Chapter One

Introduction

1.1 Background

Diabetes is a disease in which the body does not produce or properly use insulin, a hormone that is needed to convert sugar, starches, and other food into energy needed for daily life. The cause of diabetes is a mystery, although both genetics and environment appear to play roles is a chronic, disease that has no cure (USDHHS, 2010)[1].

Knowing the criticality of diabetes in the current human life globally, it is important to have appropriate guideline in managing the disease. The disease being categorized as no cure ailment requires a lifelong treatment and monitoring. The current human life style has huge demand and complication in monitoring the disease, which leads to chronic complication in most cases. As such, an appropriate knowledgebase monitoring system is found to be necessary (IDF, 2010)[1].

According to the Irish Nutrition and Dietetic Institute Fact Sheet (2007), there are two major types of diabetes: -

- ***Insulin-Dependent (type 1)*** an autoimmune disease in which the body does not produce any insulin, most often occurring in children and young adults. People with type 1 diabetes must take daily insulin injections to stay alive.
- ***Non-Insulin-Dependent (type 2)*** A metabolic disorder resulting from the body's inability to make enough or properly use insulin, it is the most common form of the disease[1].

The main symptoms of diabetes include body weakness. This is characterized by dizziness, dry mouth, decreased in appetite, nausea or vomiting. Other features of the disease are headache, lack of sleep at night. Similarly, patients are known to complain of morning headaches, nightmares, night sweat, light headedness, body shake, body weakness, severe hunger, loss of consciousness (WDD, 2006)[1].

The treatment of diabetes in aged patients with healthy body weight requires making lifestyle choices in diet, exercise, and other health habits. These help to improve blood sugar control, and prevent complications of long life diabetes. The self care involves the habits: -

- **Diet:** is the key to healthy body due to its ability to control blood sugar levels and prevent avoidable diabetes complications.
- **Exercise:** is necessary for patients as it decreases the risk of developing diabetes. Such physical activity can also reduce the risk of developing associated ailments such as heart disease, stroke, kidney failure, blindness, and leg ulcer.
- **Alcohol Use:** it is highly recommended that diabetes patients should avoid the use of alcohol. This is to moderate or reduce the risk of developing high or low blood sugar levels. Alcohol consumption is known to be associated with nerve pain called neuritis, and increase in triglycerides. Because alcohol is processed in the body very similarly to the way fat is processed, and alcohol provides almost as many calories. Excessive alcohol use is a known as one of the risk factor for type 2 diabetes[1].

- **Smoking:** can easily increase the risks of diabetes patients' health complications. Smoking effectively damages blood vessels and contributes to heart disease, stroke, and makes the limbs poor for movement, Many different structures can be used for formalizing and organizing knowledge. Particularly, in recent years, ontologies have received great attention. Ontologies are formal descriptions of a domain knowledge based on concepts and their relationships (Castilho,Lopes and Tacla, 2008)[1].

They are efficient for creating a common vocabulary between experts in order to share and reuse knowledge using accurate semantic (meaning). In essence, this study discusses the role of ontology within the health domain. An ontology, particularly in the health field arises out of a perceived need for a controlled vocabulary. It is an enriched contextual form for representing domain knowledge (Dillion et al, 2008)[1].

1.2 Problem Statement

Large national and international epidemiological studies have identified the rising number of diabetic individuals in industrialized as well as developing countries. However, over 300 million people in the world have diabetes recorded in 2010, and other 300 million are expecting to be at high risk of diabetes in 2030. Currently, the following problems are typical of developing societies.

i. Managing diabetes patients have been a big challenge to most developing countries in terms of cost.

ii. Lack of self management awareness among diabetes patients.

iii. Unmanaged diabetes leads severe vascular related risks and diseases.

iv. It has been a challenge among working adults in managing diabetes in daily life due its complexity.

1.3 Research Objectives

The main objectives of this research are:

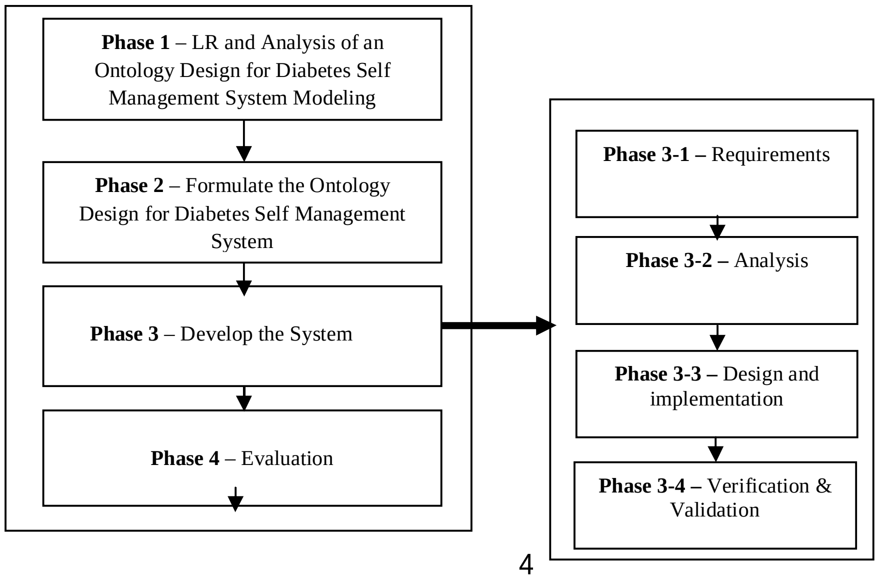
1. To build diabetes self management ontology.
2. To design a framework of self-managed internet based diabetes risk management.
3. To validate the designed diabetes system.

1.4 Significance of the study

An automated self management diabetes manager can be an effective health management solution, complement scholarly works on health automated systems, fills a gap on existing knowledge – developing a new ontological system & contributes to ongoing research for effective management of diabetes patients.

1.5 Research Methodology

This research shall be carried out in five phases as illustrated in the following research methodology framework (Figure 1.1):



Phase 1 – LR and Analysis of an Ontology Design for Diabetes Self Management System Modeling
Phase 2 – Formulate the Ontology Design for Diabetes Self Management System
Phase 3 – Develop the System
Phase 4 – Evaluation
Phase 3-1 – Requirements
Phase 3-2 – Analysis
Phase 3-3 – Design and implementation
Phase 3-4 – Verification & Validation

Figure: 1.1: Research Methodology Framework

Phase 1 – LR and Analysis of Diabetes Self Management System

This phase involves evaluation of existing Diabetes Self Management System and literature reviews on academic journals, books, and information search on groupware tools, Case Based Reasoning (CBR) and other tools supporting Diabetes Self Management System activities.

Phase 2 -Formulate a Diabetes Self Management System

The next step is to formulate new model of applying CBR techniques for collaborative Diabetes Self Management System based on earlier literature reviews and pre-survey results, areas of concerns are noted

This stage is in progress.

Phase 3 -Develop the System

The system development shall include the following steps:

- ✓ Requirements
- ✓ Analysis
- ✓ Design and Implementation
- ✓ Verification and Validation

Phase 3-1 –Requirements

Our entire system requirements will be identified in this stage.

Phase 3-2 –Analysis

Our entire system requirements will be analyzed in this stage.

Phase 3-3 - Design and Implementation

Based on the current ontology design (This stage has been completed), the next step is to develop the system. The platform and development tool shall be determined at a later stage.

Phase 3-4 - Verification and Validation

This is an iterative process, whereby the tools are tested and reworks and enhancements are carried out. We expect several builds are required to stabilize the application.

Phase 4 -Evaluation

In this stage, we shall identify the pilot site for implementation, conduct training and implement the proposed system.

1.6 Research Tools

1.6.1 Protégé platform

Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies.

1.6.2OWL (Ontology Web Language).

is an ontology language for the Semantic Web with formally defined meaning. OWL ontologies provide classes, properties, individuals, and data values.

1.7 Research Organization

Chapter One introduce the concept of Ontology and diabetes. Chapter Two gives a literature review about the domain. Chapter Three gives a brief overview of the Protégé OWL. Chapter Four focuses on building the ontology and using a Description Logic Reasoner. Chapter Five describes the research Results, and Recommendation.

Chapter Two

Literature Review

This chapter discusses the issues of Ontology's ,ontology uses, and ontology engineering in extra level of details. Also shows some ontology studies in other different domains.

2.1Ontologies

Although it is currently somewhat of a catchword, there exists some disagreement amongst researchers as to the definition of ontology (Noy and Hafner, 1997 and Pisanelli et al., 2002). For this reason and the fact that the term often takes on different meanings in different contexts, a short overview of the concept of ontology is in order[2].

2.1.1Definitions of Ontology:

- An ontology defines the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary.
- An ontology is an explicit specification of a conceptualization.
- An ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base.
- An ontology provides the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base.
- An ontology is a formal, explicit specification of a shared conceptualization[2].

2.1.2 An ontology of Ontologies

Depending on their intend use, ontologies come in variety of types, varying in their formality and their specificity.

❖ Informal Ontologies

- **Controlled vocabulary** perhaps the simplest kind of ontology is controlled vocabulary. In a controlled vocabulary, we make use of a few predefined keywords to classify entities: no perhaps, properties, or axioms. In some domains, this is sufficient.
- **Terms/ glossary** here, we have a list of terms, as with a controlled vocabulary, but some attempt is made (typically with natural language, e.g. an English explanation) to define the meaning of these terms. However, the computational power of such an ontology is roughly that of a controlled vocabulary, since we cannot in general compute with the natural language explanation. The value of the explanation is therefore usually for the ontology designer.
- **Thesaurus** a thesaurus defines synonyms: terms that have the same meaning. Thus, if you want to find a web service that provides weather forecasts, it might be useful to know that ‘meteorological forecast’ means the same thing.
- **Informal ‘is-a’ taxonomies** here we think of controlled vocabularies organized into an informal hierarchy. We find such hierarchies on web sites such as Amazon.com, for example, where goods for sale are organized into loose hierarchies. Typically the hierarchies are not formal hierarchies, because related goods (e.g. cameras and camera bag) are collected together

in the same place, without any formal definition of how or why the goods are clustered in this way [3].

