HOD^2MLC: Hybrid Ontology Design and Development Model with Lifecycle

Rishi Kanth Saripalle, School of Information Technology, Illinois State University, Normal, IL, USA

Steven A. Demurjian, Department of Computer Science and Engineering, University of Connecticut, Storrs, CT, USA

Michael Blechner, Department of Pathology, University of Connecticut Health Center, Farmington, CT, USA

Thomas Agresta, Department of Family Medicine, University of Connecticut Health Center, Farmington, CT, USA

ABSTRACT

Ontologies have gained increasing usage to augment an application with domain knowledge, particularly in healthcare, where they represent knowledge ranging from: bioinformatics data such as protein, gene, etc. to biomedical informatics such as diseases, diagnosis, symptoms, etc. However, the current ontology development efforts and process are data intensive and construction based, creating ontologies for specific applications/requirements, rather than designing an abstract ontological solution(s) that can be reusable across the domain using a well-defined design process. To address this deficiency, the work presented herein positions ontologies as software engineering artifact that allows them to be placed into the position to share the captured domain conceptualization and its vocabulary involving disparate domain backgrounds, that can then be created, imported, exported and re-used using different frameworks, tools and techniques. Towards this end, the authors propose an agile software process for ontologies referred to as the Hybrid Ontology Design & Development Model with Lifecycle, **HOD^2MLC**. *To place* **HOD^2MLC** *into a proper perspective, they explore, compare, and contrast it to existing ontology design and development alternatives with respect their various phases as related to the authors' work and phases in varied SDP models.*

Keywords: *HOD^2MLC, Hybrid Ontology Life Cycle, Model Driven Ontology Development, Ontology Meta-Modeling, Ontology Modeling*

DOI: 10.4018/IJITWE.2015040102

1. INTRODUCTION

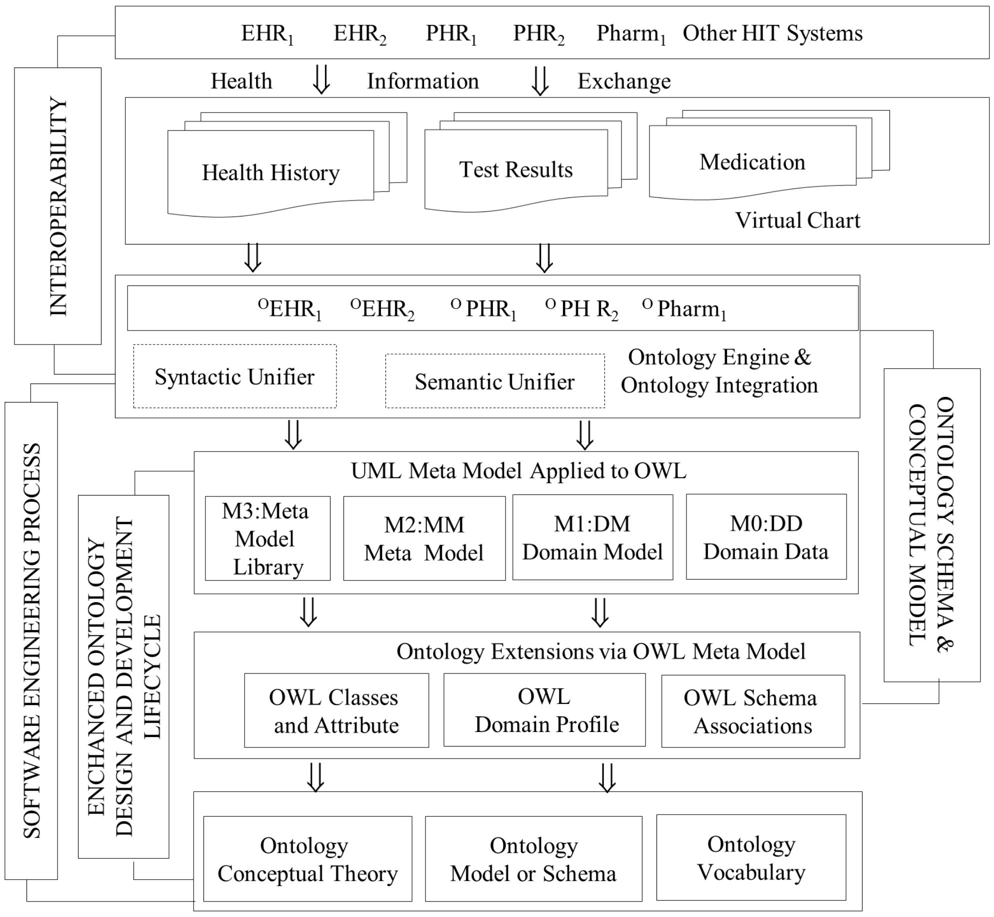
An ontology is a compilation of *concepts* that can be represented as a class composed of *attributes* coupled with *relationships* or *associations* that define interactions between the classes and augmented with *axioms* which are rules or constrains on classes, attributes, or relationships. Ontologies are promoted as a part of the Semantic Web to attach knowledge to information thereby aiding users (humans and agents) in various ways. Ontologies are heavily utilized in research such as knowledge engineering and representation, domain modeling (Saripalle, Demurjian, & Behre, 2011), database integration (Konstantinou, Spanos, & Mitrou, 2008), natural language processing (Kaihong, Hogan, & Crowley, 2011), etc. For our purposes, we have focused on clinical informatics that involves medical organizations (e.g., private practices, hospitals, etc.) that represent, store patient data (e.g., demographics, diagnosis, vitals, etc.) electronically using standards such as Continuity of Care Record (CCR)[1], Health Language Seven (HL7) Standard Clinical Document Architecture (CDA) (Boone, 2011), etc. and share the data in the form of Electronic Health Record (EHR) or Personal Health Record (PHR). As shown in the top portion of Figure 1, EHRs, PHRs, and other health IT systems are brought together by Health Information Exchange (HIE) (Demurjian, Saripalle, & Behre, 2009) (promoted by the HITECH Act[2]) into a conceptual setting called the Virtual Chart (VC) which provides a common global view for all potential users. For semantic knowledge, each of these systems employs standard medical ontologies such as: Logical Observation Identifiers Names and Codes (LOINC)[3] a standard for identifying medical laboratory observations; International Classification of Disease (ICD)[4] to hierarchically organize medical concepts such as diseases, symptoms, injuries, procedure, etc., Unified Medical Language System (UMLS) (Bodenreider, 2004) aggregation of medical ontologies and terminology such as ICD, LOINC, SNOMED, etc.; and have a semantic network and metathesaurus. The main objective of our work is to achieve interoperability among the health systems by unifying not only the data, but most importantly, unify their ontologies through their schema or model integration, as shown in the third horizontal box from the top of Figure 1.

As we seek to achieve this interoperability, we have identified a number of problems that significantly affect the process. First, individual EHRs or healthcare applications are built using different ontologies organizing domain knowledge in different ways to suit the specific business and medical requirements. This current approach is often an ad-hoc process, with minimal attention given to process and to a consideration of the impact of the ontology on other software components or ontologies. As a result, the developed ontologies are often incompatible and difficult to integrate. For example, UMLS Metathesaurus is the most comprehensive effort for integrating independent/disparate medical ontologies. The integration of a new ontology into UMLS is ad-hoc and exhaustive process that includes the following steps: automatic techniques - mostly a combination of natural language processing, corpus processing, linguistic analysis, etc., expert assessment – human intervention mostly for validating automatic techniques, and auditing protocols- for verifying the correctness of the resulting system. Second, the existing ontology building process is predominantly data intensive and construction based, often dictated by the talent and expertise of the designer rather than using any concrete well-defined process. For example, UMLS and FMA (Formal Model of Anatomy) comprise of millions of concepts and relationships, where the focus is on the acquiring the domain data rather than an ontological model that is reusable, modular and importantly, that describes the data. Third, the current ontology development processes and knowledge representational frameworks lack an ability to design solutions that are broader in scope i.e. they have a convoluted design and development processes that are clearly disjoint in a conventional software modeling approach (Kuhn, 2010; Guizzardi, 2010). Gonzalez-Perez and Henderson-Seller also state that for achieving interoperable ontological systems, the ontological domain must have an approach that executes ontology modeling and development process simultaneously (Gonzalez-Perez & Henderson-Seller, 2006). This further supports our statement on ontology design/modeling and its existing development methodologies.

To address these deficiencies, our prior work as shown in the bottom portion of Figure 1 (fourth horizontal block) has proposed an integrated ontology framework for HIE which includes a need for a software design process for ontologies that uses our three ontological modeling extensions (Saripalle, Demurjian, Algarín, & Blechner, 2013)(fifth horizontal block, Figure 1).

Our objective in this paper is to explain our preliminary ontology process (Saripalle & Demurjian, 2011), extend the process to utilize the previous research work on ontology modeling extensions, provide a detailed discussion of the importance of the software development process (SDP) (Docherty, 2005) and the vital role that different software process models can play in conceptualizing, sharing, modularizing, and reusing software artifacts across multiple setting, and using this as a basis to propose a *Hybrid Ontology Design & Development Model with Lifecycle* (HOD^{2}MLC). Historically, SDP methodologies have proven to be: efficient by *reduc-*

Figure 1. A framework for ontology design and development


EHR1 EHR2 PHR1 PHR2 Pharm1 Other HIT Systems
Health Information Exchange
Health History
Test Results
Medication
Virtual Chart
INTEROPERABILITY
OEHR1 OEHR2 O PHR1 O PH R2 O Pharm1
Syntactic Unifier
Semantic Unifier
Ontology Engine & Ontology Integration
ONTOLOGY SCHEMA & CONCEPTUAL MODEL
UML Meta Model Applied to OWL
M3:Meta Model Library
M2:MM Meta Model
M1:DM Domain Model
M0:DD Domain Data
SOFTWARE ENGINEERING PROCESS
ENCHANCED ONTOLOGY DESIGN AND DEVELOPMENT LIFECYCLE
Ontology Extensions via OWL Meta Model
OWL Classes and Attribute
OWL Domain Profile
OWL Schema Associations
Ontology Conceptual Theory
Ontology Model or Schema
Ontology Vocabulary

ing latency via defining workflows, tasks, roles, etc., to be executed at specific time intervals; promote *improved communication* between divergent teams; demonstrate a degree of *reliability* to be applied to other projects of similar time-scales and budget; and, hopefully result an *easily maintainable* system developed in a systematic allowing efficient debugging of problem and performance. HOD^{2}MLC strives to attain these same benefits when designing, developing, and deploying ontologies from software engineering perspective.

In this paper, we propose a *Hybrid Ontology Design & Development Model with Lifecycle* (HOD^{2}MLC) model by leveraging software development concepts and methodologies with the objective of narrowing the gap between ontology design and development and the SDP. To meet this objective, the remainder of the paper is organized into four sections. Section 2 introduces a complex clinical scenario on health care and the role of ontologies in clinical informatics. Section 3 provides a conceptual model of a patient medical record and background information on our previous work on ontological modeling extensions (Saripalle, Demurjian, Algarín, & Blechner, 2013) providing examples via the scenario of Section 2. Section 4, has a three-fold presentation: motivation on select SDP models that have impacted our work; an overview of the HOD^{2}MLC that describes its nine phases (*Problem Analysis*, *Integration, Knowledge Acquisition, Specification, Design, Analysis, Implementation, Testing,* and *Maintenance and Documentation*) and their interactions in general terms using a combination of SDP and other software modeling techniques; and, a detailed review of the nine phases of HOD^{2}MLC that demonstrates the process with examples that build on the clinical scenario in Section 2 and OWL (Web Ontology Language) extensions in Section 3. To place our work into its proper perspective, Section 5 introduces and reviews seven ontology development process alternatives, and compares and contrasts them against HOD^{2}MLC using a set of qualitative criteria. Finally, Section 6 concludes the paper and details ongoing and planned research.

