerConcept
MedicationClassConcept
MedicationConcept
ObservationCodedValueConcept
ObservationCriticalityConcept
ObservationFocusConcept
ObservationInterpretationConcept
ObservationMethodConcept
ObservationTargetBodySiteConcept
ObservationTargetBodySiteLateralityConcept
ObservationUnconductedReasonConcept
PreferredLangugageConcept
ProblemAffectedBodySiteConcept
ProblemAffectedBodySiteLatgeralityConcept
ProblemConcept
ProblemImportanceConcept
ProblemSeverityConcept
ProblemStatusConcept
ProcedureApproachBodySiteConcept
ProcedureApproachBodySiteLateralityConcept
ProcedureConcept
ProcedureCriticalityConcept
ProcedureMethodConcept
ProcedureTargetBodySiteConcept
ProcedureTargetBodySiteLateralityConcept
RaceConcept
ResourceTypeConcept
RoleConcept
SubstanceAdministrationApproachBodySiteConcept
SubstanceAdministrationApproachBodySiteLateralityConcept
SubstanceAdministrationCriticalityConcept
SubstanceAdministrationGeneralPurposeConcept
SubstanceAdministrationTargetBodySiteConcept
SubstanceAdministrationTargetBodySiteLateralityConcept
SubstanceDeliveryMethodConcept
SubstanceDeliveryRouteConcept
SubstanceFormConcept
SubstanceManufacturerConcept
SupplyConcept
SupplyCriticalityConcept
SupplyTargetBodySiteConcept
SupplyTargetBodySiteLateralityConcept
SystemUserPreferredLanguageConcept
SystemUserTaskConctextConcept
SystemUserTypeConcept
UndeliveredProcedureReasonConcept
UndeliveredSubstanceAdministrationReasonConcept
VmrOpenCdsConcept
VMRTemplateConcept

2. OpenCDS Concepts

a) Definition

An OpenCDS Concept is a specific instance of an OpenCDS Concept Type that identifies a clinical concept familiar or useful to clinicians in the context of decision support, and has been identified and assigned an ID in OpenCDS using Apelon DTS.

Each OpenCDS Concept may be one of zero to many instances of a Concept Type for a particular OpenCDS KnowledgeModule. For example, a ProblemConcept may have instances for diabetes mellitus, asthma, or chronic heart disease, and possibly many others. A MedicationClassConcept might have instances for bronchodilator, or ACE inhibitor, and so on...

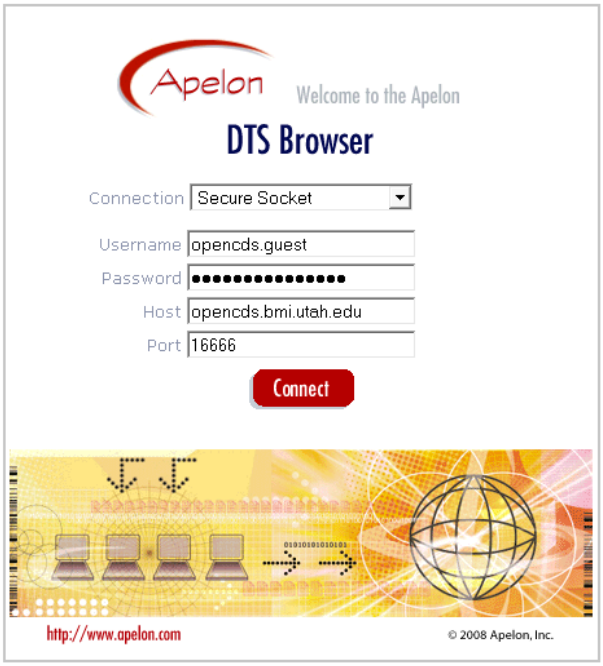
OpenCDS Concepts are created by users of OpenCDS as they are needed for particular knowledge modules. OpenCDS ships with an introductory set, and this will be continually expanded as needed.

It is our intention to maintain OpenCDS Concept codes centrally in order to ensure interoperability among knowledge modules from all OpenCDS users, and allow for sharing knowledge modules between different organizations.