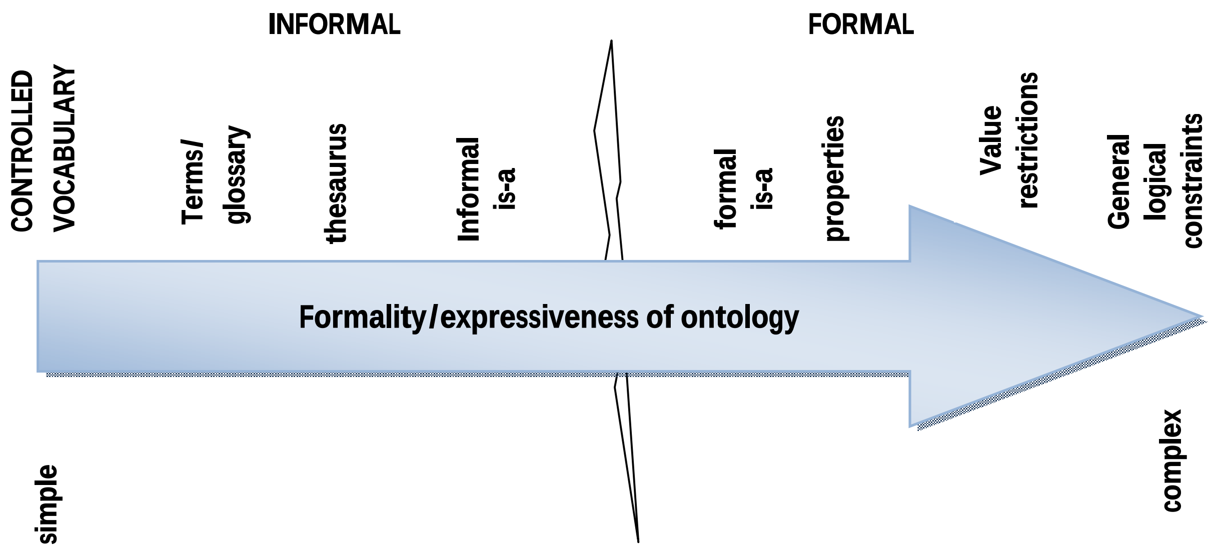


Figure 2.1: the ontology spectrum from informal up to formal expressive [3]**.**

❖ Formal Ontologies

Where some attempts is made to give the terms in the ontology some formal semantics.

- ***Formal 'is-a' taxonomies*** here, we explicitly define subsumption relationships between classes as shown in (figure 2.2).
- ***Properties*** here, we now allow classes to have properties, and together with the subsumtion relation, this permits us to draw conclusions about the properties of classes.
- ***Value restrictions*** value restrictions give additional information about relationships; for example, a typical restriction might say that 'every person has exactly one birth mother'[3].

- ***Arbitrary logical constrains*** finally, we might have ontologies with arbitrary logical constrains. Such constrains go beyond value restriction, taxonomical hierarchies, etc. In general, such constrains allow us a great degree of precision when defining an ontology. The drawback with such constrains is that, in general allowing arbitrary logical expressions leads to very high computational complexity (and even undecidability) with respect to reasoning [3].

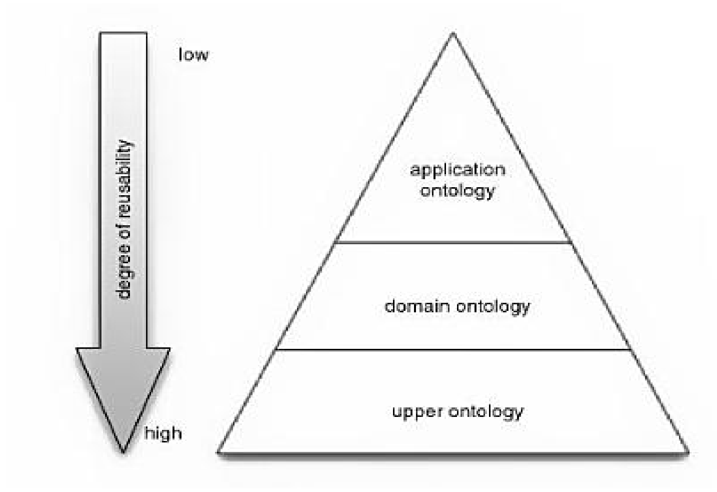


Figure 2.2: the ontology hierarchy: from (reusable) at the bottom, (not very reusable) at the top [4].

As well as distinguishing ontologies based on their formality and the types of information (Figure 2.3), can also usefully distinguish ontologies based on their role in an application (Figure 2.4). At the bottom, the most general kind of ontology: the so-called '*upper ontology* ' (even though it appears at the bottom in (Figure 2.4) such an ontology might start out by defining the most general class and then define classes that specialize this[3].

A *domain ontology* defines concepts appropriate for a specific application domain, this research is domain ontology. For example, it might define concepts relating to diabetes terminology, and be used by a number of applications in the area of diabetes. Note that domain ontology will typically build upon and make use of concepts from an upper ontology: this idea of *reuse* of ontologies is very important, as the more applications use a particular ontology, the more agreement there will be on terms[3].

Finally, *an application ontology* defines concepts used by a specific application. Again, it will typically build upon a domain ontology and it turn upon some upper ontology. Concepts from an application ontology will not usually be reusable: they will typically be of relevance only within the application for which they were defined [3].

2.2 Ontology languages

2.2.1 XML – ad hoc ontologies

Arguably the simplest ontologies to create and use are *ad hoc* ontologies: create with little effort, for a specific purpose, usually with a short expected period of use. Often such ontologies take the form of a controlled vocabulary, and XML is usually the language of choice of such ontologies.

The extensible markup language (XML) is not an ontology language, although it can be directly used to define simple ontology in an informal way. It is best thought of as a kind of extension to HTML, which in a nutshell allows us to define our own tags and document structures [3].

xml
```
    <Declaration>
        <Class IRI="#Pre_Diabetes"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Type1"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Type11"/>
    </Declaration>
        <Class IRI="#Foods "/>
    </Declaration>
```

Figure 2.3: xml code example

2.2.2 Ontolingua

Ontolingua is originally an interlingua for ontology representation and sharing developed by KSL (Knowledge Systems Lab) at Stanford University. It is designed by adding frame-like representation and translation functionalities to KIF (Knowledge Interchange Format)[KIF] which is a logic-based interlingua for knowledge representation. It can translate from and to some description logics languages such as Loom, Epikit, etc. Ontolingua itself does not have inference functionality. It has currently developed into a development environment which Provides a set of ontology development functions (browse, create, edit, modify and use ontologies) and a library of modular and reusable ontologies. Although it had been a key language for ontology representation for years since its development, it is not active recently because of the advent of XML family languages[3]

2.2.3 Resource Description Framework schema (RDFS)

Resource Description Framework (RDF) is a framework for metadata description developed by W3C (WWW Consortium). It employs the triplet model <object, attribute, value>, well-known in AI community, in which object is called resource representing a web page. A triplet itself can be an object and a value. Value can take a string or resource. Object and value are considered as a node and attribute as a link between nodes. Thus, an RDF model forms a semantic network. RDF has an XML-based syntax (called serialization) which makes it resembles a common XML-based markup language. But, RDF is different from such a language in that it is a data representation model rather than a language and that the XML's data model is the nesting structure of information and the frame-like model with slots.

Although RDF has been designed for metadata representation model, it can be used as a general-purpose knowledge representation, which might be apparent from the fact that it is a kind of semantic network model[3].

2.2.4 Ontology Web Language

Web Ontology Language (OWL) is also a language developed by W3C. OWL is designed to make it a common language for ontology representation and is based on DAML+OIL. OWL is an extension of RDF Schema and also employs the triple model. Its design principle includes developing a standard language for ontology representation to enable semantic web, and hence extensibility, modifiability and interoperability are given the highest priority. At the same time, it tries to achieve a good trade-off between scalability and expressive power[3].

OWL is a collection of several XML-based ontology frameworks, within which ontologies in these various frameworks can be expressed. Specifically, there are three main level of OWL, as follows:

- **OWL Lite.** This is simplest (least expressive) variant of OWL, which supports only basic ontology features. In particular, OWL Lite places a number of restrictions on the type of axioms one can write. The point about these restrictions is that they result in a language that is computationally more tractable (and is also somewhat easier for human to use and understand) than more expressive OWL variants.
- **OWL DL.** This language extends the properties of OWL Lite. The features of OWL DL were carefully chosen so that the language corresponds exactly to a particular formalism know as description logic.
- **OWL Full.** This is very expressive framework, providing many features for defining ontologies; however, in its full glory, the framework is so rich that many reasoning problems with OWL full (such as consistency checking) are understandable [3].

Because OWL is the most recent development in standard ontology languages and providing many features for defining ontologies, it will be adopted to be the language used to designing the ontology for diabetes.

2.3 Ontology components

Contemporary ontologies share many structural similarities, regardless of the language in which they are expressed. As mentioned above, most ontologies describe individuals (instances), classes (concepts), attributes, and relations. In this section each of these components is discussed in turn. Common components of ontologies include:

- **Individuals:** instances or objects (the basic or "ground level" objects).
- **Classes:** sets, collections, concepts, classes in programming, types of objects, or kinds of things.
- **Attributes:** aspects, properties, features, characteristics, or parameters that objects (and classes) can have.
- **Relations:** ways in which classes and individuals can be related to one another.
- **Function terms:** complex structures formed from certain relations that can be used in place of an individual term in a statement.
- **Restrictions:** formally stated descriptions of what must be true in order for some assertion to be accepted as input.
- **Rules:** statements in the form of an if-then (antecedent-consequent) sentence that describe the logical inferences that can be drawn from an assertion in a particular form.
- **Axioms:** assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application. This definition differs from that of "axioms" in generative grammar and formal logic. In those disciplines, axioms include only

statements asserted as a priori knowledge. As used here, "axioms" also include the theory derived from axiomatic statements.

- **Events:** the changing of attributes or relations [4].

2.4 Uses of Ontologies

In this section.we review and elaborate the motivations for ontologies that we discussed above. In doing so,we characterise the space of uses for ontologies. The literature is currently rich with descriptions of ontologies and their intended purposes . At a high level , most seem to be intended for some manner of reuse .Some of these purposes are implicit in the various interpretations of the word 'ontology' that are commonly found in the literature ,as noted in [15] ; (e.g. a vocabulary for [9] vs a meta_level specification of , a logical theory [32,40])Other dimensions of variation include the nature of the software with which the ontology will be used , whether it is intended to be shared within a small group and reused within that context for a variety of applications ,or whether it is intended to be reused by a larger community . Some view their ontologies mainly as a means to structure a knowledge base ; others conceive an ontology to be used as part of a knowledge Base, e.g. by loading it in as a set of sentences which will be added to as appropriate; still others view their ontology as an application_specif ic inter-lingua (e.g. ATOS) [20,21] Another important motivation for ontologies is to integrate models of di_erent domains into a coherent framework.This arises in business process reengineering (where we need an integrated model of the enterprise and its processes ,its organizations , its goals,and its customers). in distributed multiagent architectures (where different agents need to communicate and solve problems).and in concurrent engineering and design.With these intuitions ,we sub-divide the space of uses for ontologies into the following three Categories:[5]

2.4.1Communication

Recall that ontologies reduce conceptual and terminological confusion by providing a unifying framework within an organization .In this way , ontologies enable shared understanding and communication between people with different needs and viewpoints arising from their

particular contexts . We will now consider in detail several aspects of the use of ontologies to facilitate communication among people within an organization[5] .