2. A CLINICAL SCENARIO OF HEALTH CARE

This section presents a realistic scenario of patient care that includes: a patient, relevant medical problem, involved laboratory tests and results, resulting medical diagnosis, the role of involved HIT systems in the process, motivation to fully place our work in this paper in an appropriate context as shown in Figure 1. To begin, suppose that a 72 year old male, Mr. Smith, with a history of type 2 diabetes presents to the emergency room (ER) complaining of shortness of breath (SOB, also known as dyspnea) on exertion and is triaged by an ER physician and nurse. He reports experiencing increased difficulty climbing the one flight of stairs in his house. Mr. Smith also indicates experiencing occasional chest pressure on exertion (stable angina). He has recently developed swelling in his ankles and feet (edema). He indicates that he takes metformin for his diabetes and benazepril for his blood pressure. He also takes an aspirin a day because his regular doctor told him that he should. The physical exam of Mr. Smith reveals a gentleman in mild respiratory distress with moderate pedal (foot) and lower extremity edema (fluid in tissues). He has a regular pulse at 90 beats per minute and blood pressure of 140/90 (mildly elevated). Oxygen saturation[5] on room air is 88% and rises to 98% when given supplemental oxygen by nasal cannula at 2 liters/min. The diagnosis is that Mr. Smith is suffering from an exacerbation of congestive heart failure (CHF). Further, the occasional chest pressure on exertion suggests that he has coronary artery disease where inflammation and deposition of cholesterol in the vessel wall results in a localized expansion of the vessel wall that can impede blood flow.

In terms of HIT, there are many systems that are involved and must be consulted. Mr. Smith provided the ER physician with access to his PHR that had been recently initiated by the patient's son who was not present. Through HIE, the PHR can be utilized by the provider to find out information on supplements or other medications that the patient has neglected to tell him about that are accessible via the PHR to pharmacies and drug stores. A search of the regional HIE revealed that the patient had a recent admission at another hospital for CHF; thus data must be gathered from that EHR, his primary physicians EHR, and potentially others in support of his ongoing care. The discharge summary from that admission indicated that the patient had improved after 2 doses of a diuretic that increases urination and the loss of fluid decreases your blood volume which often improves CHF symptoms and the patient had been discharged. A query of the electronic prescription database determines that the patient never filled the Lasix prescription, and this is confirmed by reviewing the mediations in the PHR loaded from Mr. Smith's pharmacy. The HIE also informed the ER physician that the patient has a documented allergy to sulfa containing medications; the PHR may have other information on allergies or medication reactions.

The scenario as presented in this section has the underlying aspect of information that has been gathered for this patient from multiple data sources that have been utilized for his treatment. From a research perspective, an almost limitless number of questions can arise when considering Mr. Smith's case or similar patients. A number of broad research topics (RT) can be posed that:

- **RT1:** Effects of a specific medical therapy on a patient's co-morbid conditions:
    - How does metformin used for glucose control in type 2 diabetics effect the incidence and natural history of CHF and Chronic Renal (kidney) Failure or stable Angina (chest pain successfully treated)?
    - If there is an effect on any of these conditions, is type 2 diabetics dose dependent on metformin or does the absence of any other medications alter it?
    - Symptoms and Diseases with high BNP (800 pg/mL)?
- **RT2:** Comparative study of different diabetic therapies with CHF using various patient groups:
    - What is the patient's profile with CHF and associated medications involved for diabetic therapies?
    - What are the incidence and/or severity of CHF for diabetes patients who use the class of hyperglecimic agents?
    - Is metformin more or less effective in maintaining glucose control in type 2 diabetics with a history of stable angina as compared to other anti-hyperglycemic medications?

Individually, each of these research topics provides a means to allow a physician or clinical researcher to explore various aspects of a disease, its symptoms, medications, therapies, interactions with other conditions, etc.

3. BACKGROUND ON ONTOLOGIES AND OWL EXTENSIONS

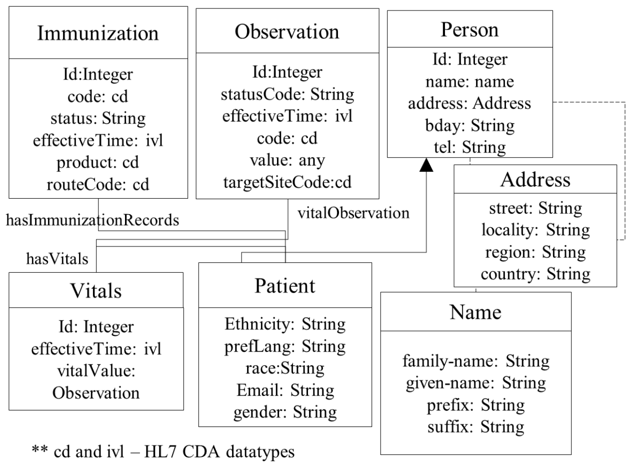
This section provides background information on our previous work on Web Ontology Language (OWL) (Allemang & Hendler, 2011) extensions for Attribute, Domain Profile, and Schema Associations (Saripalle, Demurjian, Algarín, & Blechner, 2013), and our use of the UML meta-model as applied to OWL and the associated extensions (see bottom of Figure 1). By presenting our prior ontology modeling work coupled with an ontology design of the scenario of Section 2, we provide a set of ontology design components as realized in the figures in this section. As a result, these ontology design components are utilized in Section 4 to illustrate the process the Hybrid Ontology Design & Development Model with Lifecycle (HOD^{2}MLC)

To begin, the sample EHR UML class diagram that we will use for examples throughout the paper is shown in the Figure 2a, and has; Person and Patient classes with their respective attributes; the Immunization class for the patient's immunization records; the Vitals class captures all of the patient body vital signs such as blood pressure, heart rate, respirations, weight, height, BMI etc.; and the Observation class captures any other patient medical information such as allergies, interventions, lab results (captured as a group of Observations), adverse reactions, etc. Our first extension to OWL is *Attribute*, which augments OWL with the capability to capture characteristics owned by a class and define an OWL Class as an aggregation of attributes and datatypes. We introduced a first class element OWL Attribute (owl:Attribute) and redefined the semantics of the existing OWL Class to handle attributes. For example, Vitals and Patient are connected using association hasVitals and Vitals has attributes effectiveTime of type IVL and vitalValue of type Observation. Thus, classes Patient, Vitals, Observation, etc., from Figure 2a are of type owl:Class, while the attributes attached to the class (e.g., hasName, vitalObservation, hasEffectiveTime, has Address, etc.) are of type owl:Attribute, and the associations between classes (e.g., hasVitals, hasImmunizationRecords, etc.) are of type owl:ObjectProperty. Figure 2b renders the OWL attribute representation.

The next set of ontology design components serve as an abstract representation of ontologies at a metamodel level and facilitates the ability to define sample ontology domain models as ontology design components for the scenario of Section 2. Figure 3a (left, Figure 3) is the Diagnosis Model which captures the required knowledge for identifying a patient's conditions, its causes and the approaches to mitigate these conditions by defining classes such as Metabolic System Diseases, Digestive System Diseases, Cardiac System Diseases, Respiratory System Procedure, Digestive System Medications, Fractures, Dislocations, etc., that capture the respective medical vocabulary/data and are connected via associations such as has GeneralSymptoms, hasCardiacSymptoms, hasCardiacProcedure, causedByGeneralInjury, etc.

Similarly, Figure 3b (right, Figure 3) details the Test Model to capture knowledge on various medical tests for analyzing patient conditions, and Figure 3c (bottom, Figure 3) has the Anatomy Model to capture knowledge on physical characteristics of the human body. These design models

Figure 2. EHR model and OWL attribute representation

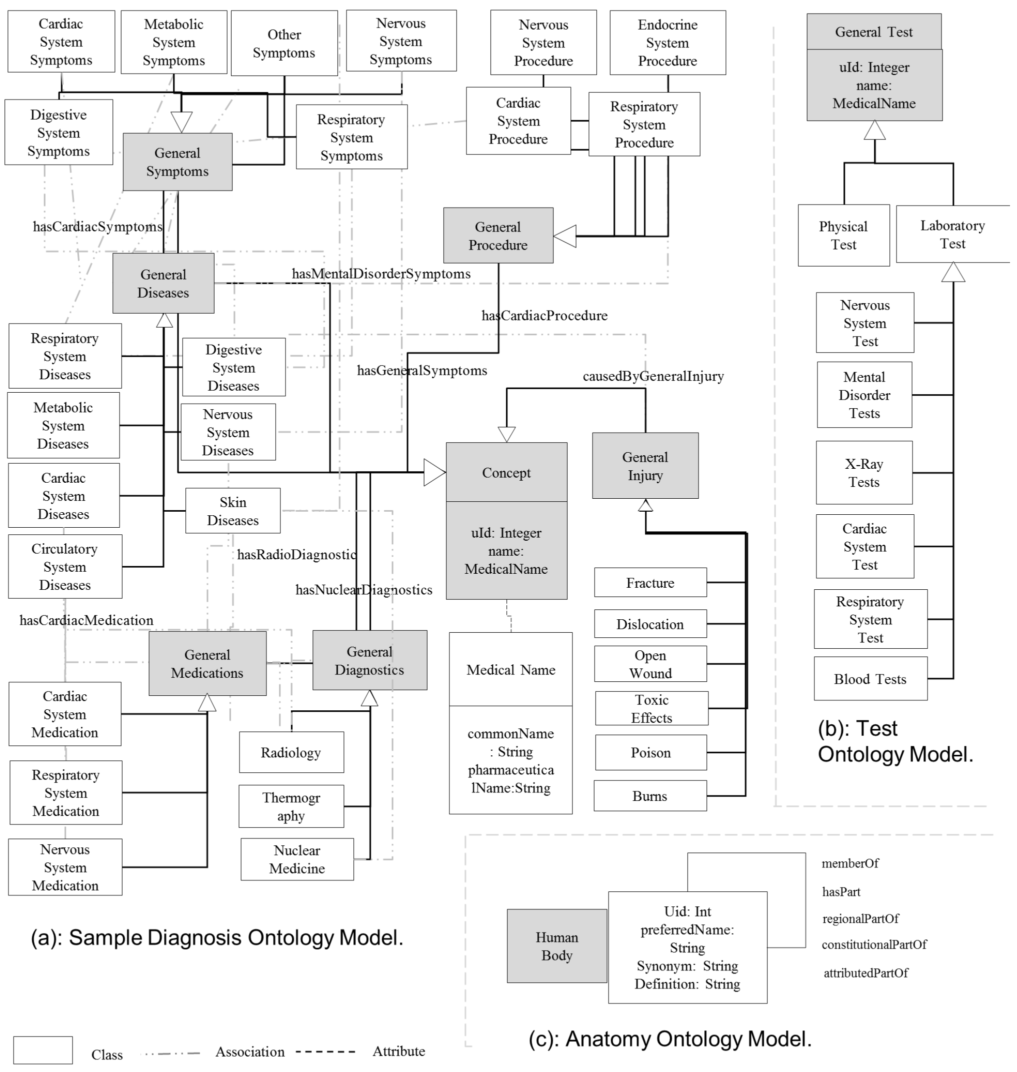


(a) *Conceptual EHR Model in UML via HL7 CDA*

```
<owl:Class rdf:Id="Vitals"/>
<owl:DatatypeProperty rdf:Id="Id"/>
<owl:Attribute rdf:Id="hasEffectiveTime"/>
< owl:Attribute rdf:Id="hasVitals"/>
<owl:Class rdf:Id="Observation"/>
<owl:DatatypeProperty rdf:Id="id"/>
<owl:DatatypeProperty rdf:Id="StatusCode"/>
<owl:Attribute rdf:Id="effectiveTime"/>
<owl:Attribute rdf:Id="code"/>
<owl:Attribute rdf:Id="value"/>
<owl:Attribute rdf:Id="targetSite"/>
<owl:Attribute rdf:Id="vitalObservation"/>
          <owl:Domain rdf:Id="Vitals"/>
           <owl:Range rdf:Id="Observation"/>
<owl:Attribute/>
<owl:Attribute rdf:Id="hasEffectiveTime"/>
            <owl:Domain rdf:Id="Vitals"/>
            <owl:Range rdf:Id="IVL"/>
<owl:Attribute/>
```

