Software Architecture Document

Ontology Management Cell 1.4

Abstract:

This is a software architecture document for Ontology Management (ONT) cell. It identifies and explains important architectural elements. This document will serve the needs of stake holders to understand system concepts and give a brief summary of the use of the ONT message format.

Revision History

Revision Number	Date	Author	Description
1.01	8/30/07	Lori Phillips	Version 1.0
1.1	2/18/08	Shawn Murphy	Release Version 1.3
1.2	12/17/09	Lori Phillips	Release Version 1.4

Table of Contents

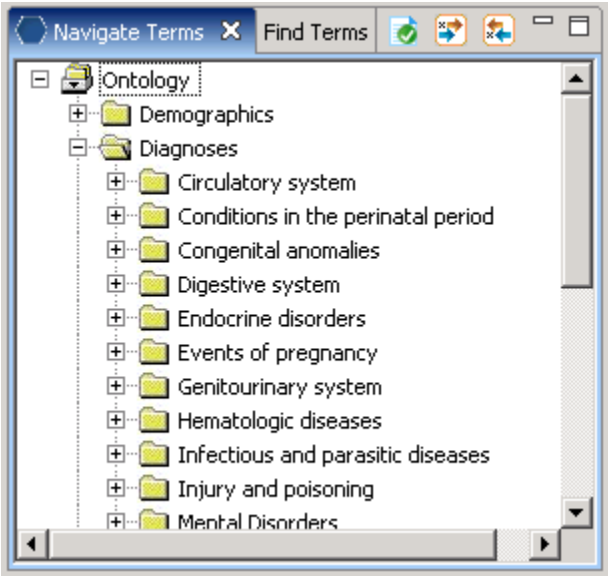
1. Overview 4
1.1. ONT Definitions, Acronyms and Abbreviations 5
1.1.1 Vocabulary Data Object (VDO) 5
1.1.1 Scheme 5
1.2 Roles 5
1.3 Security 5
1.4 Scope of the system 5
1.5 Assumptions/Constraints 6
1.6 Technical Platform 6
1.6.1 Transaction 6
1.6.2 Security 6
1.6.3 Persistence 6
1.6.4 Reliability/Availability 6
1.6.5 Performance 7
2. Use Case 7
2.1 Operations 7
3. Architecture Description 8
3.1 Components and Connector View 8
3.1.1. Client-Server Style 9
3.2 Module View type 11
3.2.1 Decomposition Style 11
3.2.2 Uses Style 12
3.3 Mappings of Styles 14
4. Data View 15
4.1 Selecting the Data Source 15
4.2 Schemas within the metadata data source 17
4.2.1. Table_Access table 17
4.2.2. Metadata tables 18
4.2.3. Schemes table 18
5. Deployment View 19
5.1 Global Overview 20
5.2 Detailed deployment model 20
References 21

1. Overview

The Ontology Management cell (ONT) is an i2b2 Hive Core cell. This cell manages i2b2 vocabulary definitions and contains concepts and information about relationships between concepts for the entire hive. It is accessed by other cells to give semantic meaning to data.

Vocabularies in the ONT cell are organized in hierarchical structures that represent the relationships between terms. The top levels in the hierarchy are called the 'parents' or 'roots', with the lower levels being their 'children'. Elements occurring on the same level are known as 'siblings'. A level in a hierarchy is sometimes referred to as a 'node', and a group of related data is called a 'category'.

A category is defined as a set of data for which there is a common rule or rules for querying against the Clinical Research Chart (CRC). A category is usually represented visually as a table of terms. An example of a category is the Diagnoses category shown in the diagram below, which consists of a table of diagnostic terms and uses a single rule to build all diagnostic queries.



Vocabularies in the ONT cell may originate as code from different sources. The ONT cell distinguishes these codes from one another by pre-pending a unique prefix to each code. Each distinct vocabulary and their associated codes is called a scheme.

1.1. ONT Definitions, Acronyms and Abbreviations

1.1.1 Vocabulary Data Object (VDO)

This object holds vocabulary definitions and information about the relationships between concepts.

