ONTOLOGY-BASED CLINICAL DECISION SUPPORT SYSTEM APPLIED ON DIABETES

Alphonsine Mukabunani

Supervisors

Jan Pettersen Nytun

Ghislain Maurice Norbert Isabwe

This Thesis is submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Information and Communication

University of Agder, 2017

Faculty of Engineering and Science

Department of Information and Communication Technology

Abstract

Medical diagnosis is a multi-step process which is complex as it requires the consideration of many factors. Additionally, the accuracy of diagnosis varies depending on the skill and knowledge a physician has in the medical field. Using ICT solution, the physicians can be assisted so that they can make an accurate decision. Many applications have been developed to enhance physician performance and improve the patient outcome, however, the quality of these applications varies depending on knowledge representation methodology and reasoning approach adopted. Nowadays, ontology is being used in many clinical decision support as it formally represents the concepts and relationships of terms associated with medical domains which in turn improves the information processing, retrieval, and decision support. However, some of these applications do not explain their reasoning process; their knowledge base may not be built using the standard medical ontologies and they build ontology but fail to develop an application that a physician can use.

The main objective of this thesis was to investigate how an ontology and its generic tools can be used to overcome the above challenges; how diagnostic can be formally represented and to implement a clinical decision support system that uses that ontology and diagnostic criteria to assist physician when making a diagnosis of diabetes mellitus.

The ontology extends DDO and has been designed in protégé 5.0.0 OWL-DL. 19 rules were created based on diabetes diagnostic criteria by using jena rule syntax and forward chaining inference. Furthermore, the system uses the jena rule engine to reason with patient data from ontology against the rules defined in a rule file to generate a patient diagnosis, recommendation and decision explanation based on the patient vital signs and blood glucose test result. As result, 4 patient's case was successful diagnosed; recommendation and decision explanation produced by the system undoubtedly matched the patient diagnosis. Overall system result is the same as the one expected by the physician, thus, makes it an expert system. The ontology supports the interoperability between CDSS and healthcare system and can be used for the management of the patient.

Keywords: *Clinical decision support system, diagnostic criteria, semantic web, ontology, diagnosis*

Preface

This thesis is the result of the IKT 590 Master’s thesis project, at Faculty of Engineering and Science, department of Information and Communication Technology (ICT), University of Agder, Grimstad, Norway. The work has started from 1st of February and ended on 22nd of May, 2017. The main objective of this work was to investigate how semantic web technology can be used to address some challenges of medical diagnosis, specifically diabetes mellitus diagnosis.

I would like to deeply thank my thesis supervisors, Professor Jan Pettersen Nytun and Ghislain Maurice Norbert Isabwe for their guidance, support and feedback on both technical and contents of the thesis during this project.

Thanks to Doctors and nurses from Rwanda, who spent their valuable time helping me to understand medical field.

Thanks to Alyn Marie, who has spent her time formulating patient case and evaluating the system’s functionalities.

Finally, I would like to thank my family, friends and colleagues for your encouragement, support, prayers during these two years of studies at the university of Agder.

University of Agder, 2017

Mukabunani Alphonsine

Contents

List of figures

List of tables

List of abbreviations

API	Application Programming Interface
BFO	Basic Formal Ontology
CDSS	Clinical Decision Support System
DDO	Diabetes Mellitus Diagnosis Ontology
ICT	Information and Communication Technology
OBO	Open Biomedical Ontologies
OGMS	Ontology for General Medical Science
OWL	Web Ontology Language
OWL DL	Web Ontology Language Description Logic
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SCT	SNOMED Clinical Terms
SNOMED-CT	Systematized Nomenclature of Medicine-Clinical Terms
SPARQL	Simple Protocol and RDF Query Language
SWRL	Semantic Web Rule Language
UML	Unified Modeling Language
UMLS	Unified Medical Language System

1 Introduction

Clinical diagnosis is critical but often complex and error prone in the medical field. It is needed to provide a good and accurate treatment to the specific individual patient. With invention of computer and the rapid growth of information technology in the field of medicine, different researchers have developed applications that can assist physicians and other healthcare professional in their daily decision making tasks. These applications are called clinical decision support systems (CDSS) and have proven to be an efficient way to improve the quality of health care, the patient safety and the efficiency in delivering health care. However, there are still the need for improvement in different areas. This chapter presents the overview of clinical diagnosis. It provides the background on clinical decision support system along with the methodology for its development to be accepted in clinical practice. The problem statement, the review of literature, the proposed solution and the outline of the master thesis are also presented in this chapter.