- **Normative Models**:

Within any large-scale integrated software system ,different people must have a shared understanding of the system and its objectives .By using an ontology ,we can construct normative model of the system . This creates a semantics for the system and an extendible model that can later be refined , and which allows semantic transformations between different contexts[5] .

- **Networks of Relationships**:

We can also use ontologies to create a network of relationships , keep track of what is linked , and explore and navigate through this network .Such a network is implicit within the system ,but people often have different perspectives and perhaps use different assumptions . Thus there is a lack of shared understanding concerning the nature of the key relationships within the system. This is particular important in applications which require the use of multiple ontologies from different domains. Ontologies serve to make all of these assumptions explicit by identifying the logical connections between elements across models of the system.

In general , we will also want the ontology to support the ability to reason about the impact of possible changes to the system. For example ,using an ontology to support enterprise modeling allows us to capture a picture of the enterprise that can b reworked .We can then answer questions about the enterprise model , such as what if scenarios related to changing different parts of the enterprise during reengineering[5].

- **Consistency and Lack of Ambiguity:**

One of the most important roles an ontology plays in communication is that it provides unambiguous definitions for terms used in a software system. Any set of software tools should be able to maintain consistency among themselves and the ontologies , though they need not be uniform. There may be the problem that a user's ontology is different from the ontology supporting the tool. In this case, we must provide an environment that can represent the different meanings for terms used by different people ("meaning mapper"). This also involves identifying the relevant assumptions used by different people, tools, or ontologies and the ability to capture multiple synonyms and utilise them in translation to various audiences[5].

- **Integrating Different User Perspectives:**

If we have a system with multiple communicating agents, this integration through shared understanding becomes vital. We face the challenge of integrating different perspectives while capturing key distinctions in a given perspective. For example, people in different positions in an organization will have different perspectives on what the organization does. What goals it achieves, and how these goals are achieved. There is also the problem of integrating global and local views of the system. By using an ontology to provide a normative model of the system, this integration can be achieved by assisting participants in communicating and coming to an agreement. This also lays the groundwork for the development of standards within a community. By adopting a shared ontology, all participants use a standardized terminology for all objects and relations in their domains[5].

2.4.2 Inter Operability:

Many applications of ontologies address the issue of inter-operability, in which we have different users that need to exchange data or who are using different software tools. A major theme for the use of ontologies in domains such as enterprise modeling and multi agent architectures is the creation of an integrating environment for different software tools. Toolkits for spot solutions exist, but there is often no consistency among these tools[5].

2.4.3 Ontologies as Inter-Lingua

Any information technology environment for business process reengineering or multiagent systems should use integrated enterprise models spanning activities, resources, organization , goals, products, and services. These integrated enterprise models serve as a common repository accessible by multiple tool sets. This can also serve to integrate existing data repositories, either by standardizing terminology among the different users of the repositories, or by providing the semantic foundations[5].

2.4.4 Dimensions of Inter-Operability

In addition to tools and repositories, there are several distinctions that can be made. First, we need to consider the nature of the relationships among the users who are sharing tools and data. It is vital that the ontologies and tools used by different agents or organizations within the same enterprise be sharable and reusable across these multiple organizations[5].

- **Internal Inter-Operability**

With internal inter-operability, all systems requiring inter-operation are under the direct control of some organizational unit. Differences exist for historical reasons and legacy systems which will no longer change, need to be integrated.

- **External InterOperability**

With external inter-operability, we have an organizational unit that wishes to insulate itself from changes imposed on it from the outside[5].

- **Integrated Ontologies Among Domains**

The other distinction for inter-operability arises from the issue of the integration of ontologies from different domains in order to support some task. For example, an ontology to support workflow management systems will need to integrate ontologies for processes, resources, products, services, and organization. The set of workflow tools would then use this set of integrated ontologies.

- **Integrating OntologiesAmong Tools**

On the other hand, we may also need to integrate different ontologies in the same domain because of legacy systems. For example, different tools may use different process ontologies; to achieve inter-operability, we need to have a common ontology that both sets of tools can use. This is the most difficult challenge facing the use of ontologies, since it is usually not possible to impose the requirement of integration on the tools themselves; rather we need to construct ontologies for tools that are already being used[5].

2.4.5 Systems Engineering

The applications of ontologies that we have considered to this point have focussed on the role that ontologies play in the operation of software systems. In this section we consider

applications of ontologies that support the design and development of the software systems themselves[5].

- **Specification:**

A shared understanding of the problem and the task at hand can assist in the specification of software systems. For example, the IBM Business System Development Method (BSDM)

develops and uses a ontology of the organization as the basis for IT design and development in that organization. The Common KADS Conceptual Modeling Language (CML) is used to build domain and task ontologies to assist specification of knowledge based systems. The role that ontologies play in specification varies with the degree of formality and automation within the system design methodology. In an informal approach, ontologies facilitate the process of identifying the requirements of the system and understanding the relationships among the components of the system. This is particularly important for systems involving distributed teams of designers working in different domains.In a formal approach, an ontology provides a declarative specification of a software system, which allows us to reason about what the system is designed for, rather than how the system supports this functionality[5].

- **Reliability:**

Informal ontologies can improve the reliability of software systems by serving as a basis for manual checking of the design against the specification. Using formal ontologies enables the use of [semi]automated consistency checking of the software system with respect to the declarative specificatio.In addition, formal ontologies can be used to make explicit the various assumptions made by different components of a software system, facilitating their integration. For example, in the the Integrated Development Support Environment (IDSE) semantic constraints and relationships between different tools must be maintained for successful tool integration. Axioms stating these constraints are interpreted and enforced semi-automatically, thus facilitating integration (see appendix B for further details). This is closely related to the use of declarative constraints to maintain semantic integrity in data bases Declaratively specified assumptions may explicitly restrict the applicability of a particular ontology to a problem domain. By proving that the ontology is capable of supporting various reasoning problems, we can demonstrate the reliability of the software system within the domain[5]

- **Reusability:**

To be effective, ontologies must also support reusability, so that we can import and export modules among different software systems.The problem is that when software tools are applied to new domains, they may not perform as expected, since they relied on assumptions that were satisfied in the original applications but not in the new ones. By characterizing classes of domains and tasks within these domains, ontologies provide a framework for determining which aspects of an ontology are reusable between different domains and tasks. Ontologies provide an "easy to reuse" library of class objects for modeling problems and domains. The ultimate goal of this approach is the construction of a library of ontologies which can be reused and adapted to different general classes of problems and environments. One such library is being constructed at the Knowledge Systems Laboratory using their online Ontology Server.

To be useful, these ontologies must be customizable, both to the class of problems and the class of users, whether they be managers, consultants, or engineers. Further, the ontologies in such a library must be extendible, allowing the incorporation of new classes of constraints and the specialisation of concepts and constraints for a particular problem.

One approach to extendibility is the notion of partially shared views [24] in the Process Inter-change Format Project.

There is a core PIF ontology which all translators operate with. In addition, there are different extensions of this core ontology which not all ontologies may share. In PIF, these extensions are captured by partially shared views, so that ontologies that have a partially shared view in common can translate without loss of expressiveness.

Similarly, in the KRSL Plan Ontology there is a set of modular specialized ontologies augment the general categories with sets of concepts and alternative theories of more detailed notions commonly used by planning systems, such as specific ontologies and theories of time points, temporal relations. and complex actions[5].

- **Closing remarks**:

Thus far, we have motivated the need for ontologies, clarified what they are and described a variety of circumstances in which they may be used. In the next few Sections, we turn our attention to the process of building and evaluating ontologies. First we describe some of the important steps in building an ontology[5].

2.5 Ontology Engineering

Ontology engineering (or ontology building) is a subfield of knowledge engineering that studies the methods and methodologies for building ontologies. It studies the ontology development process, the ontology life cycle, the methods and methodologies for building ontologies, and the tool suites and languages that support them.

Ontology engineering aims to make explicit the knowledge contained within software applications, and within enterprises and business procedures for a particular domain. Ontology engineering offers a direction towards solving the interoperability problems brought about by semantic obstacles, such as the obstacles related to the definitions of business terms and software classes. Ontology engineering is a set of tasks related to the development of ontologies for a particular domain. The main steps in the methodology are as follows (Figure 2.4)[6].

Step 1: Determine domain and scope

Development of an ontology starting by defining its domain and scope. That is, answer several basic questions:

- What is the domain that the ontology will cover?
- For what we are going to use the ontology?
- Who will use and maintain the ontology?

The answers of these questions may change during the ontology-design process, but at any given time they help limit the scope of the model [6].

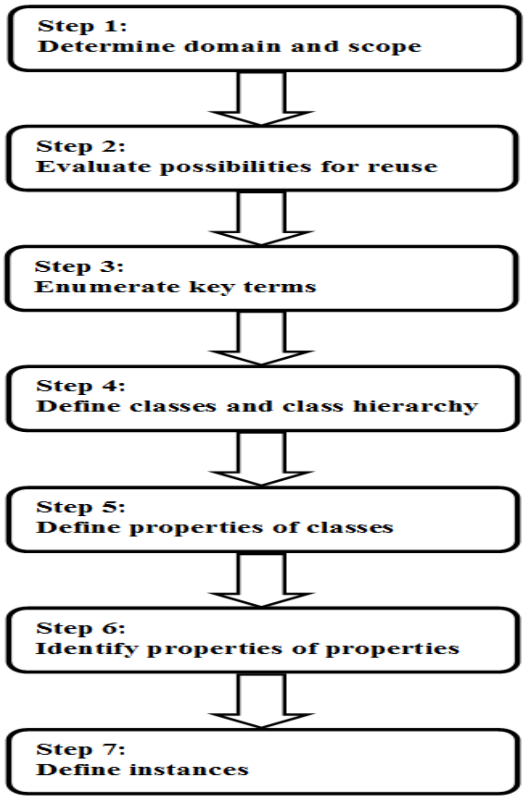

Step 1:
Determine domain and scope
Step 2:
Evaluate possibilities for reuse
Step 3:
Enumerate key terms
Step 4:
Define classes and class hierarchy
Step 5:
Define properties of classes
Step 6:
Identify properties of properties
Step 7:
Define instances

Figure 2.4: Ontology Development [6].