1.1.1 Scheme

Each distinct vocabulary and their associated codes is called a scheme. A distinction is made between codes from different sources by pre-pending a unique prefix to each code.

1.2 Roles

The primary roles/participants in the ONT system are as follows:

- User – Create queries and access them only if he/she is owner to of the query.
- Manager – Create queries and can access queries created by different users within the project.

1.3 Security

Users may access ONT with a user-id and password combination, which is authorized through the Project Management Cell. The implementation detail of Project Management Cell is considered out-of scope to this document.

1.4 Scope of the system

Some other participants, currently outside the scope of ONT are:

- Project Management Cell

1.5 Assumptions/Constraints

- The Ontology metadata database shall not contain protected health information.

1.6 Technical Platform

The product is designed to run on the following platform:

- Java 2 Standard Edition 6.0 version 16
- Oracle Server 10g database
- Xerces2 XML parser
- JBoss Application server version 4.2.2.GA
- Spring Web Framework 2.0
- Axis2 v1.1 web service (SOAP/REST messaging)

1.6.1 Transaction

The ONT system is transactional, leveraging the transaction management model of the J2EE platform.

1.6.2 Security

The application must implement basic security behaviors:

- Authentication: Authenticate using at least a user name and a password
- Authorization: User may only access categories that they are allowed to by role
- Confidentiality: Sensitive data must be encrypted
- Data integrity : Data sent across the network cannot be modified by a tier
- Auditing: In the later releases we may implement logging of sensitive actions

1.6.3 Persistence

This application utilizes JDBC calls to retrieve persisted data.

1.6.4 Reliability/Availability

The Reliability/Availability will be addressed through the J2EE platform

Targeted availability is 16/7: 16 hours a day, 7 days a week

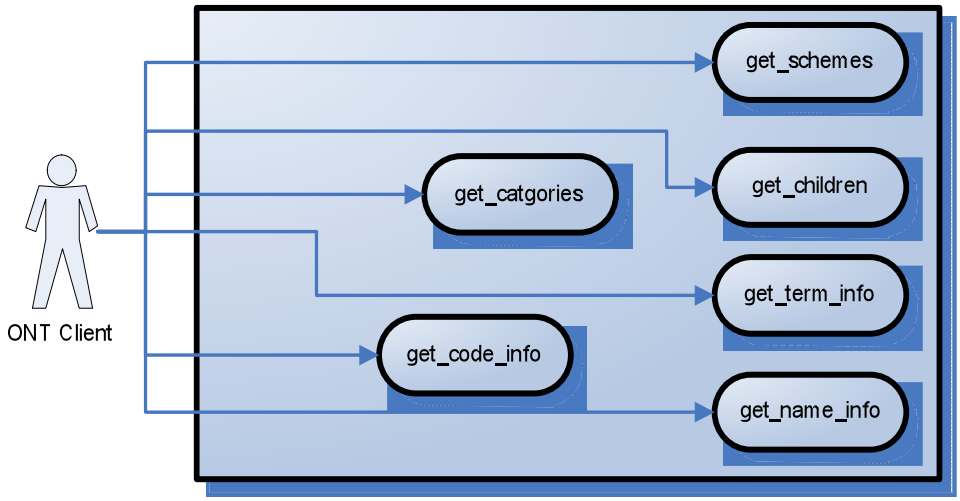
The time left (8 hours) is reserved for any maintenance activities

1.6.5 Performance

The user authentication with project management cell must be under 1 second.

2. Use Case

The diagram below depicts common use cases a user may perform with the ONT cell.



2.1 Operations

The ONT service is designed as a collection of operations, or use cases:

get_categories: returns a list of categories available for a given user. These categories are displayed in a tree format. The top level of the tree consists of all the categories a particular user has permission to see as determined by his/her role.

get_children: expands any level of a vocabulary category, providing information about its children, for a given user.

get_schemes: returns a list of schemes available in the system. This operation provides information about the different kinds of coding systems that exist.

get_name_info: returns information needed about all nodes related to a given search keyword or name.

get_code_info: returns information about all nodes related to a particular code.

get_term_info: returns information about a particular node.

3. Architecture Description

This section provides a description of the architecture as multiple views. Each view conveys the different attributes of the architecture.