1.1 Background

Clinical diagnosis is the process through which healthcare professionals attempt to identify the cause and nature of disease by carefully examining and analyzing the information collected through patient's history, physical examinations, and laboratory test **[1]**. The clinicians used to manually record all patient related information into paper (e.g. outpatient file/inpatient file/emergency file) and try to establish the necessary knowledge required for making a suitable decision. This approach is proved to be more complex, time consuming and even error prone. After recording patient information, the patient's paper health record is archived. However, the way it is archived is worried and information retrieval become an issue. Furthermore, the accuracy of their decisions depends on their medical experiences and skills in the medical domain. In other words, physicians who do not have enough experiences and skills in medical field face many challenges in their diagnostic process compared to others. However, even an experienced doctor can make a wrong diagnosis. Additionally, the diagnostic process become much worse when the signs and symptoms are not specific, that is, when a symptom or sign of one disease is present in more than one disease **[2]**. In this case, the diagnosis become much more confusing to many clinicians and consequently, the wrong diagnosis can be made which in turn affect the patient. In order to improve the quality of healthcare, patient safety and healthcare delivery, many applications have been developed to assist healthcare takers a in their daily decision-making process and these applications are called clinical decision support systems **[3]**. These systems are designed to provide variety of assistance (e.g diagnostic support, clinical guidelines, condition-specific order strategies, etc) and most of them focuses on diagnostic support. They are defined as an interactive computer system that are designed to assist clinicians in their daily decision making task. As diagnostic systems, they are intended to assist physicians in all processes involved in the diagnosis, that is, from initial consultation, diagnosis, treatment to follow up. They have to collect a huge amount of patient data (i.e manually inputted by clinician or from EMR), examine them, draw conclusion and send it to the clinicians in the form of alerts, recommendation or suggestions, diagnosis, and so on **[4] [5]**. They are not intended to replace the clinicians but rather to assist or guide them in taking a fast and accurate decision thus reducing the medical errors (both error of commission and omission) and improve the clinical practice **[6]**.

CDSSs come into two categories, that is, knowledge based CDSS and non-knowledge based CDSS **[7]**. Knowledge based CDSS have 3 main components: knowledge base and inference engine and a way to communicate to the user (input and output) **[8]**. Knowledge base is the main component of CDSS and contains all the information related to the domain of interest (e.g disease concepts, symptoms, etc.) in which CDSS works in. The information presented in knowledge base is obtained from domain expert and other literatures and need to be represented in a way that facilitate easy processing and retrieval. The clinician inputs the patient data, that is, information collected in patient history, physical examination (i.e signs and symptoms) and test results into the CDSS and inference engine applies rules stored in knowledge base to patient data and then produce the output for individual patient. The output can be the recommendation for the patient diagnosis, list of possible diagnoses, test ordering, the treatment if applicable, reminders (e.g preventive care), alerts (e.g drug-drug interaction) and so on **[8] [9]**. Non- knowledge based CDSS does not contain the knowledge base. Rather, it relies on machine learning approach and other statistic pattern recognition to find hidden pattern in patient-specific parameters. Non-knowledge based CDSS are not widely applied to diagnosis because their reasoning process is done in so called "blackBox" which means that the rules used in their reasoning do not follow any logic which makes their results not trusted by the physician **[10] [11]**.

Better understanding of the diagnosis process and the human reasoning when making diagnosis is important in the adoption and acceptance of CDSS. Similarly, knowledge representation, retrieval and sharing is also the key factors in the development of the CDSS which is beneficial to the health sector **[12]**. With the rapid growth of information technology in the field of medicine, different technologies and methods have been proposed to support the knowledge representation in CDSS **[1]**. However, many of them have been proven to be not suitable in developing CDSS with higher accuracy and acceptance. Nowadays, semantic web technology is the promising solution for knowledge representation, sharing and reasoning. Furthermore, the ontology and its generic tools were adopted in semantic web as a means of representing a knowledge formally to support reasoning and understanding of the domain of interest, not only by people but also by machine **[13]**. Additionally, the use of human centered design approach in the design and development of CDSS has been shown to be a best way of developing a usable and useful interactive system that meets user's needs which in turn increases the product adoption and acceptance rate in clinical practice as well as user satisfaction **[14]**.