Step 2: Consider reuse

Reuse is extremely important in ontology development, for the simple reason that an ontology is most useful if everybody uses it: if we all define our own ontology for every application, then we are defeating the object of sharing meaning [6].

Step 3: Enumerate all the relevant terms

This step simply involves brainstorming all the terms associated with our domain literally, list all the domain-specific words and concepts that appear in the requirements [6].

Step 4: Define classes and class hierarchy

This step organize the understanding of the domain by identify classes and organize them.

Refining classes into hierarchies can be done either *top down* (identifying the most general classes, then next most general, and so on), or else *bottom up* (clustering similar terms into progressively more general classes) [6].

Step 5: Define properties

This step involves identifying, for each class, the properties that are associated with that class. These properties attached to the most general class that has them. Properties come in several different types:

- **Intrinsic properties** are those that relate to the nature of an object.
- **Extrinsic properties** are abstract properties such as ‘name’, which are attached to an object.
- **Component of an object** if the object is structured in some way, then it might be useful to identify its component parts.
- **Relationship** linking the objects with one another [6].

Step 6: Define properties of properties

This step is to identify the properties that each property has. Properties might be:

- Cardinality constraints.
- Type constraints.
- Range constraints.
- Domain constraints [6].

Step 7: Create instances

At this stage the ontology populate with instances of classes [6].

2.6 Previous Studies:

- **Jaime Cantais and David Dominguez(2002)**

This paper describes our experience in the rapid prototyping of a food
ontology oriented to the nutritional and health care domain that is used to share knowledge between the different stakeholders involved in the PIPS project.

The paper describes a Food ontology from the nutritional and health care viewpoint. This ontology is used to share knowledge between the different stakeholders involved in the PIPS project.

The paper presented the problem addressed with the design of the Food ontology, namely the provision of nutritional advice to diabetic patients.We described briefly the development process we used to design the ontology, and we described the main features of the Food ontology[6].

- **Michal Sindlar and Tom van der(2012)**

ORC is an ontology reasoning component that builds upon existing ontology modeling tools and techniques to support the integration and interpretation of multimodal medical information. We show how to embed ORC as a reasoning capability in reactive infrastructure agents that support intelligent agents operating in COMMODITY12, a personal health environment for diabetic patients and the medical professionals that treat them. The benefits of the approach are illustrated by showing how medical information for patient profiles at different sources can be included in COMMODITY12, thus extending the generality and potential of the resulting system. The approach also illustrates how ontologies can be combined with a variety of artificial intelligence tools and techniques to support e-Health activities on the Internet, thus contributing towards the vision of NetMedicine[7]

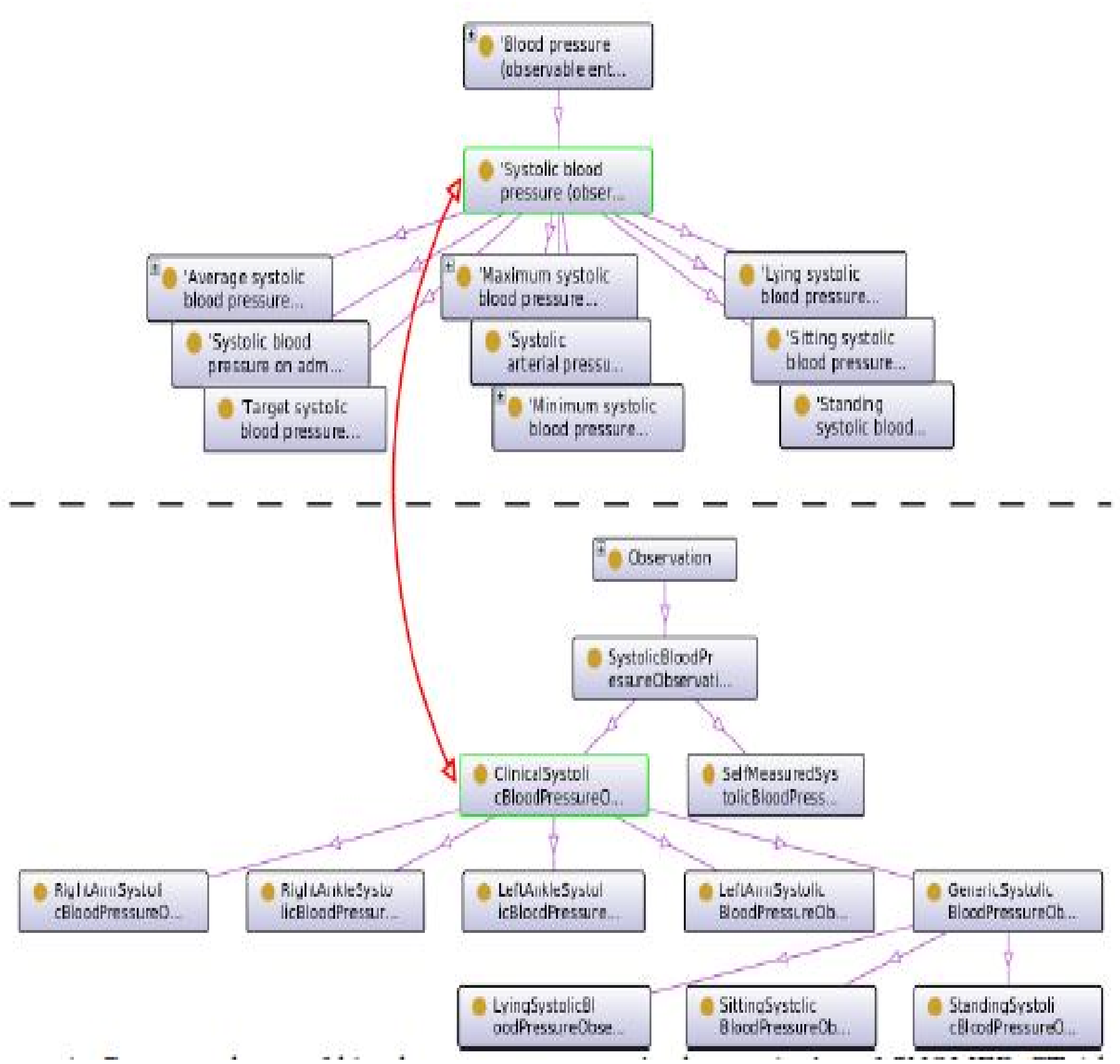
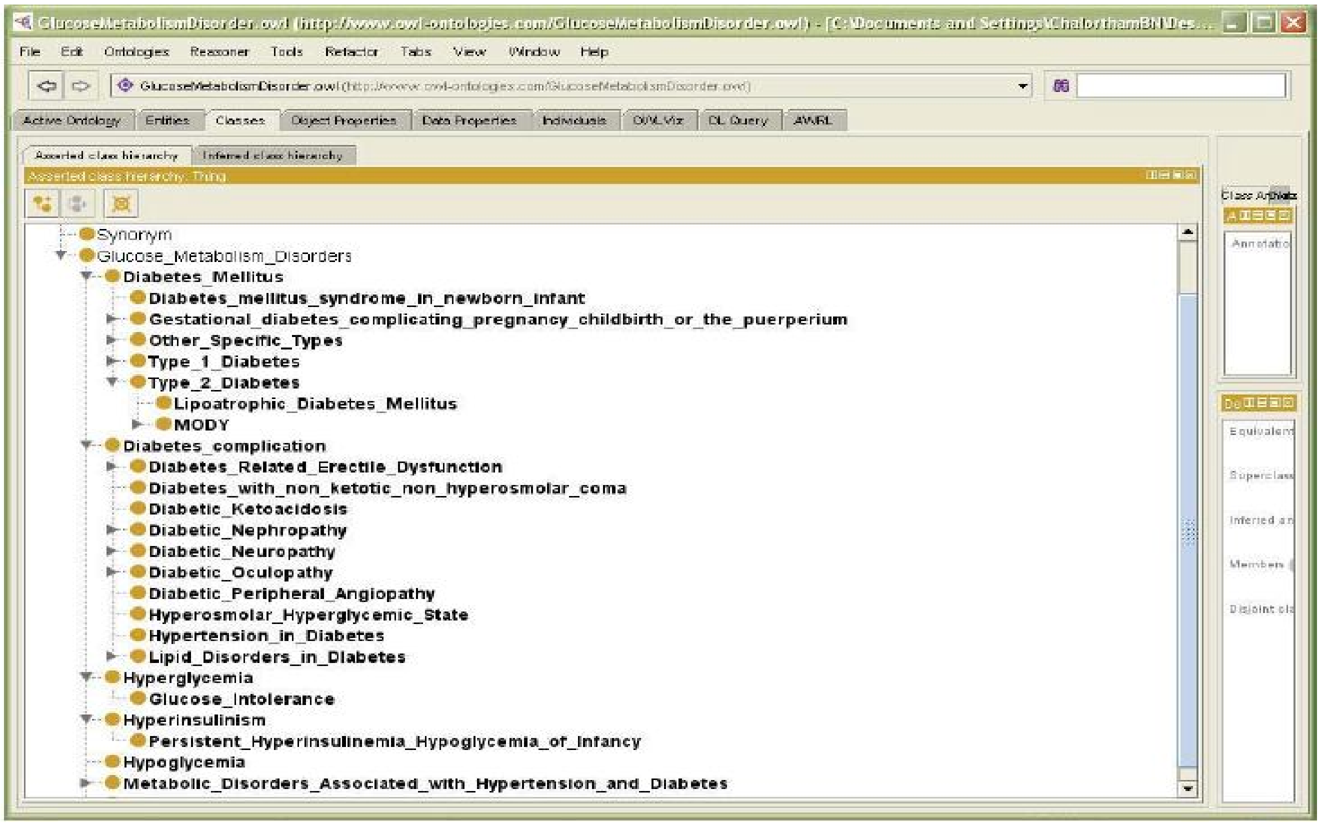

'Blood pressure (observable ent...
'Systolic blood pressure (obser...
'Average systolic blood pressure...
'Maximum systolic blood pressure...
'Lying systolic blood pressure...
'Systolic blood pressure on adm...
'Systolic arterial pressu...
'Sitting systolic blood pressure...
Target systolic blood pressure...
'Minimum systolic blood pressure...
'Standing systolic blood...
Observation
SystolicBloodPressureObservati...
ClinicalSystolicBloodPressureO...
SelfMeasuredSystolicBloodPress...
RightArmSystolicBloodPressureO...
RightAnkleSystolicBloodPressur...
LeftAnkleSystolicBloodPressure...
LeftArmSystolicBloodPressureOb...
GenericSystolicBloodPressureOb...
LyingSystolicBloodPressureObse...
SittingSystolicBloodPressureOb...
StandingSystolicBloodPressureO...