1) Components and Connector View
    a) Client-Server Style
2) Module View
    a) Decomposition Style
    b) Uses Style
3) Data View
4) Deployment View

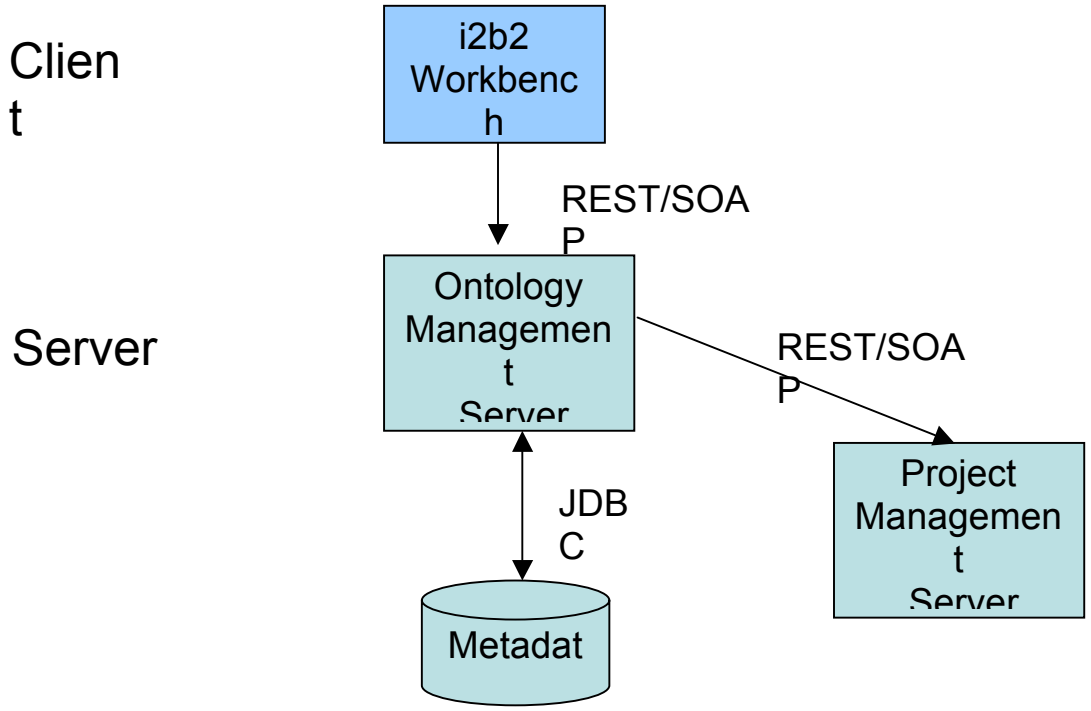
3.1 Components and Connector View

A Components and Connector view represents the runtime instances and the protocols of connection between the instances. The connectors represent the properties such as concurrency, protocols and information flows. Following diagram represents the Components and Connector view for the multi-user installation. As seen below, component instances are shown in more detail with specific connectors drawn in different notations.

3.1.1. Client-Server Style

The ONT system is represented using the C&C Client-Server view.

3.1.1.1 Primary Presentation



3.1.1.2 Element Catalog

Element Name	Type	Description
i2b2 Workbench	Client Component	Webservice client submits the requests to ONT Server components and renders response XML.
Ontology Management Server	Server Component	Provides Web Service Interface for the ONT system. It supports the REST or SOAP protocol. It directs the user to the correct data source associated with the project. It uses Project Management server to handle user authentication.
Project Management Server	Server Component	ONT cell uses Project Management cell to authenticate user. ONT cell constructs PM request message and makes a web service call to Project Management Cell.
Metadata	Data Repository Component	This repository is a database for i2b2 metadata.
JDBC	Query Connector	SQL query used as a connector between the ONT System and the Metadata database.
Web Service	Request Connector	REST protocol used to communicate with the external system.

3.1.1.2 Design Rationale, Constraints

N-tier Architecture

The client-server style depicts an n-tier architecture that separates the presentation layer from business logic and data access layer.