The ontology is a word taken from the philosophy, which mean the study of things that exist in the world and their interaction. In computer science and artificial intelligence, the ontology is a formal framework for representing the domain knowledge. It defines and represents concepts, properties and relationships between those concepts within a domain of interest in a way that computer can understand and draw new knowledge from asserted ones **[15]**. As result, it helps people to have a common understanding of structured information as it is represented in unambiguous way, ensure interoperability among software agents, enable the reuse of existing domain knowledge as well as automatic reasoning **[13] [4] [16]**. Its use in CDSSs make the knowledge to be formally represented which in turn facilitates easy knowledge storage, sharing and retrieval as well as improving inference capability.

1.2 Problem statement

Correct diagnosis is a key to ensure the most appropriate treatment. Due to diagnostic errors, patients may suffer from many health problems including death, unnecessary tests and unnecessary side effects. This is because the methodologies used by many medical centers are not accurate and out dated. For instance, in different medical centers, like in Rwanda, they still record the patient information manually on the paper and try to establish the knowledge required to make the diagnosis. This knowledge which influences their decision can be from books (e.g. Guide therapeutic standard book), school, training, etc. In other words, the accuracy of the diagnosis depends on knowledge that a physician has on the patient case and may vary from one clinician to another. Additionally, clinician can fail to gather all patient medical history which also reduces the diagnosis accuracy. With the rapid grow of information technology, many applications have been developed to assist healthcare takers in their decision-making process, however, the methodologies and techniques used to build such applications is different and in turn it makes some of them not trusted by the physician. For instance, neural network approach which does not explain how the conclusion was reached. In addition, most of the existing applications face common problems, including poor designed interface, inability to let physician expressing their understanding on the patient case and they do not contain enough knowledge required for medical diagnosis. Furthermore, the staffs need to be trained before using such application which gives a burden to some countries with low income especially African countries. In this case, most of them are not interested in integrating these applications in their health care services.

The clinical decision support system described in this thesis is based on understanding the diagnostic process, diagnostic criteria, human reasoning on the diagnostic process, physician’s needs and will be easy and effective to use by all healthcare providers including those without higher computer literacy. This application will help them do the following functions:

- ✓ Storing the medical knowledge related to diabetes mellitus, symptoms and signs, laboratory tests, recommendations, patient’s details such as address, occupation, religion, health insurance, personal identification, to name a few
- ✓ Recording and storing the patient’s informations and make them accessible when needed.

- ✓ Let the physician selects either the vital signs/symptoms or diseases and gives back a list of possible diagnoses (i.e. diabetes mellitus in this case) or a list of symptoms or signs associated to the selected disease (i.e. diabetes mellitus ’s signs/symptoms).
- ✓ Suggest diabetes mellitus laboratory tests, store their result in the system, validate them and propose a final diagnosis, that is, if the patient is diabetic, prediabetic or normal.

1.3 Research questions

Our research questions are:

RQ1: How to use ontology and its generic tools to represent different diseases, symptoms, diagnosis and so on?

RQ2: How to formulate production rules based on the diagnostic criteria?

RQ3: How the system is to lead the dialog and expose the disease affecting the patient?

1.4 Literature review

This section provides an overview of clinical decision support systems, their type based on both architecture and inference engine methodology. It also gives an overview of existing biomedical ontologies and their limitation on disease classification. The factors that limit the CDSS effectiveness and adoption was discussed along with the guidelines that current developer should follow to develop a system with a higher accuracy.

1.4.1 Clinical decision support system(CDSS)

CDSS is defined as any computerized system that provides an immediate and complete decision support to a physician after analyzing the individual patient data. A typical CDSS has three main components, that is, knowledge base, inference engine and mechanism to communicate with the users **[17]**.

Having patient data (e.g clinical signs, symptoms, test results, etc), knowledge base containing medical knowledge mostly in form of if -then clauses, inference engine combines medical knowledge with patient data to produce an individual patient assessment or recommendation to a physician in form of alert (e.g drug-drug interaction), recommendation (e.g., possible diagnoses), reminder (e.g., preventive care reminder) and so on **[18]**. Note that not all CDSS contain Knowledge base, some use machine learning and other statistical pattern recognition to reason with clinical data, however, they are not widely used as diagnostic support.