(b) *An OWL Attribute*

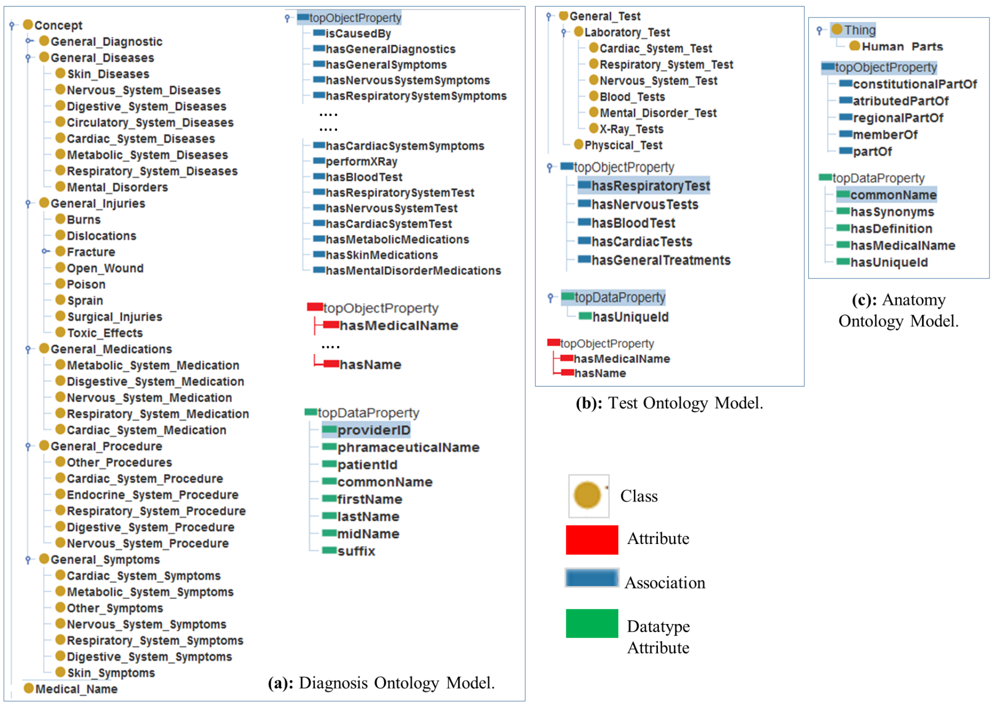
Figure 3. Sample diagnosis, test and anatomy ontology models represented in UML



are realized in OWL using the Protégé editor[6] in Figure 4a with the Diagnosis Model (Figure 3a), Figure 4b with the Test Model (Figure 3b), and Figure 4c with the Anatomy Model (Figure 3c).

The next set of ontology design components leverage Figures 3 and 4 to abstract and define *generic domain specific type concepts,* shown in Figure 5a that include: *Disease* (classes Metabolic System Diseases, Respiratory System Diseases, Digestive System Diseases, etc., which are all of *type* Disease); *Symptom* (classes General Symptoms, Cardiac System Symptoms, Respiratory System Symptoms, etc., which are all *of type* Symptom); *Medication* (classes Cardiac System Medication, Respiratory System Medication, Nervous System Medication, etc., which are all of *type* Medication)*;* etc. Within these abstract types, attribute abstract concept types can be defined, namely: *hasName* (pharmaceuticalName, commonName, family-name, given-name,

Figure 4. OWL implementation of the domain models from Figure 3



```
Concept
  General_Diagnostic
  General_Diseases
    Skin_Diseases
    Nervous_System_Diseases
    Digestive_System_Diseases
    Circulatory_System_Diseases
    Cardiac_System_Diseases
    Metabolic_System_Diseases
    Respiratory_System_Diseases
    Mental_Disorders
  General_Injuries
    Burns
    Dislocations
    Fracture
    Open_Wound
    Poison
    Sprain
    Surgical_Injuries
    Toxic_Effects
  General_Medications
    Metabolic_System_Medication
    Disgestive_System_Medication
    Nervous_System_Medication
    Respiratory_System_Medication
    Cardiac_System_Medication
  General_Procedure
    Other_Procedures
    Cardiac_System_Procedure
    Endocrine_System_Procedure
    Respiratory_System_Procedure
    Digestive_System_Procedure
    Nervous_System_Procedure
  General_Symptoms
    Cardiac_System_Symptoms
    Metabolic_System_Symptoms
    Other_Symptoms
    Nervous_System_Symptoms
    Respiratory_System_Symptoms
    Digestive_System_Symptoms
    Skin_Symptoms
Medical_Name

topObjectProperty
  isCausedBy
  hasGeneralDiagnostics
  hasGeneralSymptoms
  hasNervousSystemSymptoms
  hasRespiratorySystemSymptoms
  ....
  ....
  hasCardiacSystemSymptoms
  performXRay
  hasBloodTest
  hasRespiratorySystemTest
  hasNervousSystemTest
  hasCardiacSystemTest
  hasMetabolicMedications
  hasSkinMedications
  hasMentalDisorderMedications

topObjectProperty
  hasMedicalName
  ....
  hasName

topDataProperty
  providerID
  phrameceuticalName
  patientId
  commonName
  firstName
  lastName
  midName
  suffix
```

(a): Diagnosis Ontology Model.

```
General_Test
  Laboratory_Test
    Cardiac_System_Test
    Respiratory_System_Test
    Nervous_System_Test
    Blood_Tests
    Mental_Disorder_Test
    X-Ray_Tests
  Physical_Test
topObjectProperty
  hasRespiratoryTest
  hasNervousTests
  hasBloodTest
  hasCardiacTests
  hasGeneralTreatments
topDataProperty
  hasUniqueId
topObjectProperty
  hasMedicalName
  hasName
```

(b): Test Ontology Model.

```
Thing
  Human Parts
topObjectProperty
  constitutionalPartOf
  atributedPartOf
  regionalPartOf
  memberOf
  partOf
topDataProperty
  commonName
  hasSynonyms
  hasDefinition
  hasMedicalName
  hasUniqueId
```

(c): Anatomy Ontology Model.

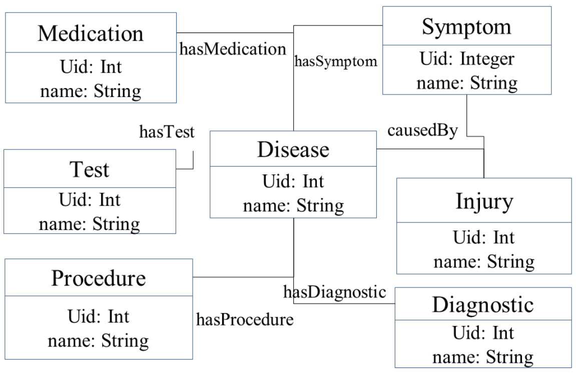
etc., which are all of *type* hasName); and *hasId* (hasUId, hasDeaNumber, etc., which are all of type hasId), etc.

The *OWL Domain Profile* (Saripalle, Demurjian, Algarín, & Blechner, 2013) enables the user to capture these abstract types as *profile concepts* as shown in Figure 5b and *impose* them onto the ontology model concepts (Level M1, Figure 1) as shown in Figure 6. The OWL Domain Profile supports the ontology OWL framework at the metamodel level by imposing the profile concepts onto the OWL domain ontology as shown in the Figure 6a.

In order to impose the profile types onto the domain model entities we have designed and implemented an algorithm *Domain Profile parser* that authenticates and validates the imposing of the profile entities onto the ontology model concepts. First, the defined sample profile (Figure 5b) is loaded into the ontology editor where there the algorithm *validates* the profile for its structural and semantic correctness against the ODP framework and *parses* the profile to identify the defined profile abstract types. Then, the domain expert selects the types and impose them onto the domain model entities, i.e., the Cardiac System Diseases (Figure 3a) is *oftype Disease* (Profile type, Figure 5b) as shown in the Figure 6c.

The final extension, *OWL Schema Associations,* captures associations across ontology models by interlinking ontology descriptive concepts defined in Ontology Meta Vocabulary (OMV) (Hartmann, Palma, & Sure, 2005). OMV model for providing metadata about the ontology. OMV provides a way to capture the metadata for an ontology including *domain, organization, language, place, version, tools,* etc. For example, this extension can be used to associate the three models: *Diagnosis, Tests* and *Anatomy* to one another. The Diagnosis Model (O_1) (Figure 4a) has OMV concept *ontologyDomain* with value "*Diagnosis*", the Anatomy Ontology model O_2 has *ontologyDomain* with value "*Anatomy*", and, the Test Ontology model O_3 has the concept

Figure 5. Sample abstract theory and its implementation using OWL domain profile

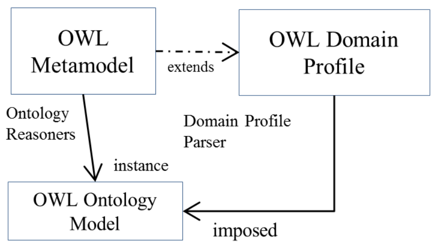


(a): Sample *Abstract Theory* for the ontology model in Figure .

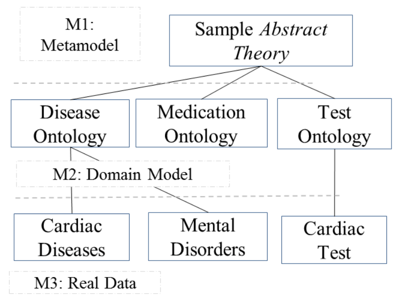
```
<?xml version='1.0' encoding='UTF-8'?>
<odp:ProfileClass rdf:ID='Disease'/>
<odp:ProfileClass rdf:ID='Symptom'/>
<odp:ProfileClass rdf:ID='Procedure'/>
<odp:ProfileClass rdf:ID='Treatment'/>
<odp:ProfileClass rdf:ID='Diagnostic'/>
<odp:ProfileClass rdf:ID='Injury'/>
<odp:ProfileClass rdf:ID='Medication'/>
<odp:ProfileObjectProperty rdf:ID='hasSymptom'/>
<odp:ProfileObjectProperty rdf:ID='hasMedication'/>
<odp:ProfileObjectProperty rdf:ID='hasTreatment'/>
<odp:ProfileObjectProperty rdf:ID='hasProcedure'/>
<odp:ProfileObjectProperty rdf:ID='hasDiagnostic'/>
<odp:ProfileObjectProperty rdf:ID='causedBy'/>
<odp:ProfileDatatypeProperty rdf:ID='hasName'/>
<odp:ProfileDatatypeProperty rdf:ID='hasUID'/>
</xml>
```

(b): Implementation of the Abstract theory (a) using ODP.

Figure 6. ODP domain profiles and the parser



(a): ODP Framework support for ontology models



(b): Imposing *Profile Concepts* onto ontology models

```
<?xml version='1.0' encoding='UTF-8'?>
<odp:Disease odp:isofType ='General_Diseases'/>
<odp:Disease odp:isofType ='Cardiac_System_Diseases'/>
<odp:Symptom odp:isofType ='General_Symptoms'/>
<odp:hasSymptom odp:isofType='hasGeneralSymptoms'/>
<odp:Procedure odp:isofType='General_Procedures'/>
<odp:hasProcedure odp:isofType='hasGeneralProcedure'/>
<odp:Medication odp:isofType='General_Medication'/>
<odp:hasMedication odp:isofType='hasGeneralMedication'/>
< odp:hasMedication odp:isofType='hasSkinMedications'/>
<odp:Injury odp:isofType='General_Injuries'/>
< odp:Injury odp:isofType='Fracture'/>
< odp:CausedBy odp:isofType='causedByBurns'/>
< odp:CausedBy odp:isofType='causedByFractures'/>
<odp:Test odp:isofType='General_test'/>
< odp:Test odp:isofType='Physical_Test'/>
<odp:hasTest odp:isofType='hasGeneralTest'/>
< odp:hasTest odp:isofType='hasPhyscialTest'/>
<odp:Diagnostic odp:isofType='General_Diagnostics'/>
< odp:Diagnostic odp:isofType='Radiology'/>
<odp:hasDiagnostic odp:isofType='hasTheormoDiagnostic'/>
< odp:hasDiagnostic odp:isofType='hasGeneralDiagnostic'/>
............
</xml>
```

(c): Imposing Abstract Theory(Figure 5) onto model entities

ontologyDomain with value "*Test*". These OMV concepts across these individual ontology models (O_1, O_2, and O_3) are associated to form *OWL Schema Associations* as shown in the Figure 7 whose implementation is shown in Figure 8. Figure 8a renders a subset of OMV concepts in OWL, Figure 8b shows *Diagnosis_Ontology*", "*Tests_Ontology*" and "*Anatomy_Ontology*" as instances of OMV concept "Ontology", and Figure 8c shows various properties of "*Diagnosis_Ontology*" including the schema associations *hasTests* and *effects* with Test and Anatomy ontologies.