Fig. 2.5 Correspondence of blood pressure concepts in the ontologies of SNOMED CT (above dashed line) and Portavita (below dashed line)[7].

- **Nopphadol Chalortham and Marut Buranarach(2009)**

  Diabetes mellitus becomes a serious problem in most countries. A continuous plan and management for DM patients to control the blood glucose level and to monitor progress of DM's complications becomes an important issue. They proposed an ontology-based development for clinical information system which enables health providers who are non-experts in Diabetes disease can suggest patients the essential activities for improving life quality and achieving goals of DM treatment. In this paper, focus on the ontology development process for Type II DM. There are three main steps, 1) domain and scope setting, 2) important terms acquisition, classes and class hierarchy conceptualization and 3) instances instantiation. They show an example of using reminding system which it developed based on our ontology[8].


GlucoseMetabolismDisorder.owl (http://www.owl-ontologies.com/GlucoseMetabolismDisorder.owl) - [C:\Documents and Settings\ChalorthamBN\Des...
File Edit Ontologies Reasoner Tools Refactor Tabs View Window Help
GlucoseMetabolismDisorder.owl (http://www.owl-ontologies.com/GlucoseMetabolismDisorder.owl)
Active Ontology Entities Classes Object Properties Data Properties Individuals OWLViz DL Query AWRL
Asserted class hierarchy Inferred class hierarchy
Asserted class hierarchy: Thing
Class Annotations
Annotations
Equivalent
Superclass
Inferred an
Members
Disjoint cl

```
Synonym
Glucose_Metabolism_Disorders
  Diabetes_Mellitus
    Diabetes_mellitus_syndrome_in_newborn_infant
    Gestational_diabetes_complicating_pregnancy_childbirth_or_the_puerperium
    Other_Specific_Types
    Type_1_Diabetes
    Type_2_Diabetes
      Lipoatrophic_Diabetes_Mellitus
      MODY
  Diabetes_complication
    Diabetes_Related_Erectile_Dysfunction
    Diabetes_with_non_ketotic_non_hyperosmolar_coma
    Diabetic_Ketoacidosis
    Diabetic_Nephropathy
    Diabetic_Neuropathy
    Diabetic_Oculopathy
    Diabetic_Peripheral_Angiopathy
    Hyperosmolar_Hyperglycemic_State
    Hypertension_in_Diabetes
    Lipid_Disorders_in_Diabetes
  Hyperglycemia
    Glucose_Intolerance
  Hyperinsulinism
    Persistent_Hyperinsulinemia_Hypoglycemia_of_Infancy
  Hypoglycemia
  Metabolic_Disorders_Associated_with_Hypertension_and_Diabetes
```

Figure 2.6: Ontology of Glucose Metabolism Disorder (OGMD)[8].

- Marut Buranarach and Nopphadol Chalortham **(2009)**

Improving quality of healthcare for people with chronic conditions requires informed and knowledgeable healthcare providers and patients. Decision support and clinical information system are two of the main components to support improving chronic care. As they describe an ontology based information and knowledge management framework that is important for chronic disease care management. Ontology based knowledge acquisition and modeling based on knowledge engineering approach provides an effective mechanism in capturing expert opinion in form of clinical practice guidelines. The framework focuses on building of healthcare ontology and clinical reminder system that link clinical guideline knowledge with patient registries to support evidenced-based healthcare. They discuss approaches in integrating clinical reminder services to existing healthcare provider environment by focusing on augmenting decision making and improving quality of patient care services[9].

- **Lutes and Baggili(2006)**

This paper discussed Diabetic e- Management System (DEMS) handle the computers software which help diabetes patients self manage their condition About the methodology uses a variety of technologies and is an all encompassing mobile self management diabetic software system in the areas of insulin medications, non-insulin medications, diet, exercise, blood sugar and weight entries. The DEMS prototype system created seemed to be usable based on the SUS scores and the interviews that we conducted. About the strength of this paper generally the system was usable and easy to use. Finally the weakness of this paper:

(a) Difficult to get old people to use the   system.

(b) It requires strict compliance rates;

(c) Inability to gather data from a glucometer [9].

- **Turner et al (2006)**

A telehealth system encompasses healthcare practices, which are supported by innovative telecommunication strategies to offer additional support to treatment for people with type 2 diabetes. The software design follows an integrated approach with clinical care and the experiences of clinical staff and patients. The intervention included a Bluetooth enabled glucose meter linked to a mobile phone, an integrated diary to record insulin dose, and feedback on the phone's colour screen of blood glucose data. About the strength of this Paper:

(a) A pilot study was conducted to explore and identify the training and support requirements of patients and clinicians using the system.

(b) The technology could improve support for type 2 diabetes patients commencing insulin treatment. Finally the weakness of this paper Having potential to improve patient care. Weakness – Not many users (patients 7 nurses) are familiar with the technology[10].

- **Valls et al (2010)**

Ontological engineering process can significantly improve the management of complex distributed health systems, ie. senior Home Care assistance. The proposed ontology software design is based on a Home Care medical model framed in the scope of the K4Care European project (FP6). Knowledge engineers and medical experts worked together in order to ensure a complete transfer of the organizational medical knowledge to the formal ontology, as well as the correctness of the representation of that information. Finally the weakness of this paper Ontological paradigm and the expressiveness of modern ontology languages support the creation of profile-based interaction models in a transparent and seamless way, and increases the reusability and generality of the developed software components.

Weakness: the model is complex and biased towards the management of complex health systems[11].

- **Acampora and Lee(2009)**

This study presents a healthcare application based on agent systems, Fuzzy Markup Language (FML) and fuzzy ontology for modeling a diabetes semantic decision-making and The system is premised on the joint use of OWL fuzzy diabetes ontology and FML description represents the most suitable way to model medical concepts and fuzzy relationships using the taxonomical knowledge. The performance of the proposed agent is evaluated according to the difference between the medical staff and the proposed agent About the strength of this Paper The proposed method is feasible for diabetes semantic decision-making Finally the weakness of this paper A very complex three-layered fuzzy ontology model[12].

Paper No	Title	Authors	Work Done	The Similarities	The Differences
1	An example of food ontology for diabetes control	Jaime Cantais, David Dominguez, Valeria Gigante,Loredana Laera, and Valentina Tamma	Description of food for diabetics.	-	Focused on the description the foods that help diabetes control
2	Ontology Reasoning Component for Diabetes	Özgür Kafalı, Michal Sindlar, Tom van der Weide and Kostas Stathis	Explain reasoning and component for diabetes in general.	Explain reasoning and symptoms.	Focused on the description Reasoning and Component for Diabetes
3	Ontology Development for Type II Diabetes Mellitus Clinical Support System	Nopphado Chalortham, Marut Buranarach and Thepchai Supnithi	Focused on the description tupe 2 diabetes mellitus clinical support system.	Description the symptoms of type2.	Explain type 2 diabetes just without other types.
4	Diabetic e-Management System (DEMS)	Lutes and Baggili	Diabetic e-Management System (DEMS) handle the computers software which help diabetes patients self manage their condition.	-	Focused on technologies and is an all encompassing mobile self-management diabetic software system in the areas of insulin medications, non-insulin medications, diet, exercise, blood sugar and weight entries
5	A Telehealth System to Optimise Insulin Titration in Primary Care	Turner et al	A telehealth system encompasses healthcare practices, which are supported by innovative telecommunication strategies to offer additional support to treatment for people with type 2 diabetes.	support to treatment for people with type 2 diabetes	focused on A telehealth system encompasses healthcare practices, which are supported by innovative telecommunication strategies to offer additional support to treatment for people with type 2 diabetes
6	Using ontologies for structuring organizational knowledge in Home Care assistance	Valls et al	Information Technologies and Knowledge-based Systems can significantly improve the management of complex distributed health systems, where supporting multi disciplinarily is crucial and communication and synchronization	-	The ontological engineering process for building the Actor Profile Ontology (APO) is performed under the On-To-Knowledge methodological approach which is based on a 5 step process.

Figure 2.7: Related Work Summary

Chapter Three

Ontology Tools

This chapter introduces protégé-OWL in more details, protégé and its platforms for modeling ontology's. Also OWL Ontologies and its components.

3.1 Protégé

Protégé is a free, open source ontology editor and a knowledge acquisition system. Protégé provides a graphic user interface to define ontologies. It also includes deductive classifiers to validate that models are consistent and to infer new information based on the analysis of an ontology. Like Eclipse, Protégé is a framework for which various other projects suggest plugins. This application is written in Java and heavily uses Swing to create the rather complex user interface. Protégé recently has over 200,000 registered users. According to a 2009 book it is "the leading ontological engineering tool"[13].

Protégé is being developed at Stanford University in collaboration with the University of Manchester and is made available under the Mozilla Public License 1.1.

The Protégé platform supports two main ways of modeling ontologies:

- **The Protégé-Frames editor** enables users to build and populate ontologies that are frame-based, in accordance with the Open Knowledge Base Connectivity protocol (OKBC). In this model, an ontology consists of a set of classes organized in a subsumption hierarchy to represent a domain's salient concepts, a set of slots associated to classes to describe their properties and relationships, and a set of instances of those classes - individual exemplars of the concepts that hold specific values for their properties [13].

- **The Protégé-OWL editor** enables users to build ontologies for the Semantic Web, in particular in the W3C's Web Ontology Language (OWL). "An OWL ontology may include descriptions of classes, properties and their instances. Given such an ontology, the OWL formal semantics specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics. [4].

The Protégé-OWL Editor enables users to:

- Load and save OWL and RDF ontologies.
- Edit and visualize classes and properties.
- Define logical class characteristics as OWL expressions.
- Execute reasoners such as description logic classifiers.

3.2 OWL Ontologies

Ontologies are used to capture knowledge about some domain of interest. An ontology describes the concepts in the domain and also the relationships that hold between those concepts. Different ontology languages provide different facilities. The most recent development in standard ontology languages is OWL from the World Wide Web Consortium (W3C)1. Like Protégé, OWL makes it possible to describe concepts but it also provides new facilities. It has a richer set of operators - e.g. intersection, union and negation. It is based on a different logical model which makes it possible for concepts to be defined as well as described. Complex concepts can therefore be built up in definitions out of simpler concepts[4].