CDSS has been proven to be the best solution to reduce adverse events that might occur in medicine such as medication errors, cost, test duplication, medication side effect, among others **[19]**. A CDSS is not designed to replace the physician instead to assist him/her in taking a fast and accurate decision that he/she may or may not take when working alone. In other words, CDSS is there to enhance clinician’s performance and patient outcome. More specifically, a physician should not rely only on the CDSS decision instead he/she should sometimes accept or reject it based on his/her own diagnostic capabilities.

1.4.2 Type of CDSS based on architecture

CDSS can be implemented using a variety of platforms based on a kind of problem being addressed and people’s needs. Many factors can influence CDSS implementation such as security, clinical work flow, existing clinical systems and so on. Author in **[20]** classifies CDSS into four categories based on their architecture, that is, standalone, integrated, standard-based and service oriented.

In standalone system**,** the CDSS is built independent from HIS or EMR and physicians are required to enter clinical findings into system, thus, make it time consuming and user driven, however, it can be easily moved from one institution to another. Integrated system, on the other hand, CDSS is tightly coupled with HIS or HER and patient information is obtained from HIS thus makes the system to proactively provide decision support without requiring the physician to input findings and could avoid data duplication. However, it is difficult to transfer and reuse clinical knowledge across applications since they are tightly coupled with vendor specific HIS and sometimes, HIS information is (1) unavailable, (2) often incomplete and (3) represented in unstructured and non-machine-understandable form which hinders an accurate decision to be taken. Standard-based system uses standard formalism to encode, store, and share clinical knowledge to enable interoperability and scalability **[21]**. Researchers and developers put much effort in the development of such application since they can deal with the complexity of the medical field, a huge amount of medical data and support data integration. The downside of such system is the lack of agreed standard for knowledge representation **[22]**. Finally, service-oriented system, CDS and HIS are implemented separately and standardized, service-based interfaces (i.e Application Programming Interfaces) are used to facilitate communication between them.

Each type of system has its own advantages and disadvantages and the choice depends on problem to be addressed and where the system is intended to be used. That’s why, we are going to develop a standalone, standard-based application that can assist physician when making diagnosis of diabetes mellitus.

1.4.3 Type of CDSS based on their inference mechanism

Medical diagnosis is very complex task that involves a carefully analysis and processing of patient information. In earlier days, this process was supposed to be done by a physician and the accuracy of decision varies from one physician to another based on their medical knowledge and skills. With the invention of computer and integration of ICT in domain of medicine, several systems to assist physician in their daily clinical decision making task are constantly developed. These systems aim to mimic human reasoning procedure when making medical diagnosis so that the produced decision would be the same as an expert clinician, however, the current system does not necessarily need to produce the right decision, instead to assist physician in taking an accurate decision. To achieve this, each system should be equipped with a reasoning component (inference engine) to reason with patient data. The accuracy of decision depends on the type of inference engine used, how the reasoning process is achieved, and how medical knowledge is represented. This sections gives an overview of type of CDSS based on inference used.

1.4.3.1 Rule based system

Rule based systems are the early type of CDSS (Knowledge based system). They are referred as expert system since they are designed to behave like an expert clinician. They have 5 components, that is, knowledge base, fact base, inference engine, explanation facilities and user interface **[23] [24]**. Knowledge base contains clinical knowledge encoded in form of IF-THEN statements to express logical process used by clinician when making diagnosis. Production rules are obtained from and maintained by a domain expert. Fact base contains a set of facts that inference engine uses to infer decision. An inference engine plays an import role in rule based system since it is the one which performs the reasoning task. It does so by matching facts contained in fact base with rule body to determine the rule to be fired. Explanation facility explains how the conclusion was reached. User interface allows the physician to interact with the system (i.e input patient data and receive output). A well-known example of rule based system is MYCIN **[25]**.

1.4.3.2 Ontology based system

Ontology was firstly used in philosophy where it was defined as the study of things which exist in real world**.** Later, it was adopted in the field of artificial intelligence and computer science and can be used in variety of applications such as information retrieval systems, internet search engine, medical diagnosis system, among others **[26]**. It was further adopted in semantic web technology as knowledge representation tool to support reasoning in semantic web application. Semantic web is described in **section2.2**.