Figure 7. Illustrating ontology schema relationships between ontology models

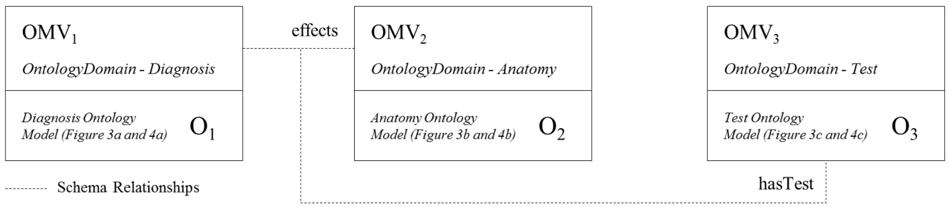
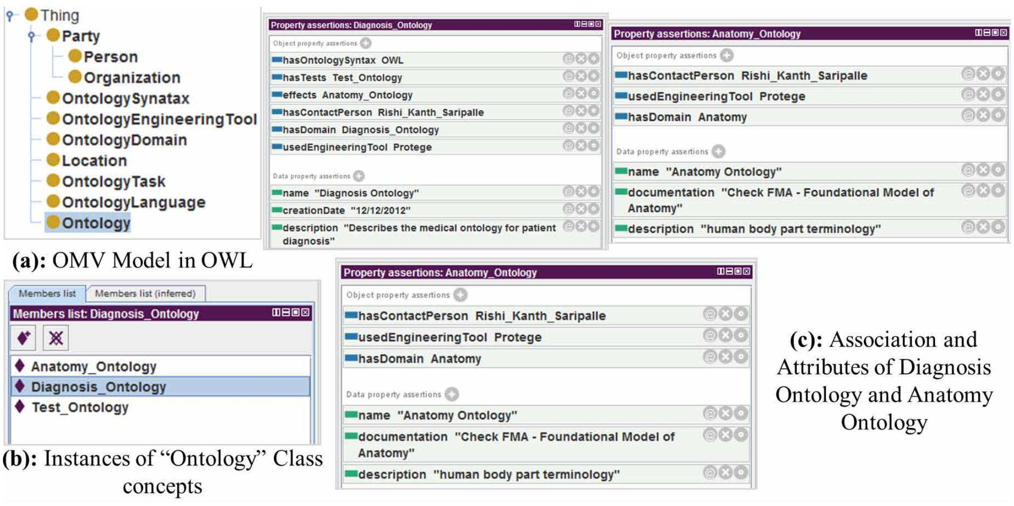

OMV_1
OntologyDomain - Diagnosis
Diagnosis Ontology Model (Figure 3a and 4a)
O_1
effects
OMV_2
OntologyDomain - Anatomy
Anatomy Ontology Model (Figure 3b and 4b)
O_2
OMV_3
OntologyDomain - Test
Test Ontology Model (Figure 3c and 4c)
O_3
Schema Relationships
hasTest

Figure 8. Ontology schema relationships implementation



Property assertions: Diagnosis_Ontology

Object property assertions
hasOntologySyntax OWL
hasTests Test_Ontology
effects Anatomy_Ontology
hasContactPerson Rishi_Kanth_Saripalle
hasDomain Diagnosis_Ontology
usedEngineeringTool Protege

Data property assertions
name "Diagnosis Ontology"
creationDate "12/12/2012"
description "Describes the medical ontology for patient diagnosis"

Property assertions: Anatomy_Ontology

Object property assertions
hasContactPerson Rishi_Kanth_Saripalle
usedEngineeringTool Protege
hasDomain Anatomy

Data property assertions
name "Anatomy Ontology"
documentation "Check FMA - Foundational Model of Anatomy"
description "human body part terminology"

(a): OMV Model in OWL

Members list: Diagnosis_Ontology
Anatomy_Ontology
Diagnosis_Ontology
Test_Ontology

(b): Instances of "Ontology" Class concepts

Property assertions: Anatomy_Ontology

Object property assertions
hasContactPerson Rishi_Kanth_Saripalle
usedEngineeringTool Protege
hasDomain Anatomy

Data property assertions
name "Anatomy Ontology"
documentation "Check FMA - Foundational Model of Anatomy"
description "human body part terminology"

(c): Association and Attributes of Diagnosis Ontology and Anatomy Ontology

4. HYBRID ONTOLOGY DESIGN AND DEVELOPMENT MODEL WITH LIFECYCLE (HOD^{2}MLC)

In this section, we define the various phases involved in designing and building ontologies, and then organize these phases to propose a **H**ybrid **O**ntology **D**esign & **D**evelopment **M**odel with **L**ife**c**ycle (**HOD^{2}MLC**) model. To begin, in Section 4.1, we expand our motivation from the introduction of the usefulness and utility of the software development process (SDP) for ontology design and development, by examining the process in detail; this sets the context for both the rest of this section, and the comparison of other ontology processes in Section 5. Using this as a basis, Section 4.2 presents a high level view of HOD^{2}MLC which contains nine phases (*Problem Analysis*, *Integration, Knowledge Acquisition, Specification, Design, Analysis, Implementation, Testing,* and *Maintenance and Documentation*) and the interactions among them, influenced by the agile methodology and feature-driven development. Then, in Section 4.3, we examine each of these phases in detail, explore their interactions with one another, and provide a comprehensive example based on the OWL extensions in Section 3 and using Figures 2-7 supplemented with additional material.

4.1. Motivation of the Software Development Process

A *Software Development Process* (SDP) is a systematic procedure of developing a software application that provides a set of phases that all stakeholders (end users, domain specialists, designers, developers, maintainers, etc.) participate in at different times for different purposes towards the conceptualization, specification, design, implementation, and deployment of a working system. There are many different SDP methodologies that have emerged over the history of computing. One of the first approaches, the *waterfall model* (Docherty, 2005) involve the classical phases (requirements, design, implementation, analysis, maintenance, etc.), with the requirement that each phase being completed before moving to the next phase. To address this deficiency, the *iterative model* (Docherty, 2005) that introduces loops and cycles among waterfall phases allowing stakeholders to analyze and revise their solutions. Taking a leap forward from these models was the *spiral model*, a cyclical process of four stages to: identify objectives and design alternatives, evaluate alternative and identify/deal with potential risks, develop and verify the next level product, and review results and plan for the next iteration. More recent emphasis on SDPs has been with the Unified Process Model (Kruchten, 2003), a conglomeration of iterative and incremental approaches that employs use case design to focus end user participation, creating multiple architecture and system views. Today, the *agile development lifecycle* (Craig, 2003), also available as a unified process model, has become a dominant approach to SDP, blending the best aspects of many different process models and expanding the scope and role of stakeholders of all types to participate in a process that truly results in the desired application.

One of the characteristics that all of the aforementioned SDP shares are a set of phases organized in different ways, but having similar purposes and intent across the varied process models. The *requirements phase* is intended to capture the main features and capabilities of the intended application in a manner suitable for all stakeholders; for a clinical application, bringing in appropriate medical professionals of all types relevant staff, and for our purposes, ontology designers, will be vital to achieve this phase. The *specifications phase* provides a clear, unambiguous description of the way the components of the software should be inter-connected, utilized and their responses to inputs ranging from normal working environment to exceptional situation; the specification may also include formal models that are later used to measure and assess the system. The *design phase* proposes alternative solutions in terms of conceptual models or domain schemas to solve the problem, and provides the ability to assess and evaluate them. Many different techniques can be employed at this phase including software architectures, software design patterns, UML design, entity-relationship and database design, ontology design, etc.; again, in a clinical application, stakeholders will be necessary to provide vital domain expertise. The *implementation phase* at early stages of the process can involve rapid prototyping of user interfaces for stakeholder input transitioning to implementation of design patterns, components, subsystems, and eventually the entire system. Throughout this process, the *testing phase* allows the software to be tested against the system requirements to see if it fits the original goals, again ranging for stakeholder input on user interfaces, component testing, acceptance testing, etc. *The deployment phase* may involve testing on staging servers with fake and then real data, stress testing to gauge impact on performance for expected number of users, testing with different hardware and software configurations, and developing appropriate manuals and training materials; in a clinical domain this will require input from stakeholders with the requisite medical expertise. Finally, the *maintenance phase* helps the team to keep track of the systems performance, updates, hardware maintenance, etc. While not an exhaustive list, it represents the phases that are employed by many SDPs; what differs is how and when they are utilized.

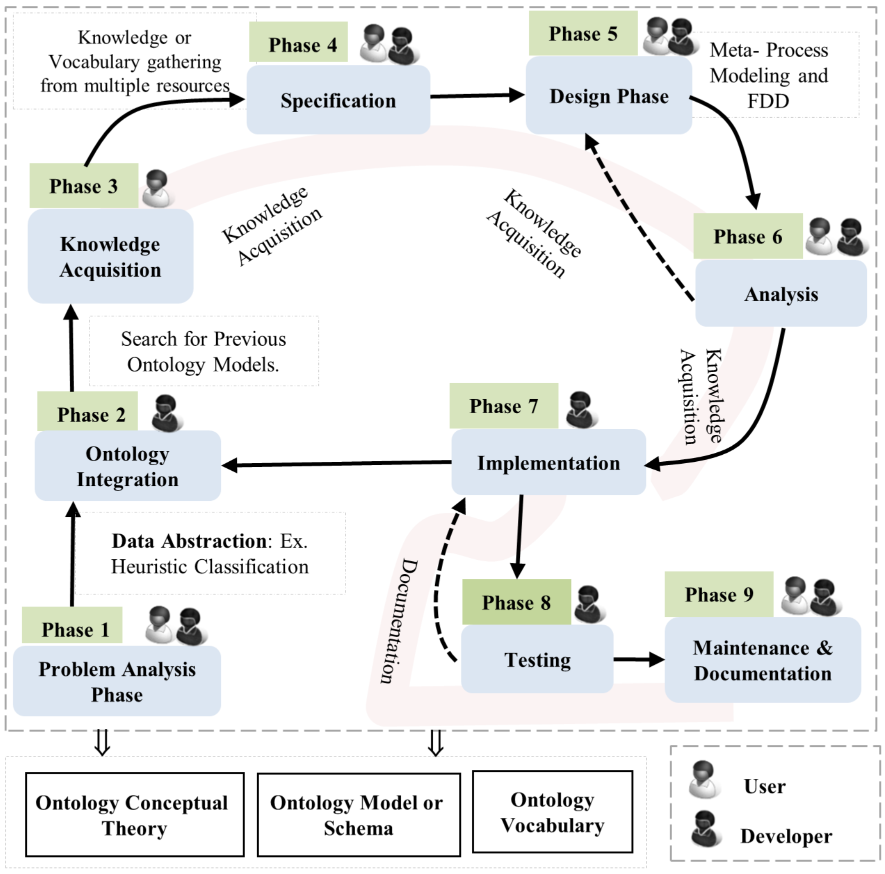
4.2. The HOD^{2}MLC Model

The *Hybrid Ontology Design & Development Model with Lifecycle* (HOD^{2}MLC) model is an *agile* model comprised of a collection of sequential and parallel phases stakeholders have to execute as shown in the Figure 9. This process results in the definition and creating of abstract type concepts, an ontology model, and an associated ontology vocabulary/data as seen in the bottom box of Figure 1, and replicated in Figure 9; this represents the output of the HOD^{2}MLC. The agile lifecycle model is adopted as it provides adaptive planning, incremental development, a timed iterative approach, and encourages rapid and flexible response to dynamic project changes. HOD^{2}MLC has a set of phases (*Problem Analysis*, *Integration, Knowledge Acquisition, Specification, Design, Analysis, Implementation, Testing,* and *Maintenance and Documentation*) to be explored in Section 4.3, with each phase assigned a well-defined role or responsibility which is utilized in order to identify the best-fit software design model or approach for it role in the entire life cycle. The HOD^{2}MLC commences with *Problem Analysis Phase* (Phase 1) which analyzes the issues that are need to support ontology development and applies *heuristic abstraction techniques* (Clancey, 1985) for assisting in problem definition, its analysis, and foundations for other phases in the cycle. Heuristic abstraction is a framework equipped with multiple methods (data abstraction, definition abstraction, and qualitative abstraction) which can be applied to a range of scenarios primarily faced in Phase 1.

