DDI – Cross Domain Integration: Detailed Examples and Use Cases

Contents

I. Overview

This document provides a set of detailed examples from the DDI – Cross Domain Integration (DDI – CDI) Data Description and Process specifications. Some of the examples use the XML representation of these specifications. Other examples paint a picture of how the XML representations might be transformed to tell the same story with other standards and formats. The intent here is to create representations at a level of specificity that will provide guidance to users who want to exercise the DDI - CDI standard in realistic situations. The first four examples together tell a data story, and this story has a context:

The Karonga Health and Demographic Surveillance System (Karonga HDSS) in northern Malawi currently has a population of more than 42 000 individuals under continuous demographic surveillance since completion of a baseline census (2002-2004). The surveillance system collects data on vital events and migration for individuals and for households. It also provides data on cause-specific mortality obtained by verbal autopsy for all age groups, and estimates rates of disease for specific presentations via linkage to clinical facility data. The Karonga HDSS provides a structure for surveys of socioeconomic status, HIV-prevalence and incidence, sexual behavior, fertility intentions and a sampling frame for other studies, as well as evaluating the impact of interventions, such as antiretroviral therapy and vaccination programs. Uniquely, it relies on a network of village informants to report vital events and household moves, and furthermore is linked to an archive of biological samples and data from population surveys and other studies dating back three decades.

Here is the data story:

- It begins with a DDI - CDI description of the HDSS event history data model that is able to capture both demographic and health events
- It continues with a description of the data workflow that loads HDSS operational data stores and their entities into the HDSS event history data model
- It continues with a description of the metadata workflow that reformats and turns the HDSS event history data model into a schema.org Dataset description
- It ends with a schema.org Dataset description of the HDSS event history data model produced by the metadata workflow above

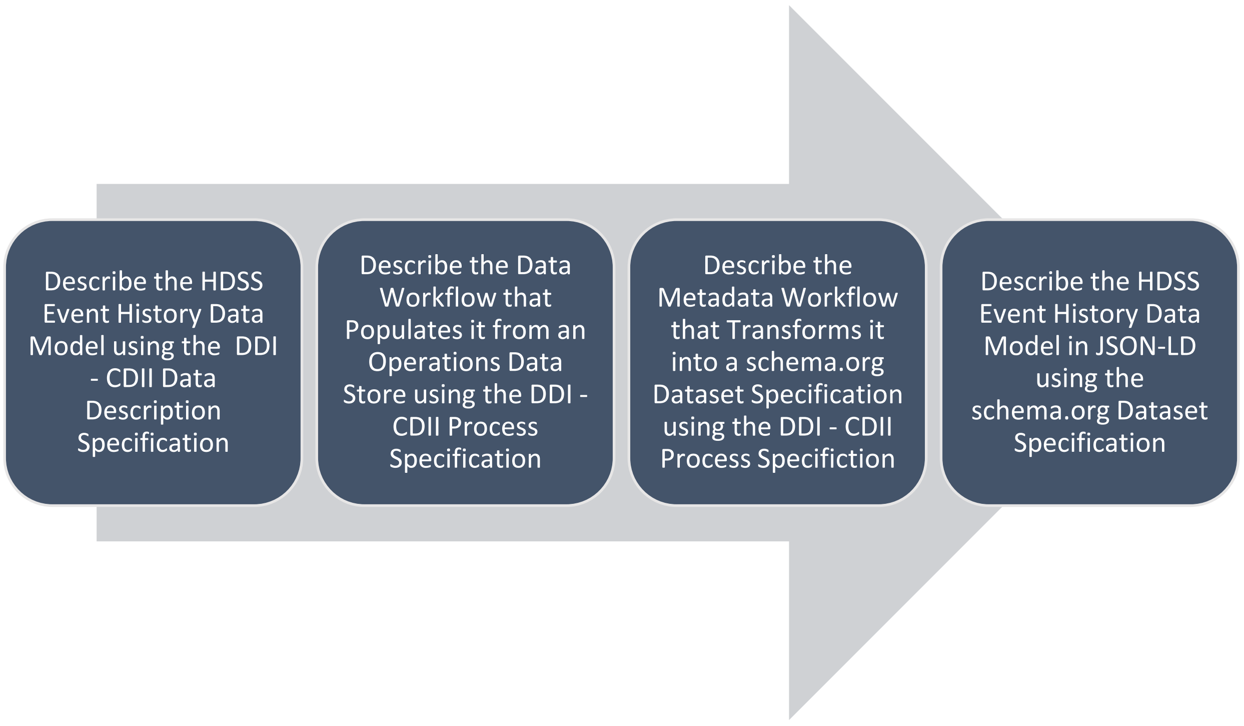

Describe the HDSS Event History Data Model using the DDI - CDII Data Description Specification
Describe the Data Workflow that Populates it from an Operations Data Store using the DDI - CDII Process Specification
Describe the Metadata Workflow that Transforms it into a schema.org Dataset Specification using the DDI - CDII Process Specification
Describe the HDSS Event History Data Model in JSON-LD using the schema.org Dataset Specification

Figure 1: The Karonga HDSS Data Story Told in Four Examples

Additionally, there is an example of how a time series can be described using the DDI - CDI Data Description specification

II. The HDSS Event History Data Model Example

In DDI - CDI, following GSIM, a DataSet has a DataStructure, and the DataStructure has DataStructureComponents:

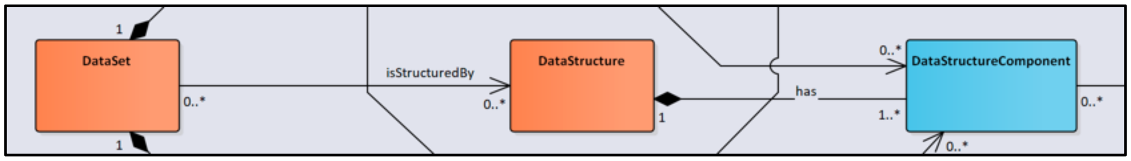


Figure 2: DDI - CDI DataSet Composition

DDI - CDI includes a rich set of components because it is intent on describing many types of DataSets that are encountered in research across many domains:

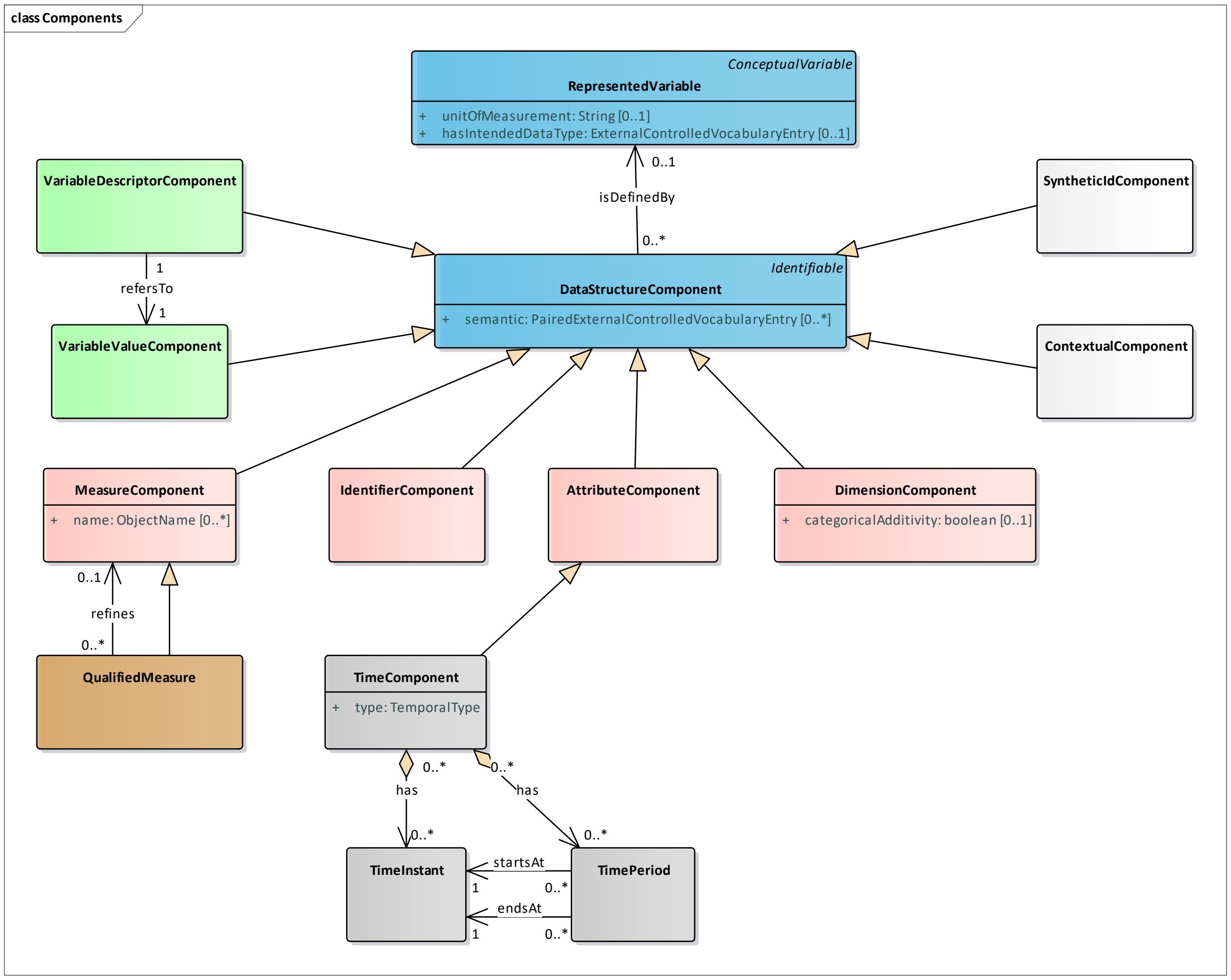


Figure 3: DDI - CDI DataStructure Components

In DDI - CDI Data Description the HDSS Event History Dataset has a LongDataStructure composed of a subset of the DataStructure components from Figure 3. With a LongDataStructure there is typically just one MeasureComponent per record[1]:

[1] There are two canonical variations of the LongDataSet and its LongDataStructure. In one variation, as happens with event histories, the measure does not change from one record to the next. In another variation, like what happens in the RAIRD Information Model, while each record has just one measure, that measure may change from one record to the next. In this variation the LongDataStructure, in place of a MeasureComponent, has a VariableDescriptorComponent and a VariableValueComponent. The VariableDescriptorComponent identifies a variable and the VariableValueComponent takes any value.