The Ontology is defined as a formal knowledge representation of concept in the domain of interest. It represents a knowledge in term of concepts, properties and relationship between concepts. The ontology consists of the concept which defines the set of entities that exists in the domain, relationship that defines the interaction between those concepts, instance which represents the real-world objects within the domain and set of axioms that denotes the statements that need to be true **[26]**. The knowledge represented in ontology is not only human understandable but also computer understandable and processable, thus, make to it suitable for CDSS application. The advantage of using ontology in CDSS application includes easy sharing, reusing, reasoning, retrieving, and updating of clinical knowledge.

In medical field, several ontology-based systems are proposed for diagnosis and management of diabetic patient. Most of them use ontology for knowledge representation and rule for problem solving method. Alharbi et al proposes a diagnosis and treatment recommendation system for diabetes. The proposed CDSS is based on the use of domain ontology and rules. The rules are created in SWRL based on clinical practice guidelines and executed by Jess engine. To generate decision, the system takes into account patient information, sign and symptoms, laboratory tests and diabetes risk factors **[27]**. This system is incomplete since it only modeled an ontology and rules but failed to show a user interface that user can use to get assistance. Additionally, the ontology does not extract its concepts from standard ontology. Sherimon PC et al propose a rule based CDSS for assessing and predicting the risk of diabetic patient and suggest treatment. Patient information such as smoking, alcohol, physical activity, etc are collected through adaptive questionnaire which is served by adaptive ontology. The reasoning was done by inference engine which uses SWRL rules encoded based on clinical practice guideline **[4]**. This system does have an interface but failed to explain how inference was reached and does not use standard ontology during ontology conceptualization. Chen Rung-Ching et al propose a rule based system for recommending antidiabetic-drug to diabetic patient. The proposed system uses a domain ontology and SWRL rule; jess rule engine for logical deduction **[23]**. This system is complete but failed to use standard ontology. Other ontology based CDSS can be found in **[28]**, **[29]**, **[30]**.

1.4.3.3 Neural network based system

Neural network system uses machine learning approach and statistical pattern recognition to find hidden pattern in patient clinical data. It is commonly used in pattern recognition (e.g image recognition and speech recognition), natural language processing and medical applications (e.g radiography image classification, diagnosis support, prognosis evaluation, treatment planning decision and dentistry) **[31]**.

It was inspired by biological model of brain and consists of three layer (i.e input layer, hidden layer and output layer), each having a set of neurons interconnected with other neurons at next layer by means of weight. Neural network requires training to produce a better decision. In training process, training data set (i.e known patient clinical data) is fed to the input layer of neural network and processed by hidden layer by adjusting weight to derive hidden pattern in clinical data (i.e closer to desired output) and the result is sent to the output layer which communicate it to the user **[32]**. Once the network is trained, new patient case can be correctly classified.

The advantages associated with such system is its ability to work on incomplete data and does not rely on rules or direct input from expert, however, training process took much time, it does not explain how the conclusion was reached as the learning process is done in so called "black box" and its accuracy depends on training data set (i.e a large sample size data is required to produce better output). The example of neural network application developed for diabetes can be found in **[33] [34]**.

1.4.3.4 Genetic algorithm based system

Genetic algorithm based system uses multistep process to find problem solution. The process starts with the selection of initial population (i.e chromosomes) either randomly or based on predefined rules from large population and then applies fitness function to them to generate individual capable for reproduction. The generated individuals are muted and combined (i.e crossover) to produce new generation (i.e descendants) and the process repeats until the fittest individual (i.e solution) is obtained or when desired generations are reached. Genetic algorithm was rarely used in medical application, more specifically diagnosis application.