OpenCDS Concepts can be temporarily created through the OpenCDS configuration files during software and knowledge module development, but any such *temporary* concepts should be added to the central OpenCDS Concept list, which is maintained by the University of Utah group on an Apelon DTS terminology server. Adding the Concept to the OpenCDS Apelon instance will assign it a unique ID that can then be used by all implementers of OpenCDS.

The contents of this server can be viewed as follows:

- Go to **Apelon.opencds.org**
- Login with the following parameters (password is “welcome2opencds”).



The master set of concepts can be generated for local use within the OpenCDS service from this central Apelon repository using a utility in the **opencds-decision-support-service module** named **org.opencds.terminology.OpenCDSConceptsFileCreator**.

3. Concept Mapping Specifications

a) Definition:

A concept mapping specification in OpenCDS is an OpenCDS Concept that has an associated Determination Method and may or may not specify an associated Code System.

It is normal usage to create one or more concept mapping specifications for every OpenCDS Concept that is used. Concept mapping specifications will normally be populated with lists of codes from the associated code system(s) that meet the requirements of the associated determination method, and when this happens the concept instance is more commonly called simply a concept mapping. However, it is possible to define a concept mapping specification without mapping instances.

We frequently don't accurately distinguish between concept instances without mappings and concepts with mappings, because defining the mapping specification is normally the first step to creating the mappings. However, it is useful to separate the notion of a concept mapping specification from the mappings, because it is

a two-layer process: There may be multiple sets of concept mapping specifications for each OpenCDS Concept, and there may be multiple mapping instances to specific codes for each concept mapping specification.

(1) Associated Determination Method

One important attribute of a concept mapping specification in OpenCDS is that it is always associated with a "determination method." This allows for the creation of similar concept mappings that are based on different requirements. This is particularly useful when you want to work with a concept as defined by some regulatory body, such as HEDIS or NQF. These organizations typically have very specific lists of codes and code systems that describe particular concepts they are interested in.

It is also possible to define your own "determination method" for specific concept instances. Some possibilities might include "best available," "in approved formulary," "board approved," "expensive," or "cost effective," and so on. The idea is to label the criteria used to map particular instances of codes and code systems to the generic concept.

It is entirely acceptable to have more than one determination method for a single OpenCDS Concept, and it is possible to use more than one in the same set of rules. For example, you might want to be able to distinguish between the NQF determination method, the HEDIS determination method, and the best available determination method for your particular clinical use case in talking about diabetes mellitus.

Typically, however, you may want to work with a single determination method in any one particular KnowledgeModule. We include sample DSL to select a single determination method for a rule, but you can use them in any way that meets your requirements.

(2) Associated Code System

A second important attribute of a concept mapping specification in OpenCDS is that it may be associated with one particular code system. This means that there may be several distinct instances of the OpenCDS Concept for different code systems, such as LOINC, CPT, ICD9, ICD10, SNOMED-CT, RXNORM, proprietary in-house systems, and so on.

To summarize, an OpenCDS concept mapping specification will be associated with an OpenCDS Concept, and will identify a particular instantiation of that OpenCDS Concept, with variations for different determination methods and possibly for different code systems. Typically the name of the concept mapping specification will include a reference to the clinical concept, the determination method, and possibly the code system.

It should be noted that the name of a concept instance has no computable value, it is informative for human consumption. While we try to give meaningful names to the files, all of the files in the directory are loaded by OpenCDS at startup, regardless of the file name. All of the computable knowledge is specified by XML within the concept instance.

4. Concept Mapping Instances

a) Definition:

Concept mapping instances in OpenCDS relate a concept mapping specification to one or more codes from the one or more associated Code System(s) as specified by the associated Determination Method.

A concept mapping instance will include a list of CD elements (code system and code) that imply or specify the clinical concept (aka OpenCDS Concept) associated with the concept mapping specification. They are determined by the associated determination method, and may or may not all be from one associated code system.

In the current version of OpenCDS, these mappings are stored as files, but a future version will store them in a database. In either case, the place where they are stored, and from which they can be retrieved is known as a **Knowledge Repository**.