3.2 Module View type

The module view shows how the system is decomposed into implementation units and how the functionality is allocated to these units. The layers show how modules are encapsulated and structured. The layers represent the “allowed-to-use” relation.

The following sections describe the module view using Decomposition and Uses Styles.

3.2.1 Decomposition Style

The “Decomposition” style presents system functionality in terms of manageable work pieces. It identifies modules and breaks them down into sub-modules and so on, until a desired level of granularity is achieved.

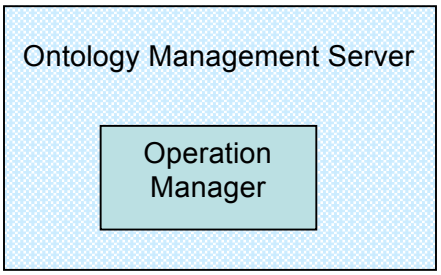
3.2.1.1 Primary Presentation

System	Segment
Ontology Management Server	Operation Manager

3.2.1.2 Element Catalog

Element Name	Type	Description
Operation Manager	Subsystem	This subsystem manages queries for ontology operations.

3.2.1.3 Context Diagram



3.2.2 Uses Style

The "Uses" style shows the relationships between modules and sub-modules. This view is very helpful for implementing, integrating and testing the system.

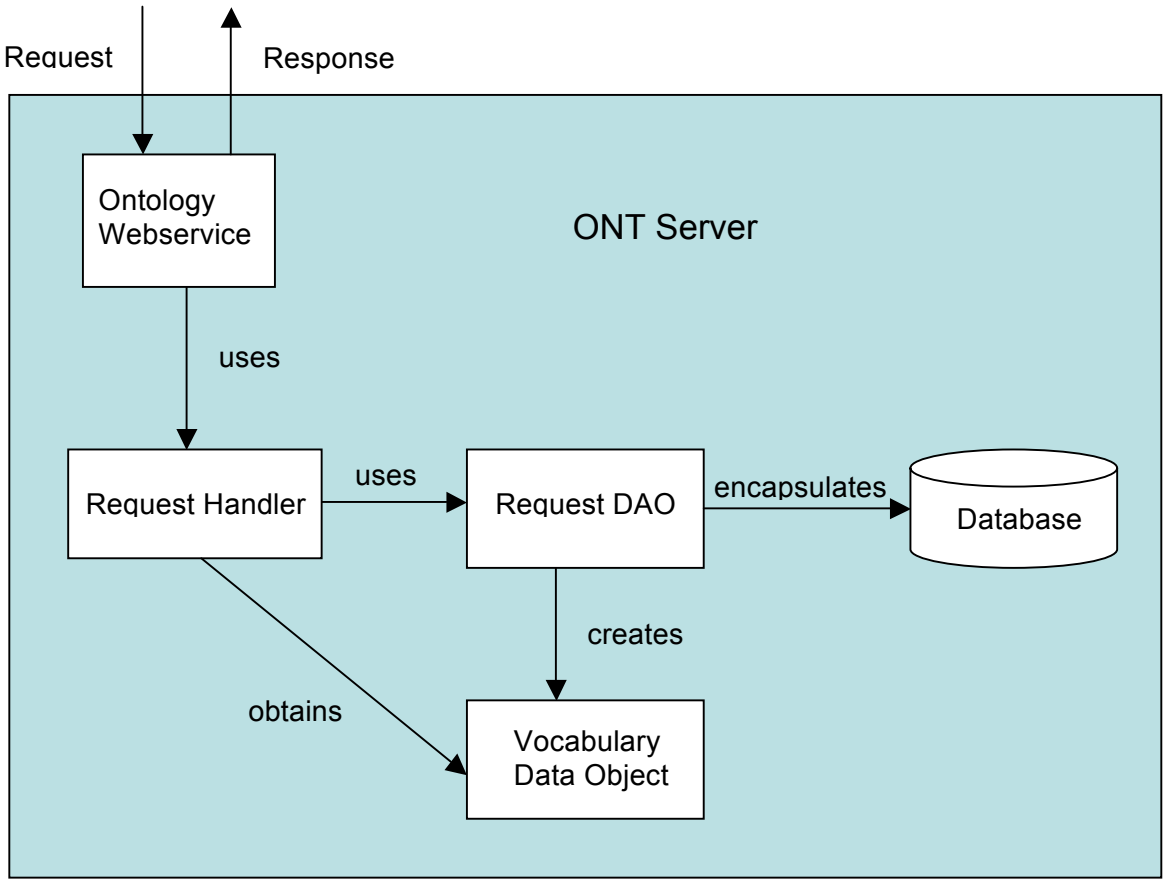
3.2.2.1 Primary Presentation

System	Segment
Ontology Management Server	ONT Module
Operation Manager Subsystem	Ontology Webservice