Furthermore, the logical model allows the use of a reasoner which can check whether or not all of the statements and definitions in the ontology are mutually consistent and can also recognise which concepts fit under which definitions. The reasoner can therefore help to maintain the hierarchy correctly. This is particularly useful when dealing with cases where classes can have more than one parent[4].

3.3 Components of OWL Ontologies

OWL ontologies have similar components to Protégé frame based ontologies. However, the terminology used to describe these components is slightly different from that used in Protégé. OWL ontology consists of Individuals, Properties, and Classes, which roughly correspond to Protégé frames Instances, Slots and Classes.

Individuals: represent objects in the domain. Individuals are also known as instances (Figure 3.1). Individuals can be referred to as being 'instances of classes'[4].

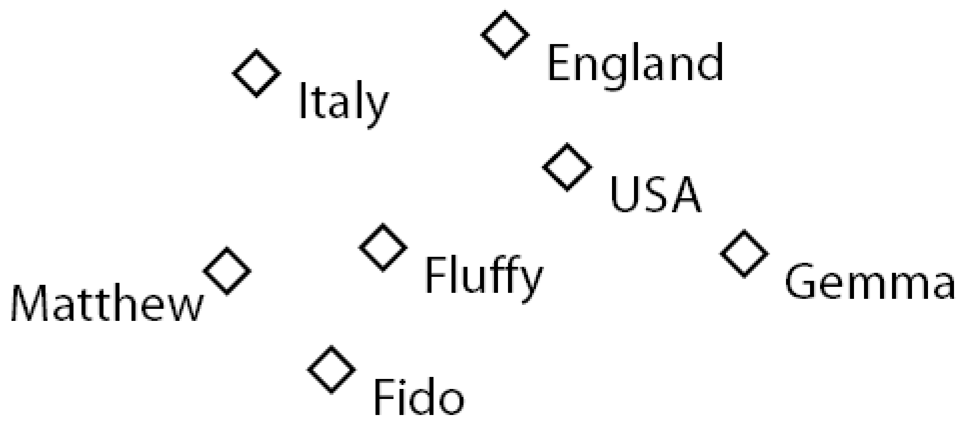


Figure 3.1: Representation Of Individuals [4].

Properties: are binary relations on individuals that link two individuals together (Figure 3.2). Properties are roughly equivalent to slots in Protégé. They are also known as roles in description logics and relations in UML and other object oriented notions. In GRAIL and some other formalism they are called attributes [4].

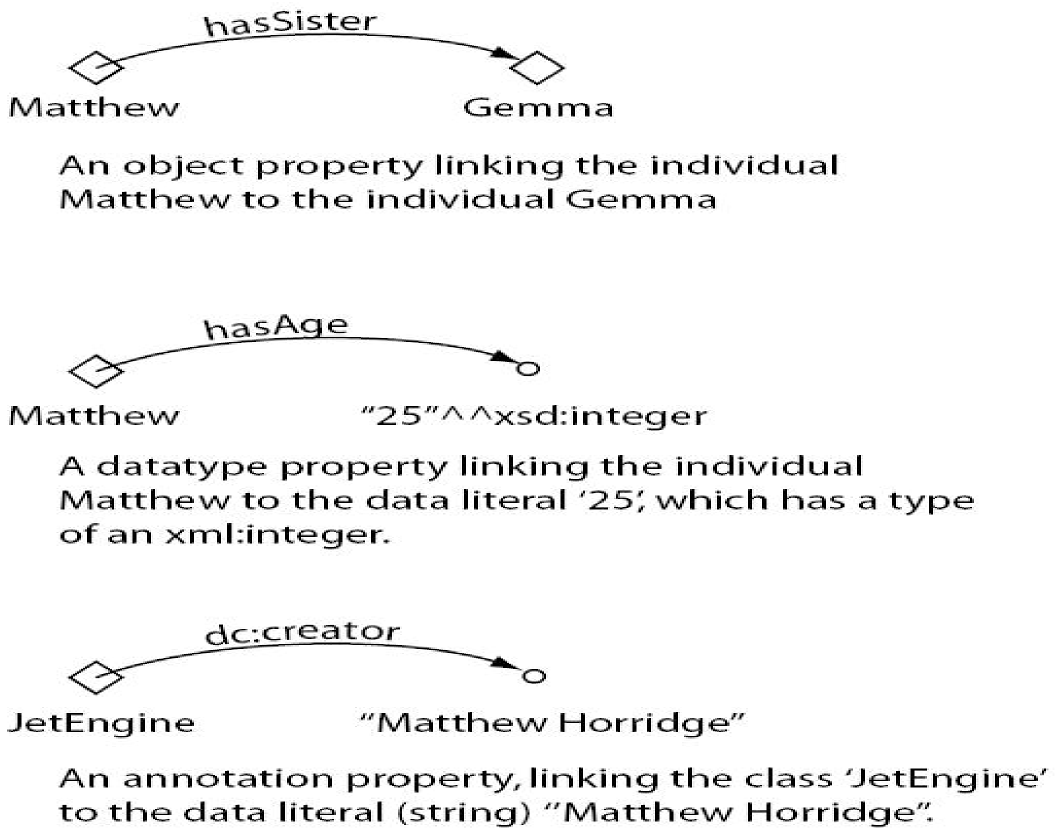

hasSister
Matthew
Gemma
An object property linking the individual Matthew to the individual Gemma
hasAge
Matthew
"25"^^xsd:integer
A datatype property linking the individual Matthew to the data literal '25', which has a type of an xml:integer.
dc:creator
JetEngine
"Matthew Horridge"
An annotation property, linking the class 'JetEngine' to the data literal (string) "Matthew Horridge".

Figure 3.2: Representation Of Properties [4].

OWL classes: are interpreted as sets that contain individuals (Figure 3.1). They are described using formal (mathematical) descriptions that state precisely the requirements for membership of the class. The word concept is sometimes used in place of class. Classes are a concrete representation of concepts [4].

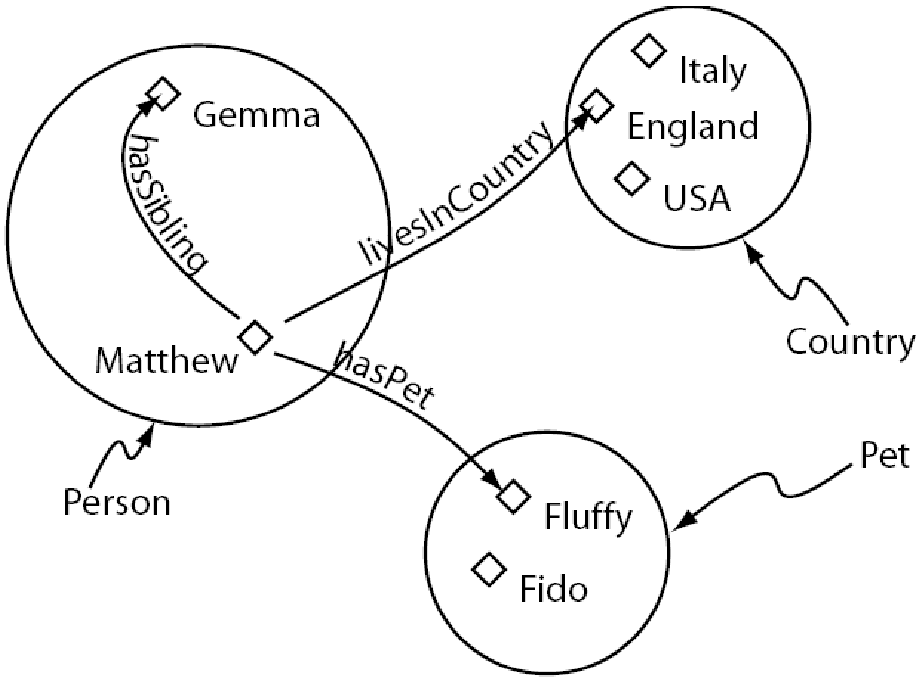


Figure 3.3: Representation Of Classes (Containing Individuals) [4].

Chapter Four

Ontology Building

In chapter two Fig ure 2.4 to develop an ontology there are several steps to be follow. In this chapter these steps will be illustrated in more details with respect to diabetes domain.

4.1 Proposed Ontology

The term ontology, in our context, can be best defined as a formal explicit description of concepts or entities, and their properties, relationships and constraints (Gruniger & Fox, 1995; Noy & McGuiness, 2001). The uses of ontology to support the multi-agent system (MAS) tool, development of ontology is critical in the following ways:

- ✓ Agents use ontology to share common terms and to communicate to other agents (Wooldridge, 2002).
- ✓ Agent must understand the environment in which they operate. When agents retrieves or store the knowledge, it needs to know how the knowledge is structured. These semantics is the ontology of the knowledge (Yusko, 2005)[4].

There are various methods available to design ontology. For example Gruniger and Fox methodology (1995), REFSENO (Representation Formalism for Software Engineering Ontologies) methodology (Tautz and Wangenheim, 1998), CommonKads and Protégé (Noy & McGuiness, 2001). In this study, Protégé OWL methodology shall be used. Protégé is chosen due to the following considerations:

- Protégé is a much easier editing tool to learn and more suitable for earlier stage of ontology building (Ribiere & Charlton, 2007).

- The user interface could be optimized to allow non-experts to input and amend ontology in the knowledge base
- Supports the OWL/DL format, which is intended to be used in the to-be-developed MAS.
- Protégé is free, available for download under the Mozilla open-source license, and has been used for thousands of projects ranging from modeling cancer-protocol guidelines to modeling nuclear power stations. Ontology models are available in the Protégé user forum for others to view[4].

For diabetes self management system process ontology, the Ruiz ontology shall be used as the basis, due to similarity of the concepts in author's diabetes self management system environment. Hence, it shall be easier to evaluate the proposed ontology later[4].

During development and combination of the diabetes self management system ontology, we discovered that certain steps are better performed in iterations, as follows:

1. Define the scope and purpose of ontology

2. Consider reusing existing ontologies

3. Enumerate important terms in the ontology

4. Define the individual examples

5. Define the properties of classes—slots

6. Define the classes and the class hierarchy for individual groups

7. Define the facets of the slots

8. Create instances[6].

4.2 Developing Diabetes Ontology

4.2.1 Determine the domain and scope of the ontology

One of the ways to determine the scope of the ontology is to sketch a list of questions that a knowledge base based on the ontology should be able to answer, competency questions. These questions will serve as the litmus test later: Does the ontology contain enough information to answer these types of questions? Do the answers require a particular level of detail or representation of a particular area? These competency questions are just a sketch and do not need to be exhaustive [6].