HOD^{2}MLC leverages aspects of the *iterative* model throughout Phases 2 to 8 to assist developers to take advantage of what was learned during a previous cycle to make changes in an *incremental* fashion which provides periodical partial output providing development guidance. The *Ontology Integration Phase* searches for existing ontologies that may be used at a starting point for the intended application. The *Knowledge Acquisition Phase* tries to gather knowledge concepts at various levels (M2, M1& M0- Figure 1). The *Specification Phase* defines concrete boundaries to refine and scope the identified problem. The *Design Phase* is the core where the domain model is developed independent of its target language platform and employs the *Feature Driven Development (FDD)* (Palmer & Felsing, 2002) technique for identifying various concepts by providing *iterative* and *incremental top-bottom* approach allowing developers to identify concepts at various levels hierarchically. The *Analysis Phase* allows developers (and domain users) to validate the domain model from the design phase with the specification; this is indicated by the backward pointing dashed arrow between these two phases in Figure 9 and may necessitate a revising of specification, design, and analyses phases. The *Implementation Phase* realizes the domain model from Phase 5 using a chosen knowledge framework such as RDF/RDFS, OWL, etc., and in the medical domain would be used to provide vocabulary support, knowledge acquisition, decision support, etc. The *Testing Phase* is intended to explore the developed knowledge ontology in detail, at both design and instance level; at this stage, stakeholders (medical domain users) would play a pivotal role in evaluating the ontology and its completeness, resulting in the potential for a feedback look back to the implementation phase. Lastly, the *Maintenance & Documentation* maintains the deployed ontology and documents the whole process for future usage; this is an active collaboration between all stakeholders.

Note that the *Knowledge Acquisition Phase* can be executed in parallel with other phases until the *Implementation Phase*, since the *Implementation Phase* is responsible for implementing the domain model and its vocabulary gathered along the *knowledge acquisition phase*. The *Documentation Phase* can also be executed in parallel starting from the *Analysis Phase* and ending in the *Maintenance* and *Documentation Phase*. The life cycle model also depicts two broad types of stakeholders *User* and *Developer* involved in various phases; in a clinical application *user* could be various medical providers, their staffs, other medical personnel, medical

Figure 9. HOD²MLC: Hybrid ontology design and development model life cycle



ontologists etc., while the *developer* may play different roles such as knowledge, specification, design, etc. To complete the entire process, the output of the HOD²MLC process is the creation of an ontology that consists of an ontology abstract conceptual theory, the ontology model (the types and structure), and the ontology vocabulary (terms); all of these can be instantiated to form a specific ontology for a particular application., e.g., one for an EHR, one for PHR, one for a Pharmacy System, etc.

4.3. Phases of HOD²MLC

In this section, we will discuss the phases involved in HOD²MLC model as shown in Figure 9. Our intent is to detail each phase starting from (Phase 1: *Problem Analysis Phase*) through its completion (Phase 9: *Maintenance* and *Documentation Phase*), the importance of the phase, interaction with other phases, the usage of the phase and illustrate by using the scenario introduced in Section 2 and Figures 2-8, complemented with additional material to fully demonstrate the methodology's usage in the ontology design and development process. The remainder of this section has subsections for each respective phase.

4.3.1. Phase 1: Problem Analysis

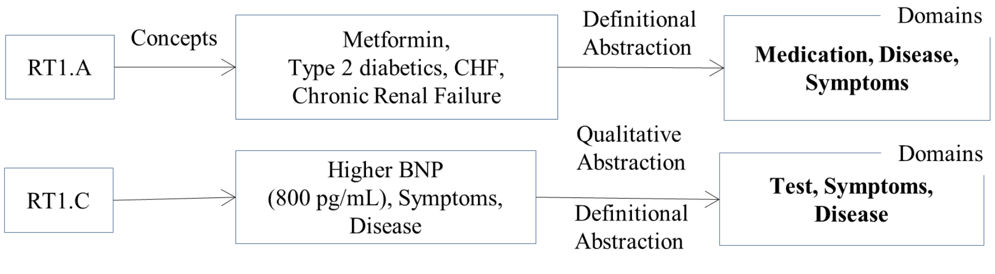
Recall from Section 2, that the objectives was to allow data from multiple sources via HIE to be available to clinical researchers who want to query the combined clinical repository to answer based on multiple clinical variables such as diseases, symptoms, medications, diagnoses, etc. The *Problem Analysis* phase is the commencing point of the proposed HOD^{2}MLC model in order to identify and analyze the overall *problem domain* (e.g., queries against combined clinical repositories) faced by the desired information system which will necessitate the development of new ontology(s) or extending existing ontology(s) or both. Our objective in this phase is to develop representative queries for data mining in support of clinical research using the research topics as illustrated in Section 2 or via UML use-case diagrams that represent samples that are specific to an instance data of the domain or a state of the ontology. Essentially, the research topics in their broad definitions provide the means for us to elicit relevant information from intended users in order to analyze them to formulate a *domain problem,* i.e., the domain level problem which encompasses multiple component problems. For example, the two RTs:

RT1.A: How does metformin used for glucose control in type 2 diabetics effect the incidence and natural history of CHF and Chronic Renal Failure or stable Angina?
RT1.C: Symptoms and Diseases with high BNP (800 pg/mL)?

The aforementioned questions are targeting a specific instance/data of the ontology knowledge that can hopefully be generalized to be used in order to formulate a boarder *domain problem* along with the respective model concepts and the vocabulary to identify the domain involved and then association (schema associations) to provide higher level overview of the problem. *Abstraction Techniques* proposed by Clancey (1985) which is further classified into: Definitional Abstraction - based on defined, essential, and necessary, characteristics features of the concept; Qualitative Abstraction - form of definition involving quantitative data; and Generalization – based on concept hierarchy, can be employed for abstracting concepts from RT's and identify their domains to formulate a *domain problem*. As shown in the Figure 10, consider that the concepts identified for RT1.A are: Metformin, Type 2 Diabetes, CHF, and Chronic Renal Failure. Based on these concepts, we can first surmise that Metformin (defined as - a preferred oral anti-diabetic drug of choice for the treatment of type 2 diabetes) is a drug involving the Medication domain.

Similarly, the concepts Type 2 diabetes, and Chronic Renal Failure involve the domains of Disease and Symptom, which are two more of the needed domains along with Medication. Likewise, in analyzing RT1.C that has a required BNP evaluation, this indicates that Test needs to be added as a potential ontology domain. While applying these techniques, the designer may

Figure 10. Applying abstraction techniques on RT's for obtaining the domains



also identify various concepts (e.g., Disease, Symptom, Test, Metformin, CHF, diabetes, etc.) useful for other phases (Phase 3 in Section 4.3.3 and Phase 5 in Section 4.3.5). Thus, the problem analysis phase assists the designers to identify the domains and concepts involved for building the medical knowledge system, allowing teams to gather and interact in order to discern the appropriate ontology concepts. If one extends the above discussion to include all RTs, then the domains required for assisting researchers to solve the problem are: *Symptom*, *Disease*, *Medication*, *Test, Diagnostics, Anatomy,* and *Procedure*.

4.3.2. Phase 2: Integration Phase

The *Integration Phase* allows designers to search for existing ontology model(s) meeting the domain(s) criteria identified in *Problem Analysis Phase* (Phase 1) and to also understand the associations that may exist among the various ontologies. For the latter case, before building an ontology model for the domain of *Medication*, the designer(s) might want consider the reuse of the *RxNorm* (Liu, Ma, Moore, Ganesan, & Nelson, 2005) model standard that provides normalized names for clinical drugs. Similarly, for the domains of Symptom, Disease, and Procedure the ontology designers can refer to standard medical systems such as: ICD, UMLS, LOINC, etc. The designers can use the RT questions and their interdependencies in order to establish associations, which are shown in Figure 8. For instance, for RT1.A the designer and the medical user have to understand the association between the domains Test, Disease, and Symptom, as well as their types and domain modeling elements (Section 3.3).

4.3.3. Phase 3: Knowledge Acquisition

The *Knowledge Acquisition Phase* primarily involves designers interacting with the medical users such as clinical researchers, physicians, etc., as they attempt to discern all of the required knowledge across multiple resources in order to determine: the appropriate model concepts of the identified domains (Phase 1) at multiple levels; the vocabulary required for assisting in ontology model(s) development; and the way that the associations impact knowledge usage (Phase 2). This phase defines a *Glossary of Terms* (GT), a group of tables to encompass *types*, *classes, associations, attributes,* and *instances* (Sections 3.2 and 3.3) for the identified focus domains (Phase 1). The GT can be built by reusing methods proposed by Mariano Fernandez, et.al. (1997) - the methodology builds GT through informal textual analysis from texts and handbooks, formal and informal interviews with domain experts, etc. Asunción Gómez-Pérez, et.al. (1996) developed a data dictionary (DD) which is similar to GT which captures knowledge of a domain in terms of concept name, synonyms and acronyms, description, class features, instances, etc. This *Knowledge Acquisition* phase can be performed in parallel with *Specification*, *Design,* and *Analysis Phases*. While accumulating concepts from various sources, the designer may also want to document the concept in terms of the source origin, definition, references, usage, etc. RTs are vital to this phase, since it is not simply the information that is critical (Phases 1 and 2), but also the way the clinical researcher understands the information and any semantics s/he may be applying as a result of his/her own domain knowledge and practice in caring for patients as a physician over a long time period.

4.3.4. Phase 4: Specification

The Specification Phase is an important phase in the ontology development life cycle, similar to any software development life cycle models. In this phase, the ontology model(s) designer(s) interact with the end users such as medical experts, physicians, nurses, etc., and other potential stakeholders (e.g., HIT vendors) in order to defines the scope of the ontology models based on the domains identified (from Phase 1), their associations (Phase 1), and the critical supplemental knowledge (Phase 3). The scope assists the designers to focus on the domains without veering out of the final goal. For example, while Phase 1 identified *Disease* as one of the domains, this phase would define boundaries and specifications on Disease:

1. The ontology models will capture diseases in the domains of Mental Disorders, Respiratory System, Cardiac System, Circulatory System, Nervous System, Metabolic System and Skin related diseases to arrive at a solution akin to the domain model shown in Figure 3a;
2. The identified Diseases domain (Phase 1) concepts must be associated with concepts from the domains of Symptoms, Medication, Diagnostic and Test which are the associations in Figure 5 at detailed level, and the associations in Figure 9a at a higher conceptual level;
3. All the Disease entities must have a *UID* (Unique Identifier) and *MedicalName* to capture its scientific name and general common name.