	Request Handler
	Request DAO
	Vocabulary Data Object

3.2.2.2 Element Catalog

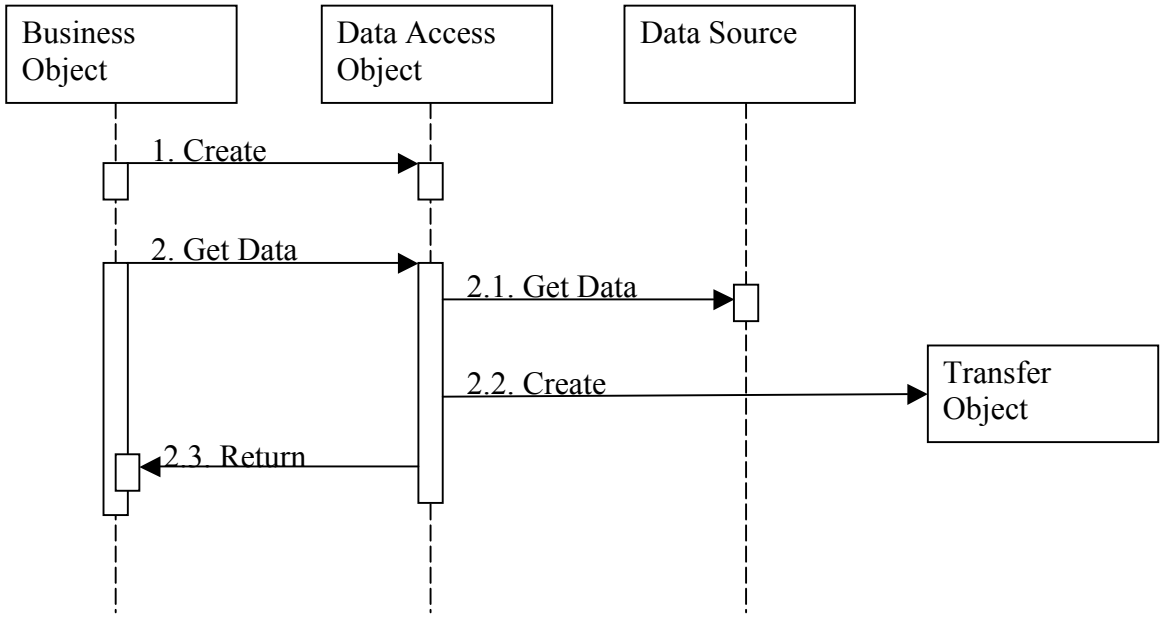
Element Name	Type	Description
ONT Module	Module	Authenticates user through PM Server System
Ontology Webservice	Communication Module	Provides web service interface to ontology operations.
Request Handler	Business Object	Delegates Ontology requests to Data Access Object layer to perform database operations.
Request DAO	Data Access Object	Supports database query operations.
Vocabulary Data Object	Transfer Object	Object representation of persisted data

3.2.2.3 Context Diagram



The Request Handler determines which Persistent Storage Location (PSL) is associated with a project request.

3.2.2.4 Sequence Diagram



3.3 Mappings of Styles

The following table is a mapping between the elements in the Component & Connector Client-Server view shown in section 3.1.1, and the Modules Decomposition and Uses views shown in sections 3.2.1 and 3.2.2.

The relationship shown is *is-implemented-by*, i.e. the elements from the C&C view shown at the top of the table are implemented by any selected elements from the Modules views, denoted by an "X" in the corresponding cell.