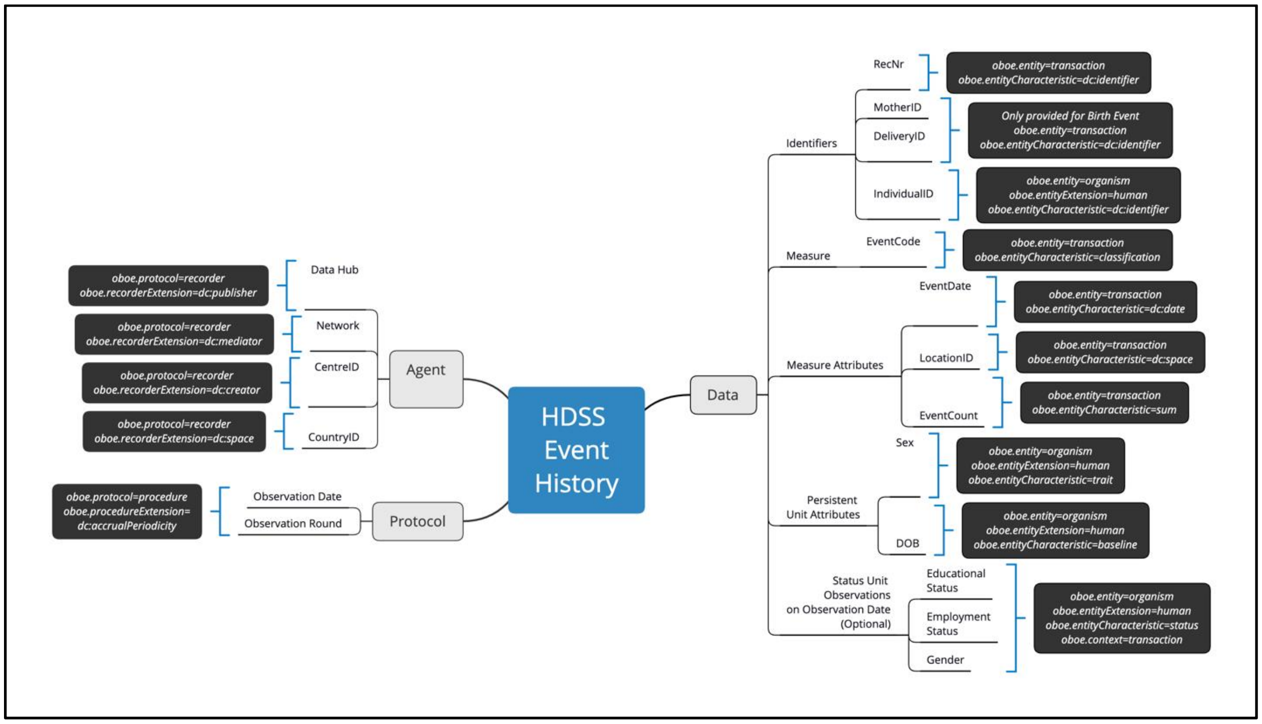

HDSS Event History
Agent
Data Hub
oboe.protocol=recorder oboe.recorderExtension=dc:publisher
Network
oboe.protocol=recorder oboe.recorderExtension=dc:mediator
CentreID
oboe.protocol=recorder oboe.recorderExtension=dc:creator
CountryID
oboe.protocol=recorder oboe.recorderExtension=dc:space
Protocol
Observation Date
Observation Round
oboe.protocol=procedure oboe.procedureExtension= dc:accrualPeriodicity
Data
Identifiers
RecNr
oboe.entity=transaction oboe.entityCharacteristic=dc:identifier
MotherID
DeliveryID
Only provided for Birth Event oboe.entity=transaction oboe.entityCharacteristic=dc:identifier
IndividualID
oboe.entity=organism oboe.entityExtension=human oboe.entityCharacteristic=dc:identifier
Measure
EventCode
oboe.entity=transaction oboe.entityCharacteristic=classification
Measure Attributes
EventDate
oboe.entity=transaction oboe.entityCharacteristic=dc:date
LocationID
oboe.entity=transaction oboe.entityCharacteristic=dc:space
EventCount
oboe.entity=transaction oboe.entityCharacteristic=sum
Persistent Unit Attributes
Sex
oboe.entity=organism oboe.entityExtension=human oboe.entityCharacteristic=trait
DOB
oboe.entity=organism oboe.entityExtension=human oboe.entityCharacteristic=baseline
Status Unit Observations on Observation Date (Optional)
Educational Status
Employment Status
Gender
oboe.entity=organism oboe.entityExtension=human oboe.entityCharacteristic=status oboe.context=transaction

Figure 4: HDSS Event History DataSet DataStructure

Each observation in an HDSS Event History is an event with an event code; one or more event identifiers; some characteristics of the event including the time of the event and its location; some characteristics of the entity (unit) being observed that do not change from one event to the next in principle like sex and date of birth; some characteristics of the entity (unit) being observed that might change from the time of one observation to the next like educational status and employment status; and some characteristics of the protocol by way of which the event was observed like who observed the event and when the event was observed.

Surrounding the MeasureComponent in an HDSS Event History Dataset, then, there are one or more IdentiferComponents and a set of AttributeComponents specific to different entities that participate in the event history including the event itself, the person in general, the person at a point in time and the recorder/recording of the event.