In the Diabetes domain, the following are the possible competency questions:

- What are types of diabetes?
- What are the symptoms of diabetes?
- What is the treatment of diabetes?
- What are the reasons for diabetes?
- What are good foods for diabetes?

Judging from this list of questions, the ontology will include the information about diabetes.

4.2.2 Consider reusing existing ontologies

Reusing existing ontologies may be a requirement if a system needs to interact with other applications that have already committed to particular ontologies or controlled vocabularies [6]. In this research, the ontology will be built from scratch.

4.2.3 Enumerate important terms in the ontology

It is useful to write down a list of all terms we would like either to make statements about or to explain to a user. What are the terms we would like to talk about? What properties do those terms have? What would we like to say about those terms? For example, important Diabetes-related terms will include:

Diabetes site, Diabetes type, Pre_Diabetes Diabetes Type1, Diabetes Type11, Pregnance_Sugar, Foods, Grain Products, Milk Products, Meat, Fruits, Special Products, Sugar Products, Diabetes Protection, Diabetes Reasons, Symptoms , Treatment, Blood Pressure, Family History, Fever, Level Cholestrol, Tirst, Urine, Weight loss, Active less, and so on.

Initially, it is important to get a comprehensive list of terms without worrying properties that the concepts may have, or whether the concepts are classes or slots.

The next two steps, developing the class hierarchy and defining properties of concepts (slots) are closely intertwined. It is hard to do one of them first and then do the other. These two steps are also the most important steps in the ontology design process.

4.2.4 Define the Classes and the Class Hierarchy

There are several possible approaches in developing a class hierarchy:

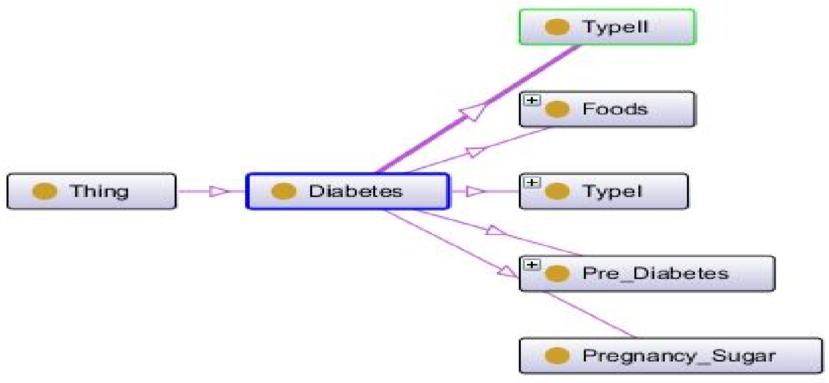


Figure 4.1: Class Thing

• **Top-down development process:** starts with the definition of the most general Concepts in the domain and subsequent specialization of the concepts.

For example, I can start with subclass Diabetes, Then specialize it by creating its subclass: Pre_Diabetes, Type1, Type11, Foods and Pregnancy_sugar (Figure 4.1).

An OWL XML code can be as follow:

```
<Declaration>
    <Class IRI="#Pre_Diabetes"/>
</Declaration>
<Declaration>
    <Class IRI="#Type1"/>
</Declaration>
<Declaration>
    <Class IRI="#Type11"/>
</Declaration>
    <Class IRI="#Foods "/>
</Declaration>
```

Categorize the Type1 to *Treatment*, *Symptoms* and *Cases* **(Figure 4.2)**.

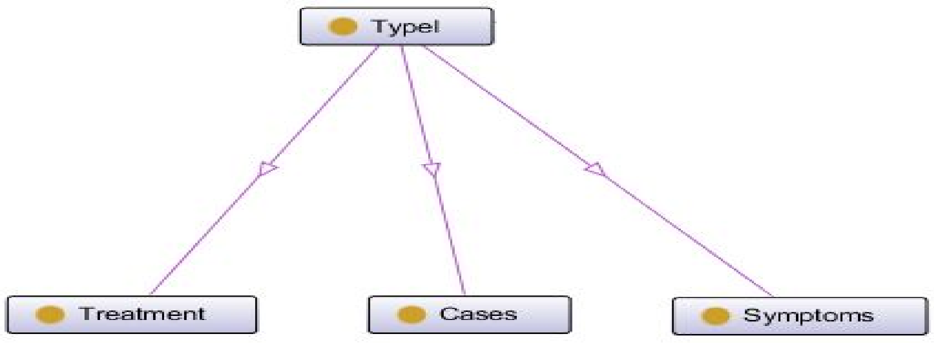


Figure 4.2: Type1 Classes

• **Bottom-up development process:** starts with the definition of the most specific classes, the leaves of the hierarchy, with subsequent grouping of these classes into more general concepts.

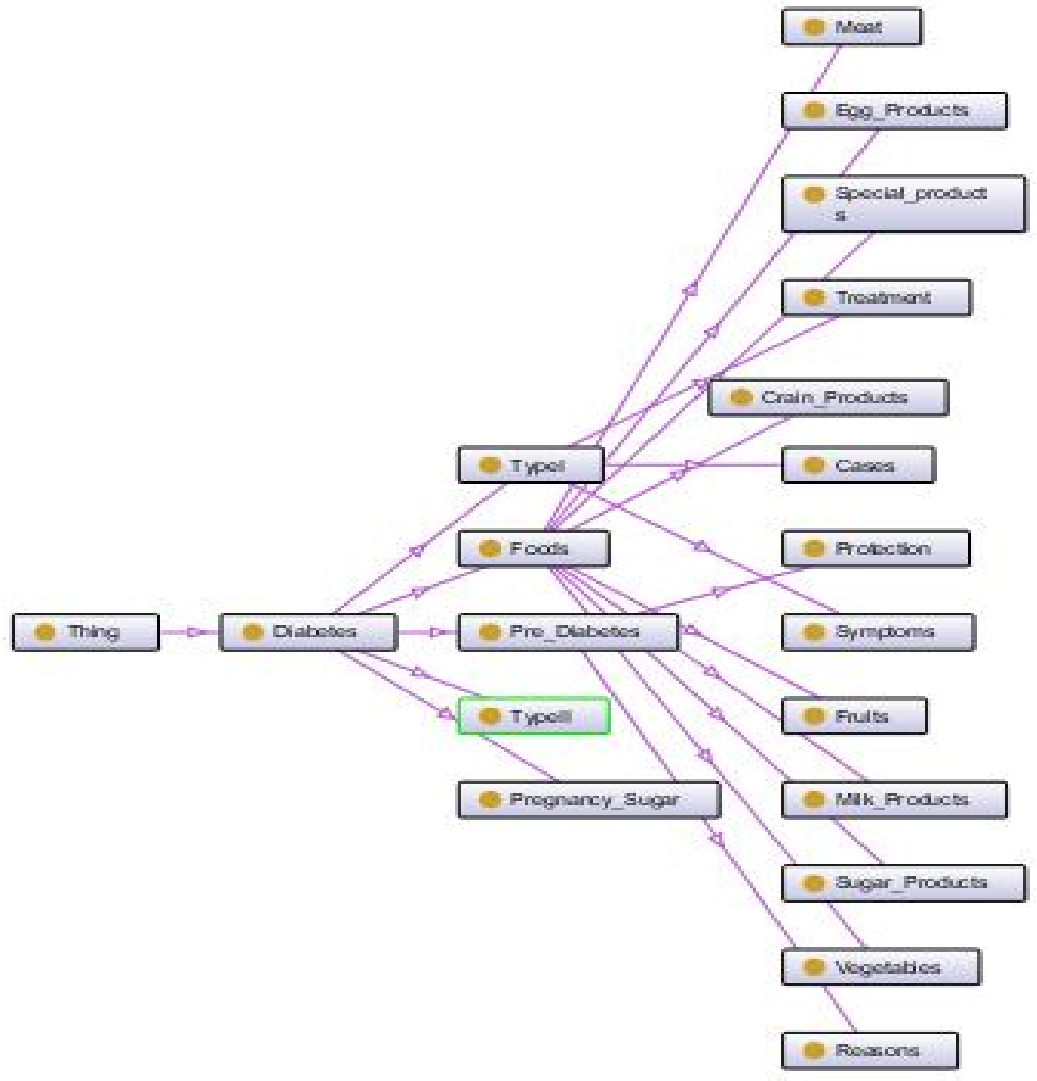


Figure 4.3: Diabetes Classes Hierarchy

4.2.5 Define the properties of classes

The classes alone will not provide enough information to answer the competency questions from Figure 2.6 Step 1. Once we have defined some of the classes, we must describe the internal structure of concepts.

We have already selected classes from the list of terms we created in Figure 2.6 Step 3. Most of the remaining terms are likely to be properties of these classes. These terms include**:**

- ***has Blood pressure.***
- ***hasFever.***
- ***hasHunger.***
- ***hasTrist.***
- ***hasUrine .***
- ***hasFamilyHistory.***
- ***hasLevelCholestrol.***

For each property in the list, we must determine which class it describes. These properties become slots attached to classes.

4.2.6 Define properties of properties

This step is to identify the properties that each property has. Properties might be:

- **Cardinality constraints:** are used to talk about the number of relationships that an individual may participate in for a given property. Cardinality restrictions come in three flavors: Minimum cardinality restrictions, Maximum cardinality restrictions and exact cardinality restrictions.

Cardinality restrictions are conceptually easier to understand than **quantifier restrictions** which it consists of three parts:

1. A quantifier, which is either the existential quantifier (some) or the universal quantifier (only).
2. A property, along which the restriction acts.
3. A filler that is a class description.)

- **Existential restrictions** describe classes of individuals that participate in at least one relationship along a specified property to individuals that are members of a specified class.
- **Universal restrictions** describe classes of individuals that for a given property only have relationships along this property to individuals that are members of a specified class.
- **Data Type constraints:** Data type properties describe relationships between individuals and data values.

Range constraints and Domain constraints

Properties may have a domain and a range specified. Properties link individuals from the domain to individuals from the range.