	ONT Server	Project Management Server	Metadata Database
ONT Service	X	X	
Ontology Webservice	X		
Request Handler	X		
Request DAO	X		X
Vocabulary Data Object	X		

4. Data View

4.1 Selecting the Data Source

Both the metadata and the patient data are distributed to projects through the existence of independent databases (in SQL Server) or schemas (in Oracle). These will be referred to in the rest of the text as the "persistent storage location" or PSL. These PSL's are organized so that the data from two metadata representations can be merged to a "Super" data set. While a person is working on a specific project, they will be directed to data in a PSL associated with that project.

In order to support the i2b2 project distribution strategy, the user is enrolled in numerous projects recorded within the i2b2 project management cell". The projects available to the user are returned in the web service call to the Project Management cell. The logic of selecting the correct PSL for the project is embodied in the following table:

DB_LOOKUP		
PK	**c_domain_id**	**VARCHAR(255)**
PK	**c_project_path**	**VARCHAR(255)**
PK	**c_owner_id**	**VARCHAR(255)**
	c_db_fullschema	**VARCHAR(255)**
	c_db_datasource	**VARCHAR(255)**
	c_db_servertype	**VARCHAR(255)**
	c_db_nicename	**VARCHAR(255)**
	c_db_tooltip	VARCHAR(255)
	c_comment	CLOB
	c_entry_date	DATE
	c_change_date	DATE
	c_status_cd	CHAR(1)

The logic for selecting the PSL is as follows:

1) There are two methods to select the correct PSL, an implicit one, and an explicit one. Both rely only on information available within the i2b2 header.
    a. The implicit one relies upon the data within the <domain> tag, the <username> tag, and the <project_id> tag.
    b. The explicit one relies upon the data only within the <project_id> tag. It has the format represented as the following string:

|"DOMAIN" | "Project"\"subproject"\"sub-sub Project"\| "USER_ID"|

These may not actually match the domain and username actually being used (since it is being built by the client), and must be checked when the PM cell is accessed.

2) The table is meant to provide a series of default locations if ones are not specifically listed. If a project is listed in the c_project_path column, then that PSL may be used, otherwise, a domain source will be used.
3) If a username is listed in the c_owner_id column, then if the project also matches the project_id, the PSL in that row may be used, otherwise, a project PSL will be used, and if a project PSL does not exist, the domain PSL will be used. For example, only if the domain|project|user_id is an EXACT match to the entries in the database will that PSL be used.
4) The project id may have associated sub projects that will be represented as project\sub-project\sub-sub-project string. If a sub-project is identified, but only the project exists in the table, the project PSL would be used.
5) The project may not have an entry in the table, and in that case any project (and sub-projects) would be designated the PSL of the domain.
6) If a general domain PSL is not available in the table, and only a specific project is associated with the domain in the table, then any incoming messages not associated with that project will return an error.
7) In the table, the "@" character is used to represent the absence of an entry (rather than a blank or a null).
8) In the explicit string and in the <project_id>, and "@" can be used to optionally represent a blank.

Other columns are specified as follows:

9) The column c_db_fullschema is used to contain the path to a table when the data source is used. Software is written so that the absence of the delimiter (usually a ".") does not need to be explicitly stated.
10) The column c_db_datasource is used to contain a short string that represents a data source configured in some other location.
11) The column c_db_servertype can be "ORACLE" or "SQLSERVER".
12) The column c_db_nicename is a string that can be used in client software to describe a data source.
13) The column c_db_tooltip contains a longer (hierarchical) representation of the nicename.

To restate, many cells need to access some kind of persistent storage, and these cells will organize their persistent storage so that it is self contained and can be apportioned in a way consistent with the project-based requirements of i2b2 described above. To that end, a table exists in many cells to make the decision of what persistent storage location to which a specific user will be directed, depending on the project and domain to which they are associated.