As a consequence, in order to capture these characteristics, the IdentifierComponent(s), MeasureComponent and the AttributeComponents in an HDSS Event History are marked up semantically. DDI - CDI is indifferent to the markup language or, again, the ontology that is employed. Instead, in DDI - CDI, just like in other DDI products, there is a PairedExternalControlledVocabularyEntry which in DDI - CDI has been associated with a DataStructureComponent:

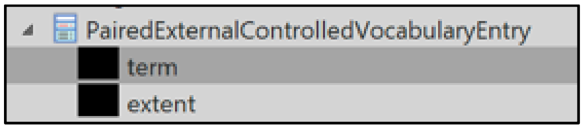

PairedExternalControlledVocabularyEntry
term
extent

Figure 5: A PairedExternalControlledVocabularyEntry

Both the term and the extent in the pair are each an ExternalControlledVocabularyEntry:

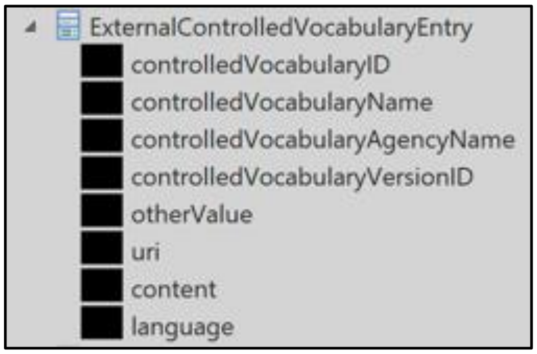

ExternalControlledVocabularyEntry
controlledVocabularyID
controlledVocabularyName
controlledVocabularyAgencyName
controlledVocabularyVersionID
otherValue
uri
content
language

Figure 6: An ExternalControlledVocabularyEntry

The PairedExternalControlledVocabularyEntry is the data type of the semantic of all DataStructureComponents.

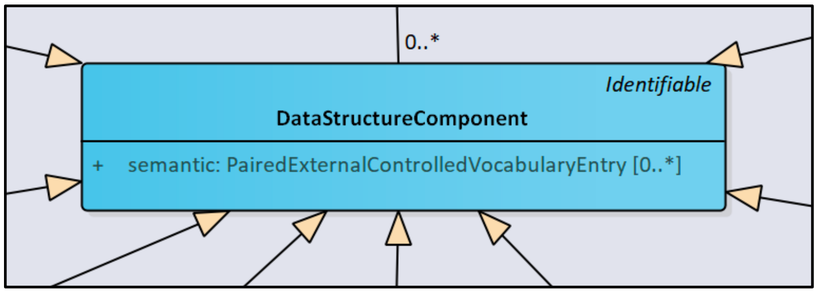


Figure 7: PairedExternalControlledVocabularyEntry is the data type of the semantic

The PairedExternalControlledVocabularyEntry is used by DDI - CDI to semantically mark up each DataStructureComponent. Each DataStructureComponent can take a succession of PairedExternalControlledVocabularyEntries in effect creating a controlled vocabulary with a hierarchical structure or, again, a taxonomy.[2]

[2] See [Taxonomies and controlled vocabularies best practices for metadata]() by Heather Hedden for an in depth discussion of the use of taxonomies. Here she says: The word 'taxonomy' means the science of classifying things, and traditionally the classification of plants and animals, as in the Linnaean classification system. It has become a popular term now for any hierarchical classification or categorization system. Thus, a taxonomy is a controlled vocabulary in which all the terms belong to a single hierarchical structure and have parent/child or broader/narrower relationships to other terms. The structure is sometimes referred to as a 'tree'. The addition of non-preferred terms/synonyms may or may not be part of a taxonomy.

Although DDI - CDI is “indifferent” to the markup language selected, the one that is employed here with the HDSS Event History Dataset is called the Extensible Observation Ontology (OBOE). OBOE, simply put, describes entities being observed and their characteristics:

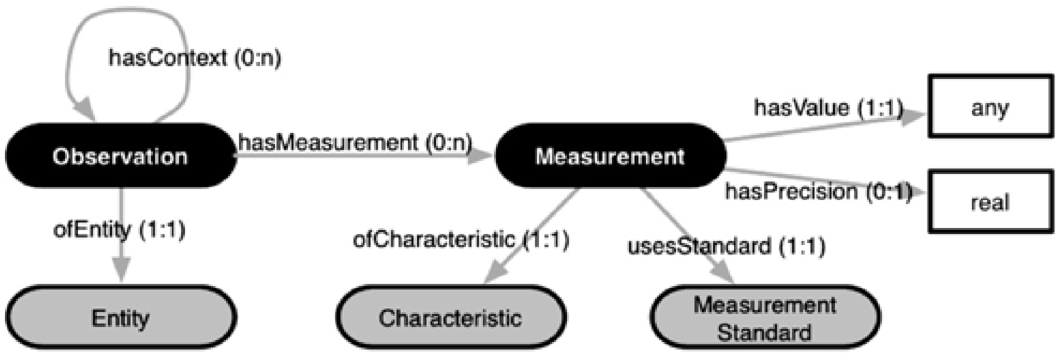


Figure 8: The core classes (ellipses) and properties (arrows) of the Extensible Observation Ontology (OBOE). Each Observation is of some Entity and can provide context for the Observation of another Entity. A Characteristic of an Entity can be represented through a Measurement. Measurements relate Characteristics to a Measurement Standard via a Value and, if applicable, a Precision. Measurements are taken by a Recorder (human or non-human) using a Protocol at a particular Time and Place

OBOE is a so-called “observation” ontology used in ecological research. OBOE was recently aligned with the Semantic Sensor Network Ontology. OBOE markup for the HDSS Event History Dataset appears in the black boxes of Figure 4 above.

An XML representation of the HDSS Event History Data Model used by many HDSSs across Sub-Saharan including Karonga can be found here[3].