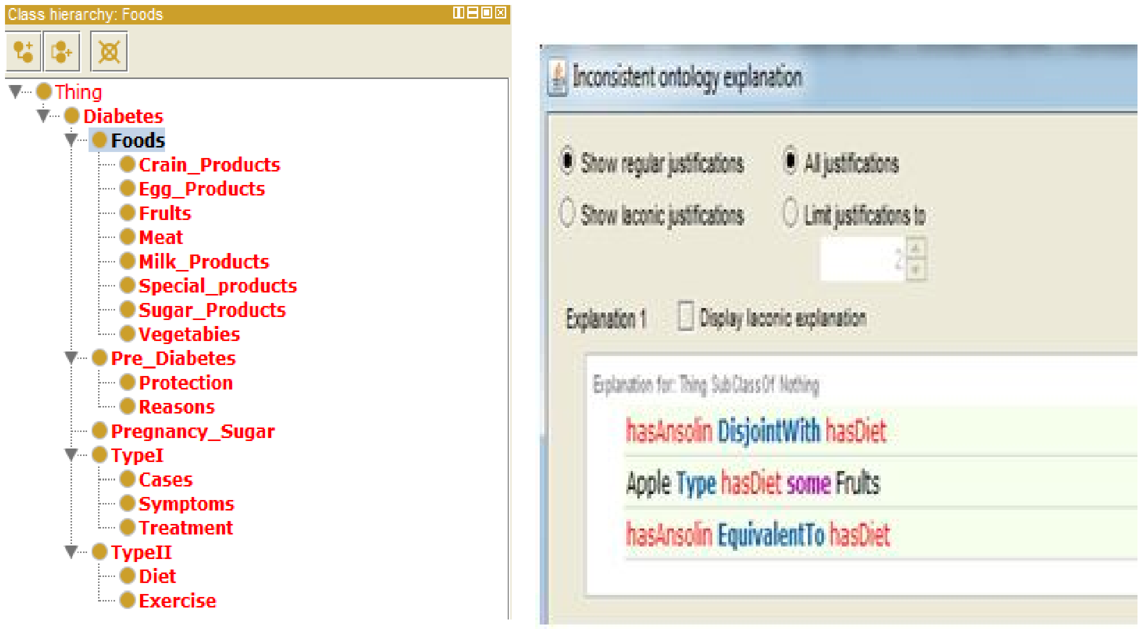
It is important to realize that in OWL domains and ranges should not be viewed as constraints to be checked. They are used as `axioms' in reasoning.

4.2.7 Create instances

At this stage the ontology populate with instances of classes. For example, class type1 may have instances such as Hozifa[5].

4.3 Invoke the Reasoner

After the Reasoner has been invoked - via the ‘start Reasoner’ button in the Reasoner drop down menu- to automatically compute the classification hierarchy, and also to check the logical consistency of the ontology ,inconsistent classes appeared and Reasoner stopped working.



Class hierarchy: Foods

```
Thing
  Diabetes
    Foods
      Crain_Products
      Egg_Products
      Fruits
      Meat
      Milk_Products
      Special_products
      Sugar_Products
      Vegetabies
    Pre_Diabetes
      Protection
      Reasons
    Pregnancy_Sugar
    TypeI
      Cases
      Symptoms
      Treatment
    TypeII
      Diet
      Exercise
```

Inconsistent ontology explanation

● Show regular justifications ○ Show laconic justifications

● All justifications ○ Limit justifications to 2

Explanation 1 ☐ Display laconic explanation

Explanation for: Thing SubClassOf Nothing

hasAnsolin DisjointWith hasDiet

Apple Type hasDiet some Fruits

hasAnsolin EquivalentTo hasDiet

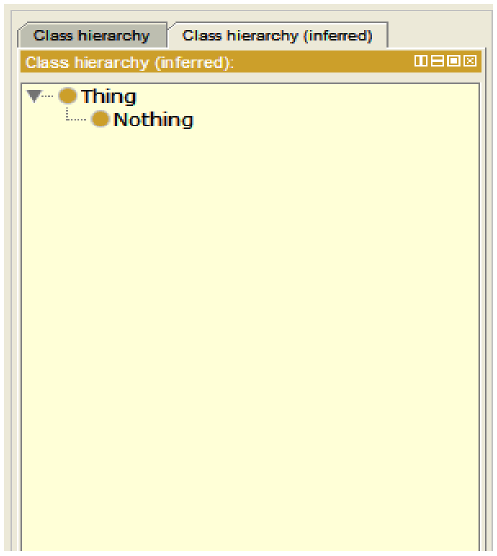
Figure 4.4: Inconsistent Classes and Ontology Explanation

After finished designing the ontology and ensure that all classes are consistent and well defined, then can make queries using SPARQL. For example,

SELECT? Subject? Object

WHERE {?subject ?rdfs: subclassof ?object}

And the result shown in (figure 4.7).



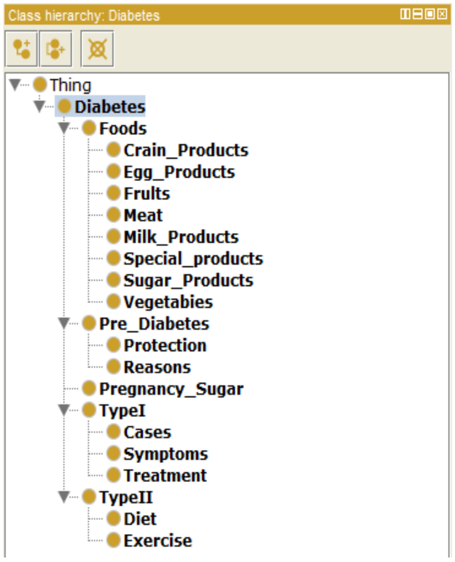
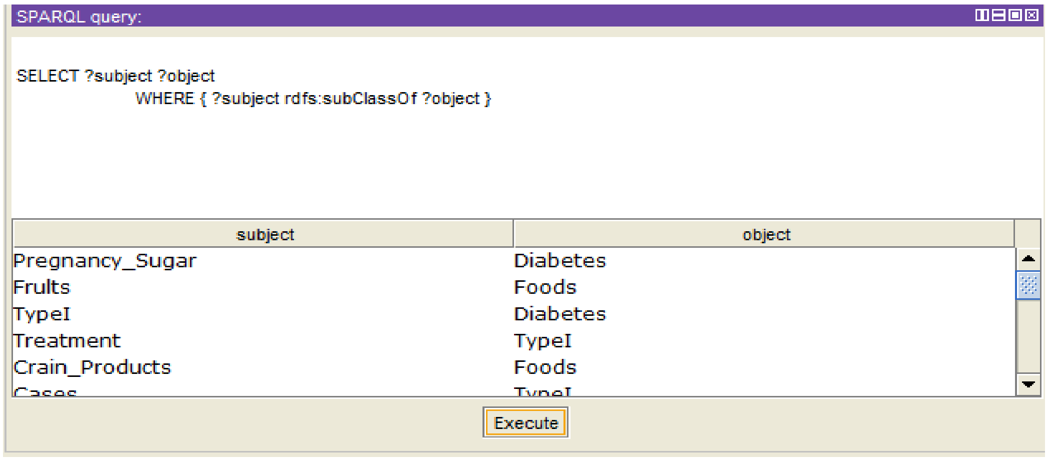


Figure 4.5: Inferred Class Hierarchy before Invoking the Reasoner

Figure 4.6: Inferred Class Hierarchy After Invoking the Reasoner



```
SELECT ?subject ?object
        WHERE { ?subject rdfs:subClassOf ?object }
```

subject	object
Pregnancy_Sugar	Diabetes
Frults	Foods
TypeI	Diabetes
Treatment	TypeI
Crain_Products	Foods
Cases	TypeI

Figure 4.7: a query using SPARQL

Chapter Five
Results

5.1 Results

After building the ontology has been finished, it sent to the reasoner to check class consistency and to compute subsumption relationships. The reasoner generates inferred class hierarchy and the researcher ensures that all the classes are consistent with its own definitions.

As result, have explicit representation and full definitions for Diabetes objects, properties and their relationships. And also can make any query to retrieve information. For example, we can make a query to know What are the main type of diabetes can or what are the Symptoms of type1 and so on.

5.2 Conclusion

In this research Protégé Framework and OWL used to design the ontology for diabetes, enumerated the terms that can possibly found in the domain, then organized them in a hierarchy depending on which class subsumes another, and defined properties for each class. After that the steps went deeply to define constraints on the properties such as: cardinality constraints, domain and range constraints. Final step in designing the ontology was defining some instances to be able to make queries. After that, the Reasoner used to check the consistency of the classes and to generate inferred class hierarchy.

5.3Future Work

1.Publish the Diabetes ontology to achieve the goal of Reusability.

2.Find an away to represent the ontology events using ontology languages (Perdurant ontology).

3. Do more ontology researches on Diabetes Domain.

4.integrate between this Ontology and Systems of human health

References

1. http://en.wikipedia.org 2:10pm 15/5/2014
2. Asunción Gómez-Pérez , ONTOLOGICAL ENGINEERING, Facultad de Informática Universidad Politécnica de Madrid.
3. Steffen Staab and Rudi Studer, Handbook of ontology's, second edition Institute AIFB Germany.
4. Matthew Horridge " A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools" , Edition 1.3.
5. Mike Uschold and Michael Gruninger, Ontologies Principles Methods and Applications, The University of Edinburgh
6. Jaime Cantais, David Dominguez, Valeria Gigante,Loredana Laera, and Valentina Tamma, An example of food ontology for diabetes control, University of Liverpool.
7. Özgür Kafalı, Michal Sindlar, Tom van der Weide and Kostas Stathis, ORC: an Ontology Reasoning Component for Diabetes, University of London.
8. Nopphado Chalortham, Marut Buranarach and Thepchai Supnithi, Ontology Development for Type II Diabetes Mellitus Clinical Support System, Silpakorn UniversityAmphoe Mueang, Nakorn Pathom, Thailand 73000.
9. Marut Buranarach, Nopphadol Chalortham, Piyatat Chatvorawit, Ye Myat Thein and Thepchai Supnithi, An Ontology-based Framework for Development of Clinical Reminder System to Support Chronic Disease Healthcare, Ministry of Public Health.

10. Lutes and Baggili,(2006), Diabetic e-Management System (DEMS), IEEE .
11. Turner et al.,(2006), A Telehealth System to Optimise Insulin Titration in Primary Care, National Institute for health Research.
12. Valls et al., (2010), Using ontologies for structuring organizational knowledge in Home Care assistance, international journal of medical informatics x x x.
13. Acampora, Lee & Wang (2009), FML-based Ontological Agent for Healthcare Application with Diabetes, IEEE.