1.4.4 Existing medical vocabulary and their limitations on disease diagnosis

In the past decade, medical centers, researchers and industries have started building several ontologies. These ontologies aim to represent clinical information so that it can be shared and reused among people or software agents. There are several ontologies that have been developed to provide common vocabularies for medical terms. Some of them are SNOMED-CT, Disease ontology, symptom ontology, UMLS, GALEN, MED and Gene Ontology **[35]**. They play an important role in biomedicine such semantic interoperability; data integration and exchange; and they play an important role in integrating data from diverse sources **[36]**. Most of these ontologies represent relationship between concepts by using “IS-A” relationship. Many studies have been performed to evaluate the completeness of knowledge represented in these ontologies and to see if these ontologies are suitable for disease classification. Author in **[37]** argues that these biomedical ontologies have a lot of entities that they have not adequately treated in their vocabulary like entities related to the way the diseases are recognized and interpreted by clinician. This is because they represent diseases based on etiological representation rather than operational definition of disease which make it difficult to do the disease classification. More specifically, they do not precisely indicate the necessary conditions (criteria) to establish the diagnosis of diseases.

The authors in **[38]** also discuss the limitations of SNOMED-CT on diagnosis of spondyloarthritis and propose a novel approach to overcome them. They propose an ontological representation of spondyloarthritis by using the diagnostic criteria presented by ASAS (Assessment of SpondyloArthritis). Ontology was developed using protégé 4.1 and OWL-DL2.0 and was tested using reasoner. As result, thirty patients were successfully classified. Authors in **[39]** also used the clinical practice guideline to develop a clinical decision support system that can be used to encourage good opioid prescribing practices during primary care visits. However, as discussed earlier, to ensure a common understanding and interoperability between people and software agents, it is recommended to represent medical knowledge using standard medical vocabularies. The above-mentioned systems failed to do so.

1.4.5 CDSS effectiveness and adoption

Research on clinical decision support system has been started in 1950s. Since then, many developers have put considerable efforts in the development of such applications and many of them were focused on diagnostic support. The first clinical decision support system was rolled out in 1970s. This system was called MYCIN and was designed to assist clinicians in diagnosing infectious diseases and choosing an accurate drug. A physician answers a series of questions to get better suggestion (i.e ordering test and treatment recommendation) and can ask the system how the conclusion was reached. Even though this system was considered as an expert system, it was not widely used because of complexity to maintain its knowledge base and integration in clinical workflow **[40]**. Few year later, other systems have been developed, however, most of them was also rarely adopted in clinical practice. There are many factors that limit adoption and acceptance of clinical decision support system **[19]**. Some of them includes the way data are inputted into the system; the development, maintenance, sharing and reuse of knowledge base; user interface; physician’s computer literacy and time.

Meenal B et al propose guidelines that every clinician should follow to ensure that the CDSS fit into clinical workflow and meet the requirement of its intended users. They have summarized them into 5 rights which are “provision of the right information, to the right person, in the right format, through the right channel, at the right point in workflow “ **[41]**. In other words, CDSS intervention or recommendation should be provided to the right physician at the point of care (i.e., not before and after an encounter). It should be made for patient-specific data and presents information in a form that allow easy and unambiguous interpretation and should be justifiable (i.e., explaining how the conclusion was reached) **[42]**. According to the systematic review conducted by Kawamoto Kensaku et al, 30 out of 32 clinical decision support system having these four features have proven to improve clinical practice **[43].**

The author in **[44]** also provides a list of recommendations that the current developer should follow to develop a CDSS with a higher adoption rate. These includes the use of standard vocabulary and knowledge representation standard to develop a machine-processable knowledge base and that knowledge base should be built based on clinical evidence and scenarios suitable for your system and son on. Other recommendations and desired features of CDSS can be found in **[45] [46]**.

1.5 Proposed problem solution

The main task of this thesis is to develop a clinical decision support system that can be used for diagnostic of diabetes. It could assist Rwandan clinicians at the point of care and the decision is made based on the individual patient's conditions. It is intended to replace paper-based diagnosis that is being used in Rwanda thereby enhancing physician performance as well as the patient outcome. The summary of proposed solution is shown in **Figure 1.1**. The middle of the figure represents the current diagnostic process while the right and left of the figure represent a proposed novel approach for making a diagnosis. The proposed approach uses the ontology to model a knowledge base that contains the key concepts and relationships from diagnostic criteria of diabetes to facilitate sharing, reuse, retrieval and update of clinical knowledge. It also based on decision rules that was created and formalized based on jena rule syntax. The patient data, vital sign, test result was extracted through user interfaces and formalized into ontology by using property and individual insertion and SPARQL update query. Finally, they are sent to jena rule engine which matches a set of facts from ontology against rule set to deduce a diagnosis, recommendation and explanation on how a conclusion was reached. The result is made available to the physician through user interface. The implementation of this solution is described in detailed in section 3.6.