III. An HDSS Data Workflow Example

In this example just a fragment of the actual HDSS data workflow is described. The fragment contains three Activities each of which contain several Steps:

[3] This example XML references two other files – DDI_CDI.xsd and xml.xsd. All three files need to be placed in the same directory.

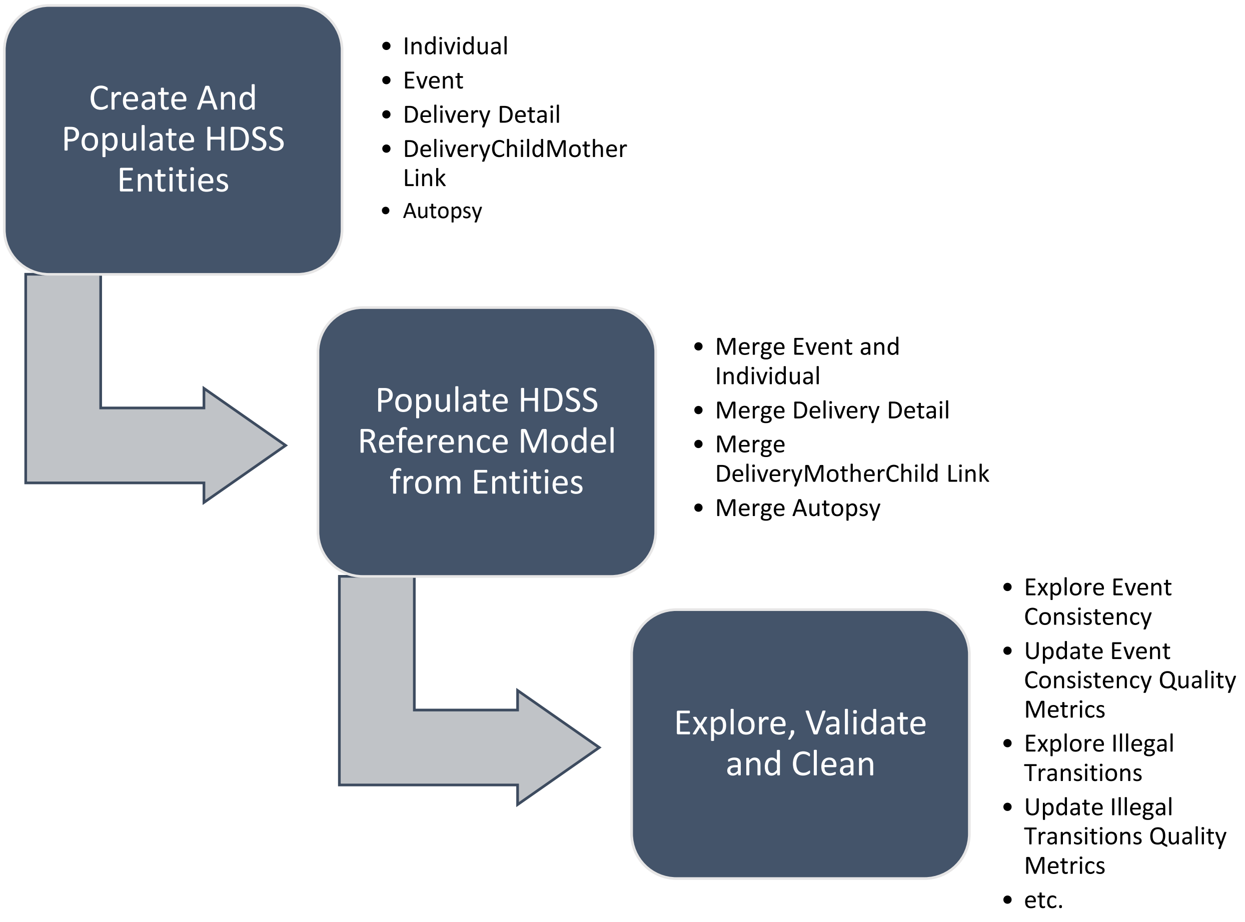

Create And Populate HDSS Entities
Individual
Event
Delivery Detail
DeliveryChildMother Link
Autopsy
Populate HDSS Reference Model from Entities
Merge Event and Individual
Merge Delivery Detail
Merge DeliveryMotherChild Link
Merge Autopsy
Explore, Validate and Clean
Explore Event Consistency
Update Event Consistency Quality Metrics
Explore Illegal Transitions
Update Illegal Transitions Quality Metrics
etc.

Figure 9: The HDSS Pentaho Data Workflow (Fragment)

The succession of Activities is sequential. Within the *Create and Populate HDSS Entities* Activity the succession of Tasks runs in parallel. Within both the *Populate HDSS Reference Model from Entities* and *Explore, Validate and Clean* Activity the succession of Tasks is sequential.

Finally, each Task uses and/or produces one or more InformationObjects with the execution of each of these Tasks conditioned by the existence of the InformationObject(s) it uses. In this example, parameters and the pathways they follow from one Task to the next are not utilized. Instead, Tasks communicate by checking the existence of InformationObjects in the data store. In fact, this is the approach HDSSs are taking in Pentaho. Had a different platform and/or approach been taken, the description might have utilized parameters and pathways. Both approaches are supported in the DDI Process specification.

A DDI - CDI XML representation of this data workflow can be found here.