In many use cases the mappings will be created once, and then updated on a relatively infrequent basis (this is the default in OpenCDS). In other use cases (particularly with pharmaceuticals), there may be a need to have a live connection to a terminology server to query the latest mappings at run-time, because they change so frequently. Support for doing this live query against a terminology service (e.g., the open-source Apelon DTS service) will be included in a future release of OpenCDS.

For OpenCDS, the mappings stored as files can be created using any of the following methods:

- Apelon DTS with the built-in interface included in OpenCDS to produce XML files,
- any other terminology management system that can be configured to produce XML files, or
- by hand-crafted XML files.

Auto-generated concept mappings are stored separately from manually created ones (i.e., files constructed by any method other than download from Apelon), to avoid inadvertent over-writing of hand-crafted concept mappings by mappings from Apelon.

Concept mappings are intended to be maintained locally by multiple contributors in a distributed fashion, similar to knowledge modules. This is in contrast to the OpenCDS Concepts, which are intended to be maintained centrally as described above.

B. Concepts in Guvnor

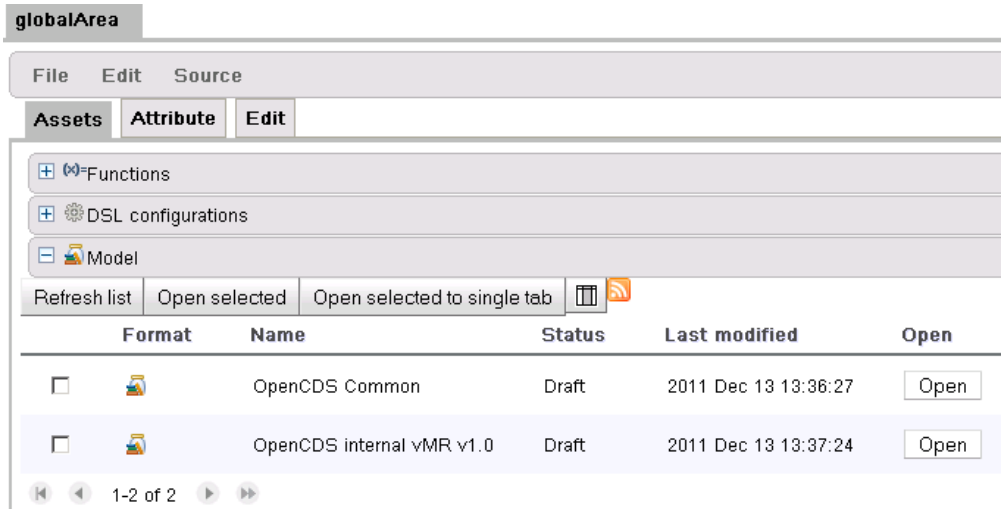
While it is possible to write rules for OpenCDS without using Guvnor, and without referring to OpenCDS Concepts at all, **we support Guvnor and strongly recommend writing your rules in reference to OpenCDS Concepts.** The combination of Guvnor and OpenCDS Concepts produces rules that have visible and easily understood logic, and can be readily reviewed for accuracy by a clinician.

- Guvnor is a tool for doing two things:
  - writing rules (which we sometimes refer to as "knowledge" about "clinical concepts"), and
  - testing the logical validity of those rules

It is a best practice to write thorough tests in Guvnor for your rules, and to run them whenever there is any change to the rules in a KnowledgeModule (aka a "Package" in Guvnor). This can be done in Guvnor with a single click to run all the tests for a single package.

1. Concept Types

The same Java classes that define Concept Types in OpenCDS are imported into Guvnor using the jar file from the OpenCDS module named "**opencds-vmr-v1_0-internal**." This then gives you exactly the same definitions of Concept Types that are used in the web service. You will also need the OpenCDS Common jar from the OpenCDS module named "**opencds-common**.":



If you select the "Edit" tab instead of the "Assets" tab, you will be able to view (and edit) the contents of the model:

If you click the "Advanced View" button, you will be able to edit the model as text, and you may need to do this:

When you first create a new model in Guvnor by importing the two jars from OpenCDS, this list will be in a random order, and it will include some elements from the common jar that you don't need. We provide a sorted list of the import statements, which you can paste into this window after deleting the generated content. This sorted list is maintained in the OpenCDS source code as a resource named "**ImportStatementsForOpenCDSGuvnor.txt**" within the **opencds-vmr-1_0-internal module**.