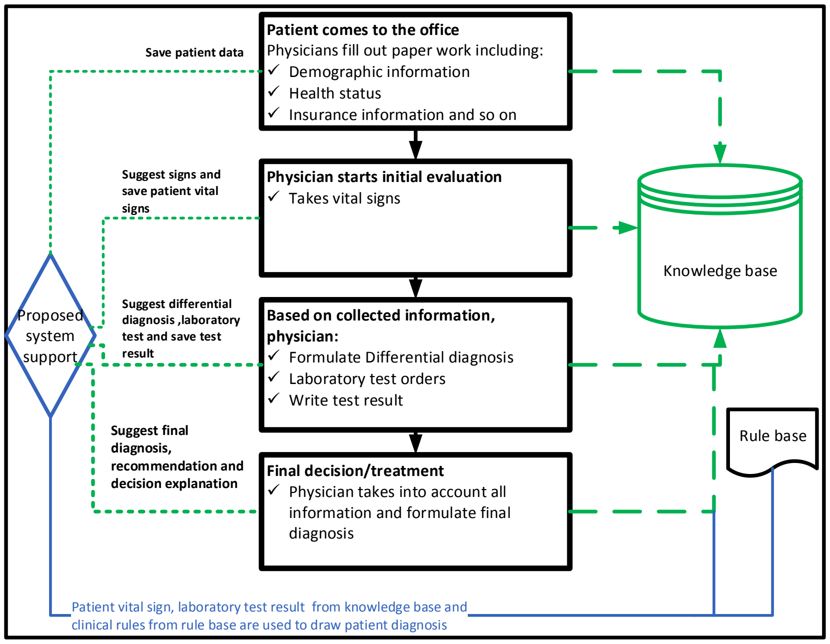

Save patient data
Patient comes to the office
Physicians fill out paper work including:
✓ Demographic information
✓ Health status
✓ Insurance information and so on
Suggest signs and save patient vital signs
Physician starts initial evaluation
✓ Takes vital signs
Knowledge base
Suggest differential diagnosis ,laboratory test and save test result
Proposed system support
Based on collected information, physician:
✓ Formulate Differential diagnosis
✓ Laboratory test orders
✓ Write test result
Rule base
Suggest final diagnosis, recommendation and decision explanation
Final decision/treatment
✓ Physician takes into account all information and formulate final diagnosis
Patient vital sign, laboratory test result from knowledge base and clinical rules from rule base are used to draw patient diagnosis

Figure 1.1: Proposed solution

1.6 Report outline

The rest of this thesis is organized in six chapters as follow**:**

Chapter two presents a brief introduction of technical background adopted in this thesis. These include languages and standards adopted in semantic web technology as well a brief introduction on diabetes, its blood test and diagnostic criteria.

Chapter three presents a detailed description of proposed solution, including the approaches and methods used during the design and development of the system; the designed solution and its implementation.

Chapter four shows the obtained outcome of the system based on test performed on real patient clinical data.

Chapter five presents the discussion carried out based on the obtained results.

Chapter six concludes this thesis by giving the summary and future work.

2 Theoretical and technical background

In this chapter, we present a background information that is relevant to the thesis. It mainly focuses on diabetes mellitus; standards and languages that enable semantic web technology.

2.1 Diabetes

Diabetes is a disorder of metabolism that is characterized by hyperglycemia due to the low production of insulin or inefficient use of produced one or body does not produce the insulin at all **[47].** This can lead to different health complications once diabetes is uncontrolled including heart disease, stroke attack, kidney failure, and blindness or even death. There are 3 types of diabetes, that is, type 1 diabetes, type 2 diabetes, and gestational diabetes. Type 1 diabetes is insulin-dependent diabetes and affects young people; type 2 diabetes is non-insulin dependent diabetes and account 92 % of diabetes case in adult people. Both adult people and overweighed young people can be affected by this type of diabetes and gestational diabetes is the type of diabetes that affect pregnant people and must be well controlled as it can affect both mother and unborn baby. This type of diabetes can cause abortion or risk of having type 2 diabetes at 10 years after giving birth, However, the mother can’t contaminate an unborn child. Patient having diabetes are characterized by hunger, fatigue, frequent urination, thirsty, dry mouth, etc. the symptom of diabetes includes polyuria, polydipsia or unexplained loss.