IV. A Metadata Workflow Example

This workflow is a future. It's idea, however, grows out of an actual product that was built by members of the DDI community – DDI2R. DDI2R constructs an R class library based on a DDI - CDI profile. Along the way it has been determined that perhaps the "right" architectural solution for DDI2R is to build an intermediate class library in Python which might be used to construct a family of products including DDI2R. The workflow below is based on that idea:

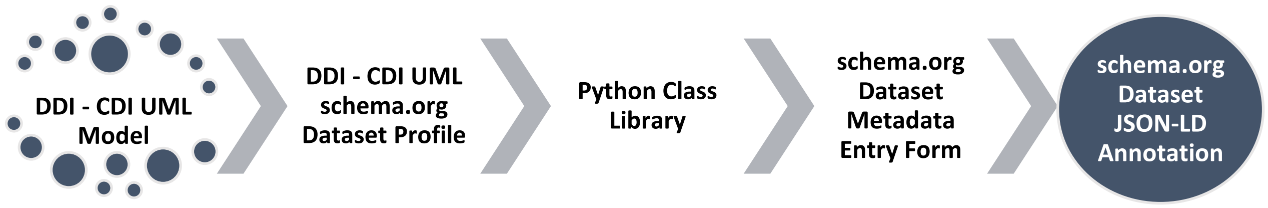

DDI - CDI UML Model
DDI - CDI UML schema.org Dataset Profile
Python Class Library
schema.org Dataset Metadata Entry Form
schema.org Dataset JSON-LD Annotation

Figure 10: DDI2JSON-LD

DDI2JSON-LD constructs a schema.org Dataset annotation organizations might use to make the datasets they publish on the web discoverable by Google Dataset Search.

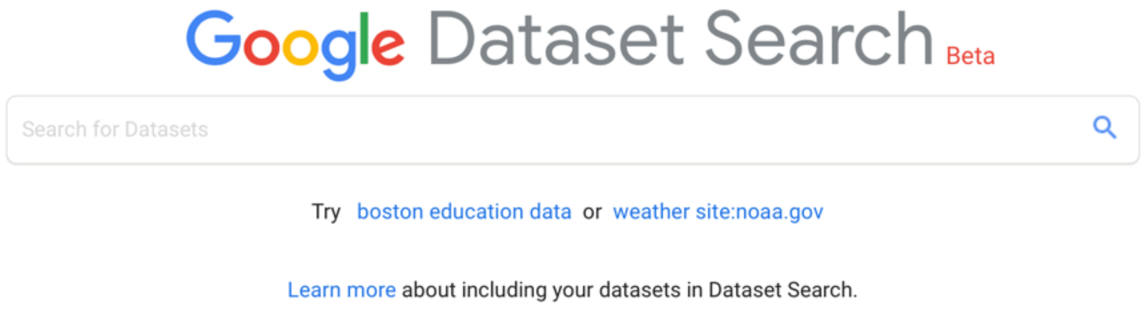

Google Dataset Search Beta
Search for Datasets
Try boston education data or weather site:noaa.gov
Learn more about including your datasets in Dataset Search.

Figure 11: Google Dataset Search

Recall that in the recent past Google launched Dataset Search “so that scientists, data journalists, data geeks, or anyone else can find the data required for their work and their stories, or simply to satisfy their intellectual curiosity.” According to Google, “Dataset Search lets you find datasets wherever they’re hosted, whether it’s a publisher's site, a digital library, or an author's personal web page.”

In this workflow a schema.org Dataset metadata entry form is created which an organization can complete. From the completed form a JSON-LD annotation is produced that the organization can attach to each dataset the organization publishes on the web.

The form is notionally built using the Python Class Library which is, in turn, based on a DDI - CDI schema.org Dataset profile. That profile includes both variable-level objects from the DDI - CDI Data Description specification and objects from the so-called Upper Model. The Upper Model is not currently in scope for DDI - CDI. It provides the research context. A discussion of the research context and the Upper Model can be found in the Section IIC1 of the Architecture Document.

The actual workflow depicted in Figure 9 is not very big. It has been rendered [here](#) in XML using the DDI - CDI process model.

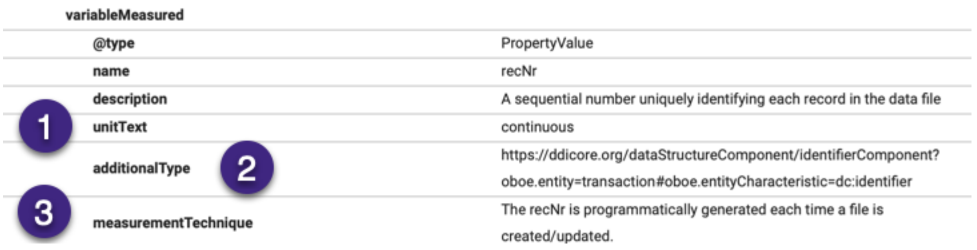
V. The Karonga HDSS schema.org Dataset

In fact, the construction of this workflow that the Karonga HDSS might use to produce annotations for the datasets it publishes is a work in progress.

In the interim a generic annotation for the HDSS Event History Reference Model used by many HDSSs including Karonga to conduct demographic and epidemiological was constructed by hand. It can be found here.