Don't forget to validate, build, and save the model after anytime you make changes to it.

Once you have the model defined, leave it alone! Changing the model may break rules that are based on the model.

2. Guvnor Enumerations start with OpenCDS Concepts

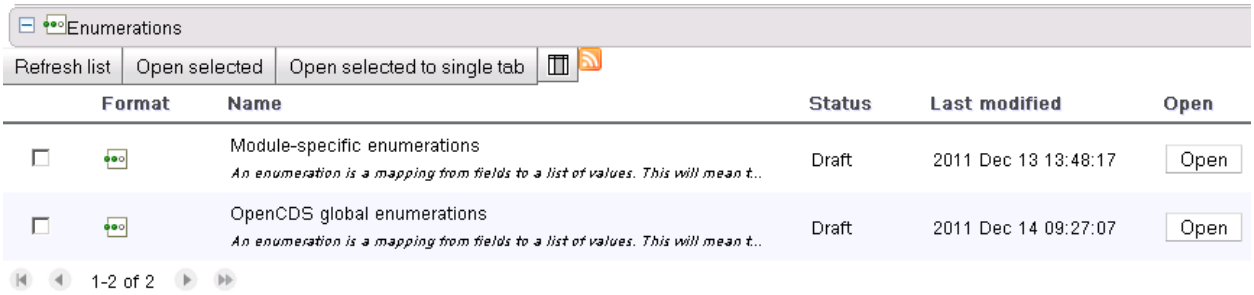
There is a utility in the opencds-decision-support-service module named "**org.opencds.terminology.GuvnorEnumerationCreator**.java" which creates a list of enumerations that can be placed into Guvnor Package Enumeration assets by using copy and paste.

Enumerations created by this utility may have been defined for each OpenCDS Concept Type (or a subset of all possible Concept Types), and they "enumerate" all of the OpenCDS Concepts available for each Concept Type.

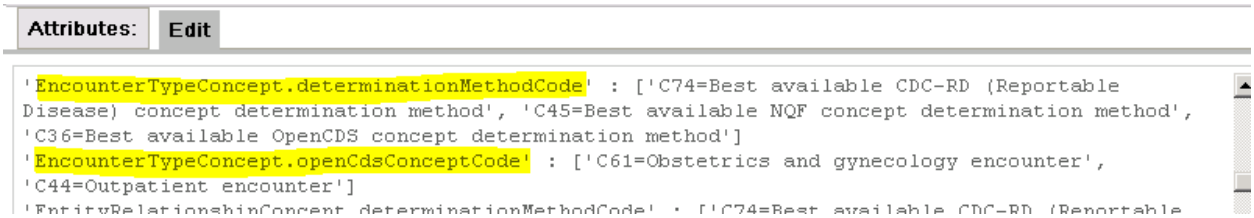
It is possible to add locally created enumerations, as well, but you should use the OpenCDS ones as a base. Local enumerations can be created manually in Guvnor, if needed. Note that you will eventually need to manually create run-time Concept mapping specifications to match any manual enumerations you create in Guvnor.

We do ***not*** provide a tool for downloading locally defined enumerations from your local instance of Apelon DTS to the run-time environment of OpenCDS Decision Support Service.

A typical package in Guvnor (aka a KnowledgeModule in OpenCDS) should have the following Enumerations:



If you click the "Open" button from one of the enumerations listed, you will see something that looks like this:



Every enumeration listed will have a "determinationMethodCode" entry, which is generated by the tool. Those which have an associated "openCdsConceptCode" will list those enumerations as well.

Refer to the official Guvnor documentation, and to our document "**OpenCDS Guvnor 5.3 DSL Best Practices**" for more information.