4.2 Schemas within the metadata data source

The following schemas provide data used by the ONT system:

4.2.1. Table_Access table

This table is used to obtain a list of "Metadata" tables within the Persistent Storage Location (PSL). Each Metadata table within the PSL is represented by a single row in this table. The primary identifier of each Metadata table in the Table_Access table is in the "c_table_cd" column. All messages that need to point to a specific Metadata table will use this identifier, for example in the <Key> element of many messages where this identifier is represented as \\c_table_name\ (see messaging specification). Within the c_table_name column is the actual name in the PSL of the Metadata table. The c_protected_access boolean column designates whether one must have the protected_access role to obtain data form this table. The other columns are defined in a similar manner to the columns of the Metadata table, with the following special notes:

The c_hlevel is almost always 0 in this table. The c_dimcode is used to allow the entire contents of a table to be queried in the data repository if the dimension table of the data repository has been set up in this fashion.

TABLE_ACCESS		
PK	**c_table_cd**	**VARCHAR(50)**
	c_table_name	**VARCHAR(50)**
	c_protected_access	CHAR(1)
	c_hlevel	**INT**
	c_name	**VARCHAR(2000)**
	c_fullname	**VARCHAR(900)**
	c_synonym_cd	**CHAR(1)**
	c_visualattributes	**CHAR(3)**
	c_tooltip	VARCHAR(900)
	c_total_num	INT
	c_basecode	VARCHAR(450)
	c_comment	CLOB
	c_metadataxml	CLOB
	c_facttablecolumn	**VARCHAR(50)**
	c_dimtablename	**VARCHAR(50)**
	c_columnname	**VARCHAR(50)**
	c_columndatatype	**VARCHAR(50)**
	c_operator	**VARCHAR(10)**
	c_dimcode	**VARCHAR(900)**
	c_entry_date	DATE
	c_change_date	DATE
	c_status_cd	CHAR(1)

4.2.2. Metadata tables

The Metadata table encapsulates the vocabulary used in the data repository. A Concept is a row from the Metadata table. It is the primary object used to pass vocabulary information to the requesting client. Typically a PSL will contain numerous Metadata tables, each with a name that indicates the domain that the vocabulary contained within represents.

METADATA		
PK	**c_fullname**	**VARCHAR(900)**
	c_hlevel	**INT**
	c_name	**VARCHAR(2000)**
	c_synonym_cd	**CHAR(1)**
	c_visualattributes	**CHAR(3)**
	c_tooltip	VARCHAR(900)
	c_total_num	INT
	c_basecode	VARCHAR(450)
	c_comment	CLOB
	c_metadataxml	CLOB
	c_facttablecolumn	**VARCHAR(50)**
	c_tablename	**VARCHAR(50)**
	c_columnname	**VARCHAR(50)**
	c_columndatatype	**VARCHAR(50)**
	c_operator	**VARCHAR(10)**
	c_dimcode	**VARCHAR(900)**
	update_date	DATE
	download_date	DATE
	import_date	DATE
	sourcesystem_cd	VARCHAR(50)
	valuetype_cd	VARCHAR(50)

4.2.3. Schemes table

The Schemes schema contains the unique prefixes obtained by different source codes. For example codes from the National Drug Code are prepended with the 'NDC' prefix, while codes from the United Medical Language System are prepended with the 'UMLS' prefix. This schema contains all the schemes recognized by the ONT system.

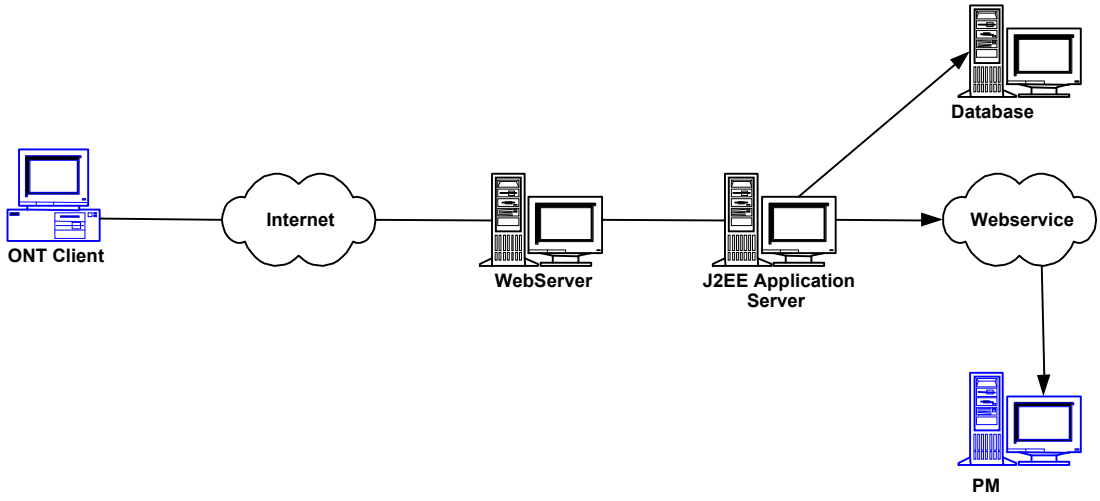
SCHEMES		
PK	**c_key**	**VARCHAR(50)**
	c_name c_description	**VARCHAR(50)** VARCHAR(100)