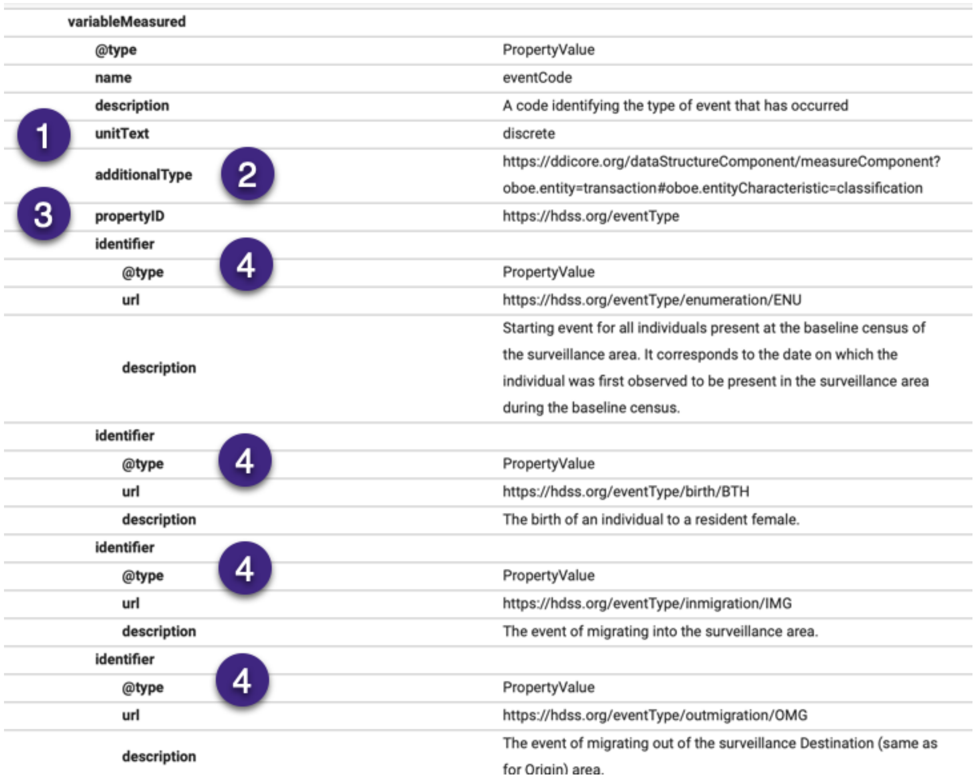
And here are a few illustrative schema.org Dataset variable descriptions taken from that annotation. They come from the Google Structured Data Testing Tool. This tool takes as input schema.org JSON-LD and produces as output a pretty description of the JSON-LD like a user would see during Google Dataset Search.

The illustrative variable descriptions are each annotated to facilitate a short discussion that follows each snapshot.



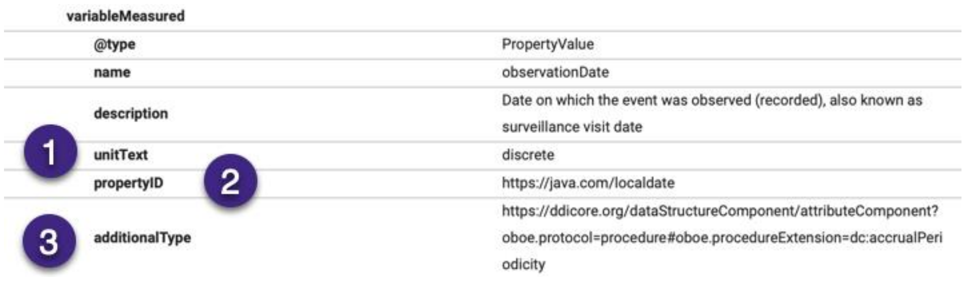
variableMeasured	
@type	PropertyValue
name	recNr
description	A sequential number uniquely identifying each record in the data file
1 unitText	continuous
additionalType 2	https://ddicore.org/dataStructureComponent/identifierComponent?oboe.entity=transaction#oboe.entityCharacteristic=dc:identifier
3 measurementTechnique	The recNr is programmatically generated each time a file is created/updated.

Figure 12: The recNr variableMeasured from the HDSS Event History Dataset is a "sequential number uniquely identifying each record in the data file". (1) unitText is a string or text indicating the unit of measurement. It is useful if you cannot provide a standard unit code for unitCode (from schema.org). (2) additionalType here both provides the DDI - CDI DataStructureComponentType of the variableMeasured – an IdentifierComponent – and its semantic markup using the Extensible Observation Ontology (OBOE). The OBOE entity measured by recNr is a transaction. And the entity characteristic measured by recNr is a Dublin Core identifier. (3) measurementTechnique is not always included. Here it is integral to the variableMeasured.



	variableMeasured	
	@type	PropertyValue
	name	eventCode
	description	A code identifying the type of event that has occurred
1	unitText	discrete
2	additionalType	https://ddicore.org/dataStructureComponent/measureComponent?oboe.entity=transaction#oboe.entityCharacteristic=classification
3	propertyID	https://hdss.org/eventType
	identifier	
4	@type	PropertyValue
	url	https://hdss.org/eventType/enumeration/ENU
	description	Starting event for all individuals present at the baseline census of the surveillance area. It corresponds to the date on which the individual was first observed to be present in the surveillance area during the baseline census.
	identifier	
4	@type	PropertyValue
	url	https://hdss.org/eventType/birth/BTH
	description	The birth of an individual to a resident female.
	identifier	
4	@type	PropertyValue
	url	https://hdss.org/eventType/inmigration/IMG
	description	The event of migrating into the surveillance area.
	identifier	
4	@type	PropertyValue
	url	https://hdss.org/eventType/outmigration/OMG
	description	The event of migrating out of the surveillance Destination (same as for Origin) area.