3. Domain Specific Language (DSL)

Enumerations make it possible to write rules using DSLs. These DSLs, when properly designed, may make reference to a Guvnor Enumeration, which then populates a drop-down list in the rule, listing all the OpenCDS Concepts that are included in the enumeration. DSLs are essentially a macro language that allows the use of plain clinical statement language that is non-technical, and should use a similar language to that a clinician would use to describe what the rule is supposed to do.

The addition of enumerations for drop-downs makes the DSLs portable, and one well-designed DSL can be used over and over in any number of rules, and do something different each time.

For example, the same DSL could populate a section of any number of different rules, some of which might read as follows:

- Patient has had 3 inpatient encounters for asthma in the past 1 year
- Patient has had 2 outpatient encounters for diabetes mellitus in the past 6 months
- Patient has had 1 emergency encounter for acute myocardial infarction in the past 10 days
- Patient has had 1 well-child encounter for EPSDT in the past 1 year

More information about using DSLs is found in the OpenCDS document named "**OpenCDS Guvnor 5.3 DSL Best Practices**."

4. Concept Mappings in Guvnor

We don't think it is a good idea to reference specific codes in Guvnor. Your rules should be written against OpenCDS Concepts, and they should be tested against OpenCDS Concepts, without worrying about how the concepts will get populated at run-time while you are in Guvnor. Run-time mappings can then be updated or modified without changing the rules at all.

For this reason, it is a best practice to do final integration testing of OpenCDS in the run-time environment rather than Guvnor by using the OpenCDS end-to-end testing tool. You will need to build a library of test data instances. This library can be enhanced and resubmitted at any time, and should be used as a final validation of both updated rules and updated concept mappings.

C. Concepts in OpenCDS Decision Support Service

OpenCDS Decision Support Service is the run-time implementation of OpenCDS. Everything about it is designed to support the separation of clinical concepts as used in rules from the values in the data which is to be evaluated.

It has been said before, and I'll repeat it here: **Write your rules against concepts**, and **separate the mappings of the codes found in your data from the rules**. OpenCDS provides internal methods at run-time which link each concept instance reference in your rules to every data element which has a mapping to the same concept instance in the concept mappings.

1. Concept Types in OpenCDS Internal Data

Concept Types have already been pretty thoroughly discussed above. The important things to remember are that

- They are broad categories,
- They correspond to all the CD data elements in the vMR,
- They are relatively fixed (adding a new Concept Type requires a number of modifications to the OpenCDS software), and
- They are implemented as OpenCDS Concepts, maintained in Apelon DTS, and implemented by user-specified concept mapping specifications which are ultimately mapped to concept codes in terminology systems.

2. OpenCDS Internal Data, Concept Mapping Specifications, and Concept Mapping Instances

OpenCDS DSS maps input data to what we call the internal vMR structure. While the external structure (the vMR schema) provides options to submit data in either listed (relational) structure or nested (object) structure, the internal structure is strictly the listed (relational) structure.

This requires mapping the external data into the internal structure (aka Drools fact lists). As the final step of this process, each CD data element that has been added to a fact list is checked against the concept mapping instances that are referenced by the requested KnowledgeModule[s].

If a concept mapping instance is found to have a code (and code system) match to the input data element, OpenCDS adds a reference to the matched OpenCDS Concept to the Drools fact lists so that your rules can reference it.

The following graphic illustrates how the mapping to the concept mapping instance produces a linked concept in the Drools fact lists:

	Drools Fact List	code	id
Input Data:	ProblemCode	codeSystem = ICD9 code 493.2	Problem id = 1.2.3^problem001
OpenCDS concept mapping instance: ICD9-asthma-bestAvailable:		493.1, 493.2, 493.3, 493.9	
Concept linked to data:	ProblemConcept	C39 = Asthma	targetId = 1.2.3^problem001

Note that it is possible to have multiple matches, or no matches at all. One particular ICD9 or SNOMED-CT code could represent more than one OpenCDS Concept, depending on what concept mapping specifications have been defined in a particular use case and installation.