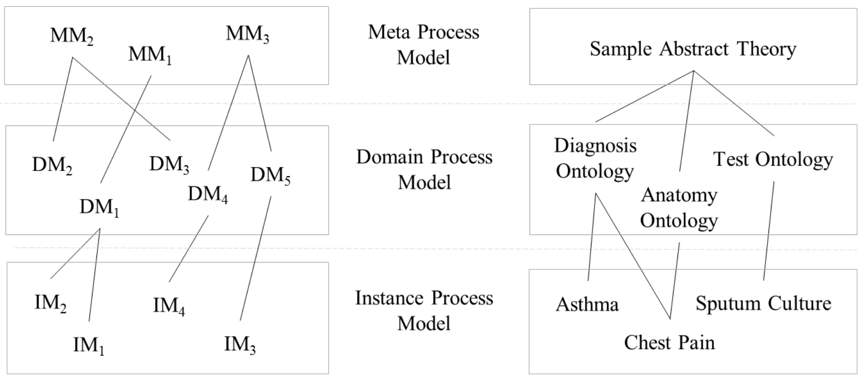
In general, the specification document is a cooperative and collaborative effort by all of the involved users in ontology model(s) design and development with significant oversight and interaction with designers. From a process perspective, the specification is evolved over time by these stakeholders using *incremental* and *iterative* approach. During this phase, the *Knowledge Acquisition Phase* (see Section 4.3.3) can be executed in parallel to capture vocabulary at various levels; we refer the reader back to Figure 9 to see the HOD^{2}MLC process. The RT's (Section 2) might assist the ontology designers and involved users in defining specifications on the knowledge system as RT's illustrate the concepts and interactions that may occur between them. For instance, RT1.A would like to know the impact of Metamorfin (a Metabolic Medication) on various diseases such as Type 2 Diabetes (a Metabolic Disease), CHF (a Cardiac Disease), stable Angina (a Respiratory Disease), etc. RT1.A will assist the designer and involved users to specify any mandatory/optional interactions between Metabolic Medication and Diseases domain modeling elements (Figure 5).

4.3.5. Phase 5: Design

The *Design Phase* is the core phase in the development life cycle, where the designer(s) identifies the *concepts* involved in the domain from the GT table built in the *Knowledge Acquisition Phase*. The *concept* encompassed can be classified into two types: *Abstract Type concepts* (abstract theory, Figure 5) and *Domain Model concepts* (see Figures 3, and 4). In software engineering, the process of defining models/concepts at various levels is called *Meta Process Modeling (MPM)* (Rolland, 1998). MPM is a type of metamodeling used in software engineering for developing modular and reusable process models as shown in Figure 11. The entities developed at the meta-level (MM_1, MM_2, and MM_3 in Figure 1) are utilized for developing domain process models (DM_1, DM_2, and DM_3) which are in turn employed to capture instance data (ID_1, ID_2, and ID_3) as previously shown in Figure 6. For example, the abstract theory and profile entities (Figure 5) can be placed at the MPM layer. The ontology domain models in the Figures 3 and 4 can be placed at the DPM layer. The *Model Concepts* are required for developing the domain model or schema for the ontology.

In order to represent the abstract theory and models at multiple levels, *top-bottom* decomposition of the domains identified in the *Problem Analysis Phase* (Phase 1) is performed by employing *Feature Driven Development (FDD)* as shown in Figure 12. FDD is a model-driven agile software process comprising of five activities: *overall model* where a high-level walkthrough

Figure 11. Layered architecture of Meta-Process Modeling (MPM)

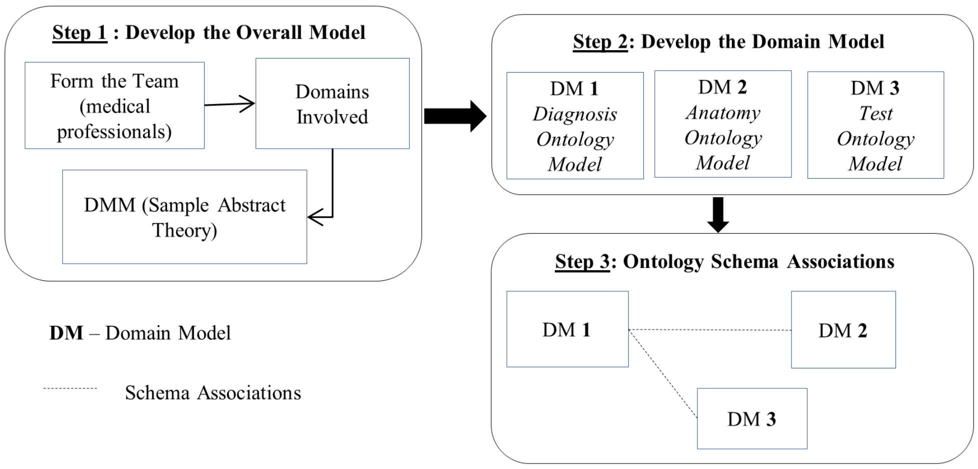


of the domain scope of the system and its context is performed; *Build Feature List* where a detailed domain walkthroughs are performed to decompose the domains into small groups which are presented for discussions where a feature can either be a class or a method call and in our model it's a concept; *Plan By Feature* where the generated features are prioritized for further the development plan; *Design By Feature* where a programmer selects a small group of features that are to be developed within two weeks; and, *Build By Feature* to develop the actual code for their classes.

Similarly, the FDD approach is oriented for HOD^{2}MLC as shown in Figure 12 and the following steps have to be followed: Step 1, a higher-level walkthrough of the *domains* identified from *Problem Analysis Phase* (Phase 1) should be performed to identify type concepts such as Disease, Medication, Symptom, hasSymptom, hasMedication, uid, hasName, etc., illustrated in Section 3.3 with The intent to define an abstract theory shown in Figure 5a or search and reuse existing abstract theory; Once the abstract theory and concept types have been defined, in Step 2 they can be decomposed into domain model classes (with respective attributes and associations) such as Respiratory System Diseases, Cardiac System Diseases, Metabolic System Diseases, etc., to define a ontology model(s) as shown in Figures 3. In Step 3, once the ontology model(s) have been built, the models can be described using OMV concepts and then be interconnected with one another to form ontology schema associations (see Figure 9).

The *iterative* nature of the cycle will assist designers in learning from the earlier phases and this resulting *incremental* process provides the degree of progress and partial output to the end users. The cycle is stopped once an agreement has been reached on structural and semantic aspects of the ontology model(s). The abstract theory (Figure 5a), ontology domain models (Figure 3) and the schema associations (Figure 8) developed using FDD approach at various levels (Figure 12) in this phase, provide structure and semantics to capture medical data. One final note, the HOD^{2}MLC model has two feedback loops in the process (see Figure 9) that promote an iterative and incremental design. The first feedback loop involves the *Analysis Phase, Specification Phase* and *Design Phase*. This loop provides flexibility to the overall lifecycle in terms of any dynamic changes to the specifications or the design model (e.g., to the RTs). For example, after the specifications have been written and the developers start to build the model, the user may need to add a specification (upon already defined specifications in Phase 4) such as, "Urine Test" must be included in the Tests domain consideration due to the presence of a new RT question. The feedback loop will help the designer to implement the new specification to

Figure 12. Feature driven development for ontology development


Step 1 : Develop the Overall Model
Form the Team (medical professionals)
Domains Involved
DMM (Sample Abstract Theory)
Step 2: Develop the Domain Model
DM 1 Diagnosis Ontology Model
DM 2 Anatomy Ontology Model
DM 3 Test Ontology Model
Step 3: Ontology Schema Associations
DM 1
DM 2
DM 3
DM – Domain Model
Schema Associations

the ongoing/developed domain model and analyze the new model without repeating the whole life cycle. As the *Knowledge Acquisition Phase* runs in parallel with these phases, the domain vocabulary can be acquired without revisiting the actual phase.

4.3.6. Phase 6: Analysis

The *Analysis Phase* is another important phase that has to be executed before *implementing* the abstract theory (Figure 9a) and developed domain model(s) (Figures 5, 6, and 7). In this phase, the designers and the end users revisit the *Specification Phase* to validate the design models developed in the *Design Phase*. The involved users (clinical researchers, domain experts, physicians, etc.) might also validate the domain model(s) to check if all the required semantic knowledge is captured. For instance, the domain expert will test the EHR model in Figure 3 and ontology models Figure 5, 6 and 7 against the RT's and their respective questions (RT1.A) in Section 2 and also verify if the general problem identified in the *Problem Analysis Phase* (Phase1) has been addressed. The developers might also check: the software principle *modularity* to verify if the developed conceptual abstract theory and domain models are simpler enough for domain expert understanding and portability; and, the software quality *reusability* to see if a conceptual model(s) can be reused in a different environment setting (by a different HIT system or by a new RT). These indicators don't have any measurable scale and is completely depended on developer's perspective and experience. The feedback loop involving *Specification, Design* and *Analysis Phases* as given in Figure 14 will bolster an incremental learning process, where the designers and end users can learn from the previous cycle. The *Knowledge Acquisition* insures that defined knowledge and semantics in the domain model is appropriately structured for specifying instances (Phase 7).

4.3.7. Phase 7: Implementation

The *Design Phase* provides the user with *structure* and *semantics* of the proposed solution to solve the *Overall Problem* and the *Analysis Phase* provides the feedback on abstract theory and domain model(s). However, in the realm of ontologies, an initial final solution is in the form of a domain vocabulary and is implemented using a formal language framework. For the domain vocabulary, we can use the *Glossary of Terms* (GT) developed in *Knowledge Acquisition Phase* responsible for gathering *all* of the useful and potentially usable domain knowledge and its meanings in terms of definition, usage, references, comments, etc. The sample GT table built for this life cycle model captures the medical vocabulary/data such as Fever, Redness of Eye, Cough, Headache, etc. (e.g. instances of General Symptoms, Figure 3a or Figure 4a) and other information such as the concepts definition, synonyms, etc. obtained from various medical sources (as discussed in Phase 2). This documentation is captured as rdfs:comment, rdfs:Label, etc. in the ODP Profile and OWL domain ontology from Section 3.3.

At this point, the designers have agreed to abstract type concepts and ontology model(s) which needs to be realized in a formal target language. Further, we are platform independent from an ontology modeling perspective, and now is the time to evaluate different implementation alternatives such as RDF/RDFS, OWL, UML, a simple database schema for *is-a* type vocabulary, etc. However, if the designers adopt *Meta-processing modeling* (Phase 5), the frameworks that support such a multiple layered architectural approach are UML profile and *OWL Domain Profile* (ODP) (Section 3.3); other implementation approaches are likely insufficient to do a lack of modeling abstraction and complexity. In our current scenario: the abstract theory obtained from FDD approach (Figure 12, Step 1) illustrated in Figure 5; the domain ontology models in Figure 3, are implemented using OWL as shown in Figure 4 using Protégé ontology editor; the dependencies in Figure 12 illustrate the OWL schema associations between the ontology models is implemented as shown in Figure 8 using the same Protégé editor. Thus, as proposed, the output of the HOD^{2}MLC model are the ontology abstract theory (and its type concepts), ontology domain model(s), ontology schema associations between the ontology models, and ontology vocabulary as shown in the Figure 9 and Figure 1.

4.3.8. Phase 8: Testing

The *Testing Phase* carries out a technical judgment of the ontology model(s) as realized in the chosen implementation alternative (see Section 4.3.7) by assessing the target software environment and its associated documentation. Testing can be carried out using the various proposed previous works such as: OWL Debugger and Repair framework for testing OWL based ontologies proposed by Kalyanpur (Kalyanpur, 2006). The implemented OWL Ontology models can also be tested using SPARQL (Prud'hommeaux & Seaborne, 2008) querying language for query accuracy. This is particularly true in regards to all of the aforementioned RTs and their questions in Section 2. This testing process for the scenario (RTs) is imperative, since it can identify situations and questions that aren't support and be able to cycle back to the appropriate phase in HOD^{2}MLC in order to revisit as much of the process as needed to address any deficiencies.

The second feedback loop in HOD^{2}MLC is between the *Implementation Phase* and *Testing Phase,* allowing designers to rectify the developed OWL ontology models for any inconsistency. For example, in larger ontologies, the user can define *CHF* and *Metabolic Disorders* to be disjoint (i.e., an *instance* or real world object can't belong to both of the classes at the same time). The *Testing Phase* identifies these problems and the feedback loop allows the designer to fix these implementation issues again without another iteration of the lifecycle. This loop is important since end users are playing a major role in both phases, which could result to redefining an RT, adding questions, or adding RTs, all of which would not just impact here, but revert to the first feedback loop in Section 4.3.5. History has clearly taught us that increments and iterations is the norm; the exception of a true waterfall model simply does not work in practice.