Figure 13: The eventCode variableMeasured from the HDSS Event History Dataset is a "code identifying the type of event that has occurred". (1) The unit of measurement is discrete. (2) additionalType here both provides the DDI - CDI DataStructureComponentType of the variableMeasured – a MeasureComponent – and its semantic markup using the Extensible Observation Ontology (OBOE). The OBOE entity measured by eventCode is a transaction. And the entity characteristic measured by eventCode is a classification. (3) The data type is an HDSS codelist called eventType. (4) is a partial enumeration of this codelist in which each entry consists of a label and a code.



	variableMeasured	
	@type	PropertyValue
	name	observationDate
	description	Date on which the event was observed (recorded), also known as surveillance visit date
1	unitText	discrete
2	propertyID	https://java.com/localdate
3	additionalType	https://ddicore.org/dataStructureComponent/attributeComponent?oboe.protocol=procedure#oboe.procedureExtension=dc:accrualPeriodicity

Figure 14: The observationDate variableMeasured from the HDSS Event History Dataset is a "date on which the event was observed (recorded), also known as surveillance visit date". (1) The unit of measurement is discrete. (2) The data type is a Java localdate. (3) additionalType here both provides the DDI - CDI DataStructureComponentType of the variableMeasured – an AttributeComponent – and its semantic markup using the Extensible Observation Ontology (OBOE). In OBOE observationDate belongs to the protocol. Within the protocol it is a procedure. More specifically it represents Dublin Core accrualPeriodicity.

VI. Cell-Oriented Time Series Example

DDI - CDI supports the descriptions of time series as part of a general multi-dimensional structure, where time is the dimension used to connect observations, as one among many. This is an OLAP-based approach to time series, but there are many systems in use today which do not handle time series in this fashion. This example uses a "cell-based" approach which is more appropriate for these systems – it demonstrates the flexibility of DDI - CDI when it comes to describing the data needed to support the needs of a particular implementation.

In this section, we provide an example for how to describe a time series. We will use the Urban Consumer Price Index (CPI-U) from the US Bureau of Labor Statistics.

CPI-U is a family of indexes, each available as a series going back many years. Some go back to 1913. Each index has a base year from which the current value is derived. The base years are mostly in the range 1982-1984. The value for the base year and period is set as 100.0.

Series data in general are dimensional. Unlike a cube where every dimensional combination (i.e., the combination of categories taken from each one of the dimensions) is represented, in a series we are interested in looking at one dimensional combination over time. One may conceive of this as looking at one cell in a cube taken over time. For the CPI-U, the dimensions available are Item (product or service) and Area (metropolitan statistical area). Each series in the CPI-U family represents the combination of one Item category and one Area category. It is possible to ignore one or the other dimension by selecting “all” instead of a specific entry in each dimension. Selecting the “all” category effectively collapses the dimension.

The CPI-U is published as an overall index for the entire urban US, for individual items (products and services), urban areas, and combinations of items and areas. Some series in the family are published as either seasonally adjusted or not. We will illustrate a non-seasonally adjusted time series for the CPI-U for Apparel in Washington, DC. Note, a seasonally adjusted series is a conceptually different series than a non-seasonally adjusted one. This adjustment is not a kind of revision for each number in the series. Seasonal adjustment affects the entire series.

The lists of items and areas, the Dimensions, are Code Lists as defined in DDI. Here are short illustrations of the Items and Areas dimension files:

Area

Code	Area
35A	Washington-Arlington-Alexandria, DC-VA
S35B	Miami-Fort Lauderdale-West Palm Beach, FL
S35C	Atlanta-Sandy Springs-Roswell, GA
S35D	Tampa-St. Petersburg-Clearwater, FL
S35E	Baltimore-Columbia-Towson, MD
S37A	Dallas-Fort Worth-Arlington, TX

Item

Code	Item
SA311	Apparel less footwear
SAA	Apparel
SAA1	Men's and boys' apparel
SAA2	Women's and girls' apparel
SAC	Commodities
SACE	Energy commodities

The codes and categories are listed in the order they appear in the BLS files.

The table below contains CPI-U for Apparel (code – SAA) in Washington, DC (code – 35A) as a non-seasonally adjusted series going back 10 years for a bi-monthly period:

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2009	138.785		146.027		144.305		142.143		147.452		140.421	
2010	136.404		147.671		140.853		134.526		144.549		138.684	
2011	142.773		151.650		154.093		151.442		151.071		153.376	
2012	143.069		159.347		155.926		144.799		160.952		153.936	
2013	139.126		142.290		144.985		140.668		151.600		149.854	
2014	141.130		147.795		148.749		139.189		156.178		148.931	
2015	135.237		153.824		148.202		134.230		147.512		141.474	
2016	142.103		153.600		159.872		151.601		162.246		152.816	
2017	152.014		153.619		157.312		149.154		165.510		154.720	
2018	164.464		162.120		163.558		156.946		177.968		165.956	
2019	169.674		167.026		170.495		157.230					

The series includes the most recent estimate available at the time of the drafting of this document. Note also, the frequency of the estimates in this series is bi-monthly. Even though the CPI-U for all items and all areas (the US estimate) is issued every month, the data here do not support a monthly release for this index. Some other detailed indexes in this family are released every month.

Another issue about indexes is they cannot be compared across series. For example, the index for New York City for Apparel in July 2019 is 116.924 and that for Washington, DC is 157.230. This does not mean, however, that apparel is more expensive in Washington than in New York. Indexes are relative to the item and area they represent.