An example of a Schemes table is shown below:

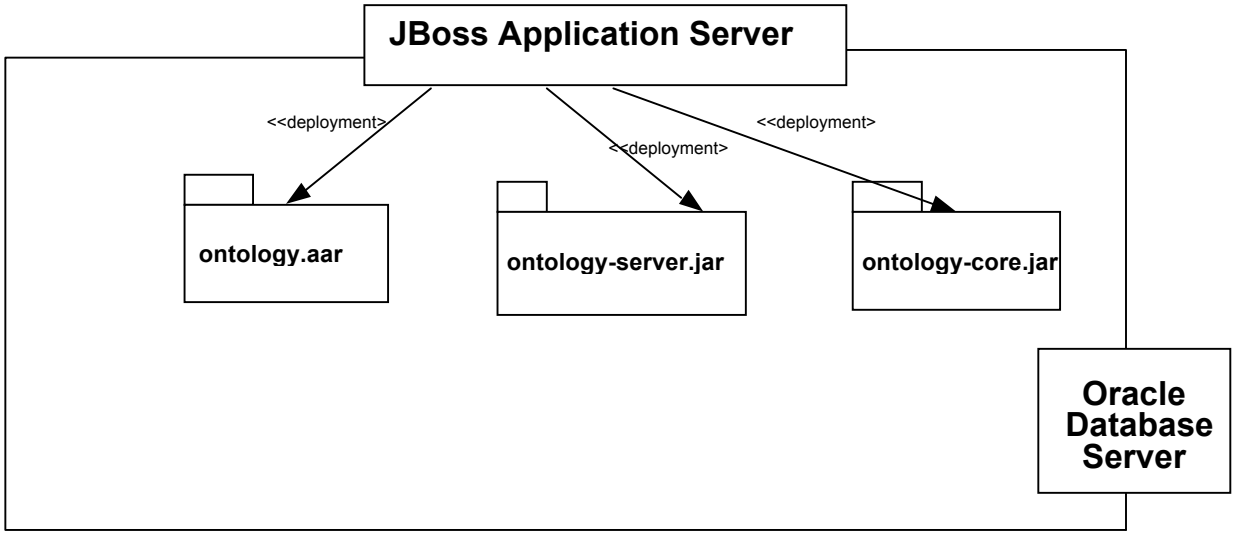
	C.KEY	C.NAME	C.DESCRIPTION
1	NDC:	NDC	National Drug Code
2	DSG-NLP:	DSG-NLP	Natural Language Processing Data
3	UMLS:	UMLS	United Medical Language System
4	LCS-I2B2:	LCS-I2B2	Local coding system for NLP results
5	ICD9:	ICD9	ICD9 code for diagnoses and procedures
6	LOINC:	LOINC	Lab codes

5. Deployment View

5.1 Global Overview



5.2 Detailed deployment model



References

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R. and Stafford, J., (2003). Documenting Software architectures – Views and Beyond. Addison Wesley, Boston, MA.

The "4+1" view model of software architecture, Philippe Kruchten, November 1995, http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf

Object Management Group UML 2.0 Specification - http://www.omg.org/technology/documents/formal/uml.htm

i2b2 (Informatics for Integrating Biology and the Bedside) https://www.i2b2.org/resrcs/hive.html