4.3.9. Phase 9: Maintenance and Documentation

The *Maintenance* and *Documentation Phase* is where the developed ontology has to be monitored for smooth and efficient performance of the knowledge system. The system administrators are primarily responsible for monitoring the system and as the system is continuously running, hence this represents the *maintenance* aspect of this phase. This phase is also responsible for maintaining version control for the ontologies developed. Various tools have been proposed for this purpose such as Redmond, et al. (Redmond, Smith, Drummond, & Tudorache, 2008), Sungawa (Sunagawa, Kozaki, Kitamura, & Mizoguchi, 2003), etc. that are all suitable to some degree to manage information in repositories that can allow roll-back and roll-forward in the event of changes or additions. The *documentation* aspect of this is responsible for detailed narrative report of the ontology concepts, axioms, and instanced and the primary contributor for this documentation is the *Knowledge Acquisition Phase*. The *Documentation* aspect of this is initialized with the *Implementation Phase* as this phase formally finalizes the concepts used in the ontology model(s) and runs in parallel with *Testing Phase*. The documentation of the ontology concepts is maintained both in textual format and encoded in the ontology model(s) as annotation using OWL annotation properties such as rdfs:comment, rdfs:isDefinedBy, rdfs:label, etc. Note that the documentation must also contain the definition of the problem (as presented in Section 2) as well as any additional stakeholder knowledge or semantics that have been defined in this process.

5. ONTOLOGY DEVELOPMENT ALTERNATIVES

In this section, we will discuss a select set of prominent ontology development life cycle models and techniques that have appeared in the literature, with an emphasis on the way that our approach compares. To accomplish this, in Section 5.1, we review seven other ontology models, and in the process, point out their similarities and differences against *HOD^{2}MLC* in Figure 9. Using this as a basis, in Section 5.2, using phases, we will compare and contrast these models to *HOD^{2}MLC* with the objective of understanding if the model completely (Full) supports the corresponding phase in *HOD^{2}MLC*, partially (Partial) supports corresponding the phase in *HOD^{2}MLC*, or is unable (None) supports the corresponding phase in *HOD^{2}MLC*.

5.1. Ontology Life Cycle Alternatives

This section details seven ontology development life cycle models and methodological approaches for building and maintaining ontologies. The first approach by Fernández, et al. (1997), has proposed the *Methontology* model which is composed by employing the following phases: *specifications* (*similar to Phase 4 in HOD^{2}MLC*)*; knowledge acquisition* phase which allows the developers to explore various source such as texts, handbooks, interviews, etc. for extracting knowledge for developing ontology vocabulary (*similar to Phase 3 in HOD^{2}MLC*); *conceptualization* phase is when the developers will structure the domain vocabulary and develop a conceptual model for the ontology (*similar to Phase 5 in HOD^{2}MLC*)*;* and, *integration, implementation* and *evaluation* (*similar to* Phases *2, 7 & 8 in HOD^{2}MLC*). The author chose an *Evolutionary Model* after contrasting it against *Waterfall* and *Spiral* models, and justified the choice by stating that the proposed model will help in expanding the ontology when needed. However, the proposed model and the phases leave crucial unanswered questions such as: How are the different phases inter-connected to form the model? and Are there any iteration between phases or whole model? Also, while enumerating the various phases in their work, the *integration* phase is responsible for reusing related domain ontology schema or conceptualization follows the *conceptualization* phase of the model, and as result, this renders the *integration* phase inefficient, as the *integration* phase is responsible for searching for existing conceptual models, look for reusable modules and use the modules in the ongoing development.

The second approach by Gomez-Perez, et al. (2006) (Gonzalez-Perez & Henderson-Seller, 2006), proposed a *method **not** a complete development cycle* for an ontology with the following steps. First, their work develops a requirement and specification document (*similar to Phases 1 and 4)*. Second, they conceptualize the domain terminology by constructing the following tables: *Data Dictionary* (*similar to Phase 3 in HOD^{2}MLC*) which identifies domain classes and instances; *Concept Classification Tree* which groups the identified concepts; *Tables of Constants* which list all of the constants of the domain; *Tables of Instance and Class Attributes* that lists all classes and instances of the domain; and, *Tables of Formulas* which describes formulas used to infer numerical values of attributes. However, much of this information is instance-based, as opposed to our highly focused abstract type and schema modeling approach. Further, the steps enumerated in the method are similar to waterfall model where the methodology phases are executed sequentially without any feedback loop or iteration and thus inheriting the drawbacks of the waterfall model.

The third approach by Uschold, et al. (1996) (Uschold & King, 1996) and Uschold (1998), proposed the Enterprise Ontology (EO) project which consists of four phases: *purpose*, *building*, *evaluating* and *documentation* (*similar to Phases 1, 7, 8* & *9 in HOD^{2}MLC*). In their first phase, the purpose of the ontology is identified, i.e., to find out why the ontology is being built and what its intended uses are, equivalent to *requirements* phase. In the second phase, there is the building of the ontology itself which is divided into three parts: capture, coding and integrating which is equivalent to combination of *knowledge acquisition* and *implementation* phases. In the evaluation phase, the intent is to check that the ontology fulfills the requirements and that it does not contain any unnecessary things, which is same as our *Analysis* phase. Their last phase is the *documentation* phase in which the ontology should be documented in some way. However, the work doesn't provide any model as to the way that the phases are connected and no good guidelines published as any optimal way to achieve the second phase.

The fourth approach Gruninger, et al. (1995) (Grüninger & Fox, 1995) have proposed TOVE project whose primary goal is to answer enterprise queries to existing or future usecase scenarios. Based on these scenarios, a set of questions named *informal competency questions* is raised that the ontology has to answer (*similar to Phase 1 in HOD^{2}MLC*). The motivating scenario and competency questions (similar to RT's) provide the developer with the information needed to develop or extend the ontology, i.e., the set of questions form the *requirement* phase. The next step is to specify the terminology of the ontology by using first-order logic forming the *conceptualization* and *implementation* phase (*similar to Phases 4* & *7 in HOD^{2}MLC*). The author details the various phases a developer has to address for building an ontology, but doesn't provide a life cycle or a model connecting these various phases.

The fifth approach, also by Uschold (1998) (Uschold, The Enterprise Ontology, 1998), presents a unified methodology by combining methodologies from the EO and TOVE projects. The first step is to define the purpose of the ontology which can be done in several ways, e.g. to identify the intended users, or as in the TOVE project with motivating scenarios and competency questions, to form the user *requirements* document (*similar to Phase 1 in HOD^{2}MLC*). In the *conceptualization* phase, the developer should decide what level of formality the ontology has to have and find the concepts and the relations among them (*similar to Phase 5 in HOD^{2}MLC*). The author describes four different approaches for constructing the ontology. First, use an ontology editor to define terms and axioms. Second, do the previous steps and then begin a formal encoding. The third approach is to produce an intermediate document that consists of the terms and definitions that appeared in the previous step; this document can be the final result, or be specification of the formal code or its documentation. The fourth and final approach is to identify formal terms from the set of informal terms. Finally, the approach has a revision cycle, where the developed ontology is compared to the competency questions or the user requirements (*similar to Phases 8 & 9 in HOD^{2}MLC*). Similar to the previous work, the author doesn't provide any model interconnecting the phases. Again, there is a focus on creating the instance of the ontology very early in the process, where our approach defers this so that an ontology model with a schema can be defined and then potentially reused.

The sixth approach by Noy, et al. (2001) (Noy & McGuinness, 2001) describes a way to develop an ontology *iterative* methodology starting with rough concepts and then revising and filling in the details. The first step in their suggested methodology is to determine the domain and the scope of the ontology fulfilling the *requirements* phase (*similar to Phases 1 & 4 in HOD-^{2}MLC*). The second step is to consider whether existing ontologies are available to use, and if so, the way to utilize them (*similar to Phase 2 in HOD^{2}MLC*). A list of all the terms that could be needed or used is produced in the third step. The class hierarchy is represented using an "is-a" relation, siblings should have the same level of generality, and also guidelines regarding when to introduce new classes or instances are given. Now the classes are defined, i.e., the terms and the relations, in the fourth step, the properties of the classes need to be specified (attributes) (*similar to Phase 5 in HOD^{2}MLC*). One important task in this step is to check whether some relations are inverse or not (e.g., boolean operators in description languages such as Frames, OWL, etc.), and whether a default value for an attribute could be useful. After this, in the next step, the value type of both the classes and the class properties are defined, this includes cardinality, domain and range. Finally, in the last step, the individual instances are created. Their work is on the border between instance and typed based approaches to ontology design and development.

Finally, the approach of UPON (Nicola, Missikoff, & Navigli, 2005) is a methodology that adopts a *unified process* for ontology development. The methodology has cycles, phases, iterations, and workflows. Each cycle consists of four phases (inception, elaboration, construction, and transition) and results in the release of a new version of the ontology. Each phase is further subdivided into iterations consisting of five workflows: requirements, analysis, design, implementation and test (*similar to Phased 1, 6, 5, 7 & 8 in HOD^{2}MLC*). Workflows and phases are orthogonal in that the contribution of each workflow to an iteration of a phase can be more or less significant. The unified approach is a generic software development methodology adopted for designing any software application. The authors simply adopt the process towards ontology development without evaluating if all of the cycles and iterations are required for developing robust ontology models. Similar to the previous work, UPON is on the border between instance and typed based approaches to ontology design and development.

5.2. HOD^{2}MLC vs. Ontology-Based Alternatives

This section presents our proposed HOD^{2}MLC and its nine phases (see Section 4.3) which is compared and contrasted with the seven ontology alternatives presented in Section 5.1. We take a qualitative approach in making our assessment, and we are biased towards HOD^{2}MLC which is broader in its approach to include abstract concepts, types, models, and schemas, and as a result has more phases as compared to the seven alternatives. To compare these models, we define three qualitative criteria to evaluate: *None* – the methodology does not support the phase; *Partial* **-** the methodology may have partial implemented the phase; and *Full* – the methodology has the phase in its life cycle; Table 1 summarizes this discussion.

Table 1. Evaluation of ontology methodologies alternatives against HOD²MLC phases

Phases	**Ontology Life Cycle Models**							
	Methontology	**Fernandaz**	**EO Project**	**TOVE**	**Uschold**	**Noy**	**UPON**	**HOD²MLC**
Problem Analysis	Partial	Full	Full	Full	Full	Full	Full	Full
Ontology Integration	Partial	None	Partial	Full	None	Partial	None	Full
Knowledge Acquisition	Full	Full	Full	Full	Full	Full	None	Full
Specifications	Full	None	Partial	Partial	Partial	Partial	Full	Full
Design	Partial	Partial	Full	Full	Full	Full	Full	Full
Analysis	None	None	Partial	None	None	None	Full	Full
Implementation	Full	None	Full	Partial	Partial	Partial	Full	Full
Testing	None	None	None	None	None	None	Full	Full
Maintenance / Documentation	Partial	None	Partial	Partial	None	None	None	Full
Model Adopted	Evolutionary	None	None	None	None	Iterative	Unified Process	Agile Process

From Table 1, all of the alternatives fulfill the *problem analysis phase* either through *requirements* generation, *usecase* scenarios (Uschold & King, 1996), formulating *competency questions* (Uschold, 1998; Grüninger & Fox, 1995; Uschold & King, 1996), or through *data abstraction.* The *ontology integration phase* is not given prominence in any alternatives except TOVE, which is primarily responsible for reuse of existing ontologies. The *knowledge acquisition phase* is the most focused phase in all of the alternatives except UPON, as they all maximize the ontology domain vocabulary. The seven alternatives either combine the *specifications* with other phases, or assume they are given to build the ontology, with Methonotology and UPON with dedicated phase for defining concrete specifications, but again contrasting to our work with a more instance based approach. HOD²MLC also has a focused *specification phase* primarily for defining boundaries and additionally for analyzing the developed ontology domain model. The *Design phase* has varied importance across alternatives, achieved in EO Project, TOVE, NOY, and UPON, primarily centered on developing domain models by using metamodels such as UML, RDF/RDFS, OWL, etc., HOD²MLC uses a layered architecture (Figure 10) and FDD (Figure 11) for designing domain models and the target implementation is our extensions to OWL (see Section 3). The *Analysis phase*, a high priority in HOD²MLC to validate the domain model with the specification phase, has limited consideration by other the alternatives except UPON. The *implementation* phase is not taken into consideration in all but Uschold (1998) and Uschold, et al. (1996), with the ontology realized at some point, without a consideration of timing. All of the alternatives have no formal *testing* except UPON, while HOD²MLC uses various frameworks for evaluating and testing the realized ontology. Various alternatives take different approaches for maintaining and documenting the developed ontology, as there are no set standards for executing them.