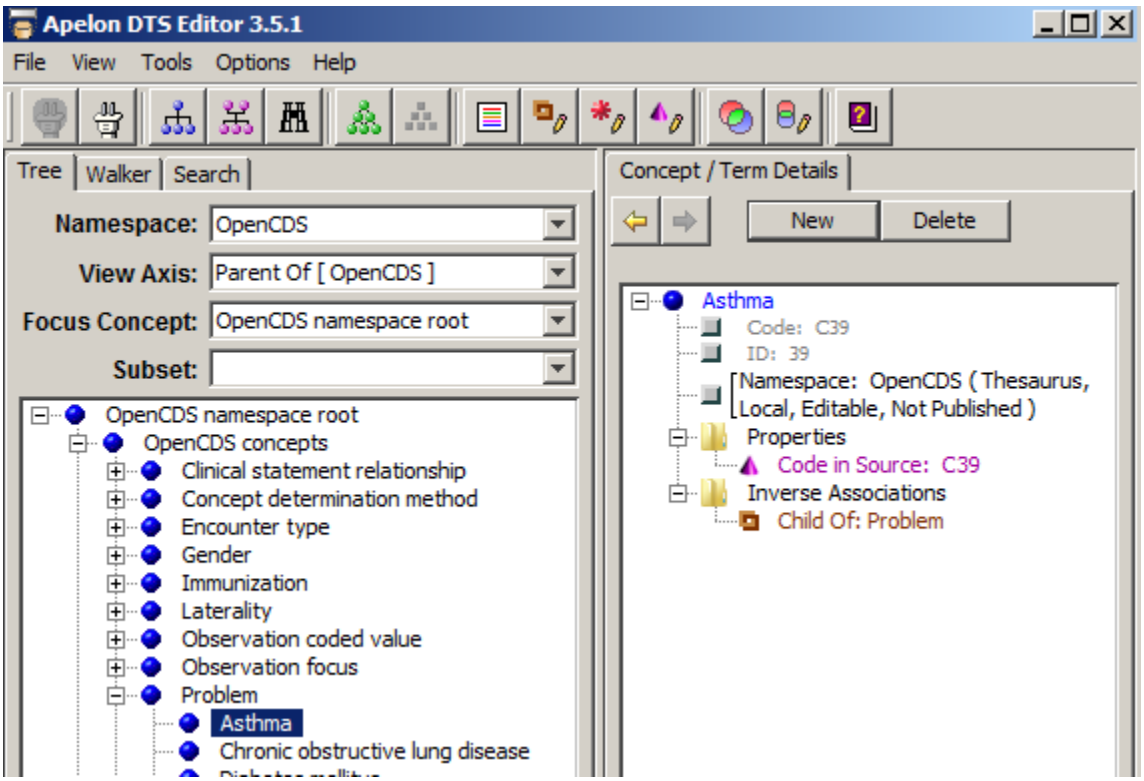
For example, if you had defined a simple concept mapping specification of asthma, another of chronic asthma, and a third mapping specification of pulmonary disease, then a data element with a code for episodic asthma would only link to the concept of asthma and the concept of pulmonary disease. A data element for emphysema would only link to breathing difficulty. A data element for chronic asthma would link to all three concept instances.

The following chart may make this example clearer:

Concepts versus Data	ProblemCode = 493.1 Intrinsic Asthma	ProblemCode = 493.2 Chronic Obstructive Asthma	ProblemCode = 510 Emphysema
Chronic Asthma		True	
Asthma	True	True	
Pulmonary Disease	True	True	True

3. Updating OpenCDS Concept Mapping Specifications and Mappings

OpenCDS concept mapping specifications and mappings can be maintained in Apelon DTS. Here is a screen shot showing maintenance of the OpenCDS Concept for Asthma in Apelon DTS:



OpenCDS provides a tool to download them from Apelon DTS, named "OpenCDSConceptsFileCreator." Instructions for the use of this tool are described earlier in the document, in Section A.2.

Concept mapping specifications and mapping instances can alternatively be maintained manually in XML files. This may be adequate for users with a limited number of concepts that have relatively stable code values.