6. CONCLUSION AND ONGOING RESEARCH

This paper has presented our work on a the *Hybrid Ontology Design & Development Model with Lifecycle, HOD^{2}MLC* which leverages an agile software process in order to design ontologies at a more abstract and conceptual level, upgrading the ontology creation process from one that is data intensive to an approach with an emphasis on modeling and design. The work towards software development process for ontologies has been motivated by our examination of the health care domain, where many of the HIT systems (EHRs, PHRs, etc.) that need to interoperate have system specific ontologies that makes information difficult to exchange. This was discussed in detail with the clinical scenario that was presented in Section 2 that included the evolving role of ontologies in HIT and health information exchange. In support of this paper, we have leveraged our work briefly reviewed in Section 3 that has extended OWL (Saripalle, Demurjian, Algarín, & Blechner, 2013) with a UML meta-model and associated modeling concepts. Using this as a basis, in Section 4, we presented *HOD^{2}MLC* in detail, explaining its overall process and detailing each of the phases and their inter-relationships. To facilitate the discussion, we utilized the clinical scenario and the research topics (and associated questions) to be posed by a clinical researcher, coupled with the associated ontology design that was presented in Section 3 based on the scenario. As a result, the discussion is Section 4.3 clearly demonstrates the process oriented nature of our work in support of creating ontologies at a conceptual or model level that are reusable. To place our work into its proper perspective, in Section 5, we reviewed seven other ontology development models and compared and contrasted these efforts to our own work. In summary, *HOD^{2}MLC* takes our current work Saripalle, et al. (2013), Saripalle, et al. (2011) to the next level by casting our extensions as software engineering artifacts, placing them in a position to be shared by a wide range of stakeholders for ontology definition (domain conceptualization and vocabulary) that promote a solution that is created, imported, exported and re-used using different frameworks, tools and techniques.

In terms of related work, we are proceeding in a number of areas. First, we have worked on extending ontologies with a modeling construct akin to software design patterns, and have published preliminary results on the work (Saripalle & Demurjian, 2011). We are interested in seeing if ontology knowledge patterns can be defined akin to software creational design patterns, but augmented with semantics. These ontology patterns can then be reused at an even higher level of abstraction, i.e., akin to Model View Controller (MVC), there may be patterns about symptoms, diagnosis, and treatments that are more abstract and semantically enhanced. Second, we are working on enhancing the design and development of our OWL Domain Profile (ODP) framework as presented in Section 3.3 as an extension to Protégé Ontology editor (which can create/load/save various ontology formats such as Frames, RDF/RDFS, OWL, etc.) and are also working on a Domain Profile parser algorithm which essentially authenticates and validates the imposing of the profile entities onto the ontology model concepts. The ODP extension is complaint with Protégé Java framework and we are also planning to define a Java API for accessing ODP's from an external Java based application, and on extending the ODP framework UI to define ontology patterns using the defined ODP concepts. Thirdly, we have also implemented owl:Attribute extension (Section 3.2) by extending the Protégé Ontology editor to define classes similar to UML classes. Currently, our focus is to define Ontology Architectural Patterns (OAP) which governs the overall architectural design and development of the ontology models and associated ontology-based system. These ontology architectural patterns can be compared to software structural design patterns which ease the architectural design issues by identifying efficient ways to realize interactions between involving entities. The research will also include a tool for implementation of the proposed OAP's on developed ontology models.

REFERENCES

Allemang, D., & Hendler, J. (2011). Semantic Web for the Working Ontologist, Second Edition: Effective Modeling in RDFS and OWL. (2nd, Ed.) Morgan Kaufmann.

Bodenreider, O. (2004). The Unified Medical Language System (UMLS): Integrating Biomedical Terminology. *Journal Nucleic Acids Research*, *32*(1), 267–270. doi:10.1093/nar/gkh061 PMID:14681409

Boone, K. (2011). *The CDA book*. Springer. doi:10.1007/978-0-85729-336-7

Clancey, W. J. (1985). Heuristic classification. *Journal in Artificial Intelligence*, *27*(3), 289–350. doi:10.1016/0004-3702(85)90016-5

Craig, L. (2003). *Agile and Iterative Development: A Manager's Guide* (1st ed.). Addison-Wesley Professional.

Demurjian, S., Saripalle, R., & Behre, S. (2009). An Integrated Ontology Framework for Health Information Exchange. *Proceedings of 21st International Conference on Software Engineering and Knowledge Engineering*, (pp. 575-580). Boston.

Docherty, M. (2005). *Object-Oriented Analysis and Design: Understanding System Development with UML 2.0*. Wiley.

Gonzalez-Perez, C., & Henderson-Seller, B. (2006). An Ontology for Sofware Development Methodologies and Endeavours. In C. Coral, F. Ruiz, & M. Piattini (Eds.), *Ontologies for Software Engineering and Software Technology*. Springer. doi:10.1007/3-540-34518-3_4

Grüninger, M., & Fox, M. (1995). Methodology for the Design and Evaluation of Ontologies. *Proceeding of Workshop on Basic Ontological Issues in Knowledge Sharing*.

Guizzardi, G. (2010). Theoretical foundations and engineering tools for building ontologies as reference conceptual models. *Semantic Web*, *1*, 3–10.

Hartmann, J., Palma, R., & Sure, Y. (2005). OMV– Ontology Metadata Vocabulary for the Semantic Web. *Proceeding of International Workshop on Ontology Patterns for the Semantic Web*.

Kaihong, L., Hogan, W. R., & Crowley, R. S. (2011). Natural Language Processing Methods and Systems for Biomedical Ontology Learning. *Journal of Biomedical Informatics*, *44*(1), 163–179. doi:10.1016/j.jbi.2010.07.006 PMID:20647054

Kalyanpur, M. (2006). *Ph.D. Disseration - Debugging and Repair of OWL ontologies*. University of Maryland.

Konstantinou, N., Spanos, D.-E., & Mitrou, N. (2008). Ontology and Database Mapping: A Survey of Current Implementations and Future Directions. *Journal of Web Engineering*, *7*(1), 1–24.

Kruchten, P. (2003). *Rational Unified Process: An Introduction*. Addison-Wesley Professional.

Kuhn, M. (2010). Modeling vs Encoding for semantic web. *Journal of Semantic Web-Interoperability, Usability. Applicability*, *1*(1), 11–15.

Liu, S., Ma, W., Moore, R., Ganesan, V., & Nelson, S. (2005). RxNorm: Prescription for electronic drug information. *IEEE IT Professional*, *7*(5), 17–23. doi:10.1109/MITP.2005.122

Nicola, A., Missikoff, M., & Navigli, R. (2005). A Proposal for a Unified Process for Ontology building: UPON. *Proceeding of 16th International Conference on Database and Expert Systems Applications*. doi:10.1007/11546924_64

Noy, N., & McGuinness, L. (2001). *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford Knowledge Systems Laboratory.

Palmer, S. R., & Felsing, J. M. (2002). *A Practical Guide to Feature-Driven Development* (1st ed.). Prentice Hall.

Prud'hommeaux, E., & Seaborne, A. (2008). *SPARQL Query Language for RDF*. Retrieved from http://www.w3.org/TR/rdf-sparql-query

Redmond, T., Smith, M., Drummond, N., & Tudorache, T. (2008). Managing Change: An Ontology Version Control System. *Proceeding of 5th International Workshop of OWL Experiences and Engineering (OWLED)*.

Rolland, C. (1998). A Comprehensive View of Process Engineering. *Proceeding of the 10th International Conference on Advanced Information Systems Engineering*, (pp. 1-24).

Saripalle, R., & Demurjian, S. (2011). Towards a Hybrid Ontology Design and Development Life Cycle. *Proceeding of 2012 International Conference on Semantic Web and Web Services*, (pp. 45-52). Las Vegas.

Saripalle, R., Demurjian, S., Algarín, A., & Blechner, M. (2013). A Software Modeling Approach to Ontology Design via Extensions to ODM and OWL. *International Journal on Semantic Web and Information Systems*, *9*(2), 62–97. doi:10.4018/jswis.2013040103

Saripalle, R., Demurjian, S., & Behre, S. (2011). Towards Software Design Process for Ontologies. *Proceeding of 1st International Conference on Software and Intelligent Information*.

Sunagawa, E., Kozaki, K., Kitamura, Y., & Mizoguchi, R. (2003). An Environment for Distributed Ontology Development Based on Dependency Management. *Proceeding of 2nd International Semantic Web Conference*, (pp. 453-468). doi:10.1007/978-3-540-39718-2_29

Uschold, M. (1998). The Enterprise Ontology. *Journal of The Knowledge Engineering Review*, (pp. 31-89).

Uschold, M., & King, M. (1996). Building Ontologies: Towards a Unified Methodology. *Proceeding of the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems.*

ENDNOTES

[1] Logical Observation Identifiers Names and Codes, http://loinc.org/
[2] International Classification of Diseases, http://www.who.int/classifications/icd/en/
[3] A measurement of the amount of oxygen being carried in the blood, normally around 98%
[4] Protégé Editor, www.protege.stanford.edu, 2015

Rishi Kanth Saripalle received his PhD in Computer Science & Engineering from University of Connecticut, under the supervision of Dr. Steven A. Demurjian. His research interests are primarily in biomedical and health informatics modeling, tools and frameworks, software engineering and modeling, ontology design and development, clinical data engineering, and secure software engineering. His thesis focused on imposing software engineering, modeling and life cycle concepts on ontologies, with the end purpose of defining a software engineering based approach to design and development of ontologies to ease semantic interoperability issues. Dr. Saripalle has received his Masters in Computer Engineering from the University of Massachusetts, where his research focused on developing ontological knowledge bases for aiding biomedical decision support system.

S. A. Demurjian is a Full Professor and Director of Graduate Studies in Computer Science & Engineering at the University of Connecticut, and co-Director of the Biomedical Informatics Division. His research interests include secure-software engineering, security for biomedical applications, and security-web architectures. Dr. Demurjian has over 150 archival publications, in the following categories: 1 book, 2 edited collections, 54 journal articles and book chapters, and 98 refereed conference articles.

M. Blechner is an Assistant Professor of Pathology and Laboratory Medicine and Director of Pathology Informatics and Transfusion Medicine and a faculty member of the Biomedical Informatics Division at the University of Connecticut Health Center. His major research interests are in medical knowledge representation and ontologies and their use in intelligent tutoring systems. Other areas of research interest and expertise are in computerized decision support for laboratory medicine, data warehousing and optimization of clinical laboratory data for research and patient safety initiatives.

Thomas Agresta is the Director of Medical Informatics for Family Medicine, Director Clinical Informatics for the Center for Quantitative Medicine and Section Leader in Informatics for the Connecticut Institute for Primary Care Innovation at the University of Connecticut. He is a seasoned Family Physician educator, administrator, researcher and innovator with a history of bringing together multidisciplinary teams to focus on developing novel methods for creating, using and evaluating technology in both clinical and teaching settings. He has a bachelors in Biomedical Engineering from Stevens Institute of Technology, a medical degree from New Jersey Medical School and a masters in Biomedical Informatics from Oregon Health Sciences University.