2.1.1 Diabetes blood test

In addition to the symptom of diabetes, different tests are also performed by physician to confirm the existence of diabetes. These are blood glucose tests, including glycated hemoglobin (A1c) test, a fasting plasma glucose (FPG) test, an oral Glucose tolerance (OGT) test and casual plasma glucose test. Among these tests, the fasting plasma test is the most preferred method to diagnose diabetes mellitus as it is simple, appropriate and very cheap to perform compared to other tests according to American diabetes association **[48]**. It is performed on the patient who has spent 8 hours without taking any meals specifically in the morning and the diabetes is diagnosed to a patient with blood sugar level of greater or equal to 126 mg/dl. When the blood sugar level is normal that is, having the value less than 110 mg/dl and that the patient has the sign and symptoms of diabetes, the doctor can order oral glucose test Tolerance test or can repeat FPG test at the following day. If the test result shows the blood sugar level which is very high (ie Greater than 200 mg/dl), then diabetes is confirmed. In addition, this test is usually performed on pregnant people to diagnose gestational diabetes.

The casual plasma glucose test is also another method to diagnose diabetes to the patient regardless the time of his/her last meal. when the glucose level is greater than 200mg/dl, the diabetes is confirmed. The hemoglobin A1c test is also an agreed method to diagnose diabetes mellitus when its value is equal to or greater than 6.5%. It is also used to determine the way your diabetes is being controlled and helps in diabetes management.

Having patient with symptoms of diabetes mellitus and at least two positive finding of these tests is sufficient to confirm the existence of diabetes. There is no specific treatment for diabetes mellitus, however, taking healthy diets and doing physical activities, avoiding smoking, insulin injection, stress level reduction and normal weight maintenance are recommended for the patients having diabetes as it helps him to live with diabetes easier.

2.1.2 Diagnostic criteria of diabetes

The diagnosis of specific disease involves the results of sequential steps. The first step, a physician takes medical history (i.e the Symptom), the second step involves performing physical examination (i.e the signs), the last step is ordering a laboratory test. After going through all these steps, physician uses his knowledge and skills to analyse all these informations and comes up with a right diagnosis. All these steps should be performed to ensure that the accurate and better diagnosis decision is taken. In order to ensure that all the clinician’s decision is consistent and that the quality of care is the same to all patient, different diagnostic criteria were developed to assist them when making the diagnosis. These are the standardized definitions that involve a combination of sign/symptom, and test results that a clinician can use and reach a correct diagnosis. Additionally, the threshold values for test’s measurements are specified to identify patient with disease from patient without. According to systematic review that was conducted, Farquhar et al found out that these criteria are good source of advice, improve quality of care and serve as a tool for education **[49]**.

Diagnostic criteria have been played an important role in clinical research and has been used in clinical practice with the intention of identifying as many people with condition as possible and was designed to be applied to individual patient **[50]**. There are many diagnostic criteria that was developed specifically for diagnosis of diabetes. For instance, diabetes criteria from American diabetes association journal **[51]**, diabetes criteria that are found on American Association website **[52]** and so on. **Figure 2.1** and **Figure 2.2** show an example of diagnostic criteria of diabetes.

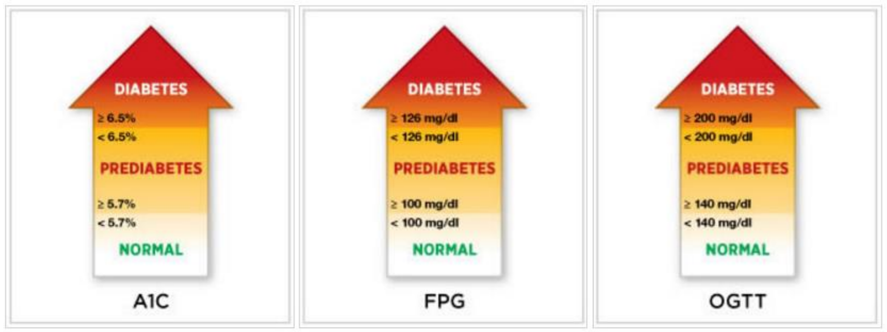
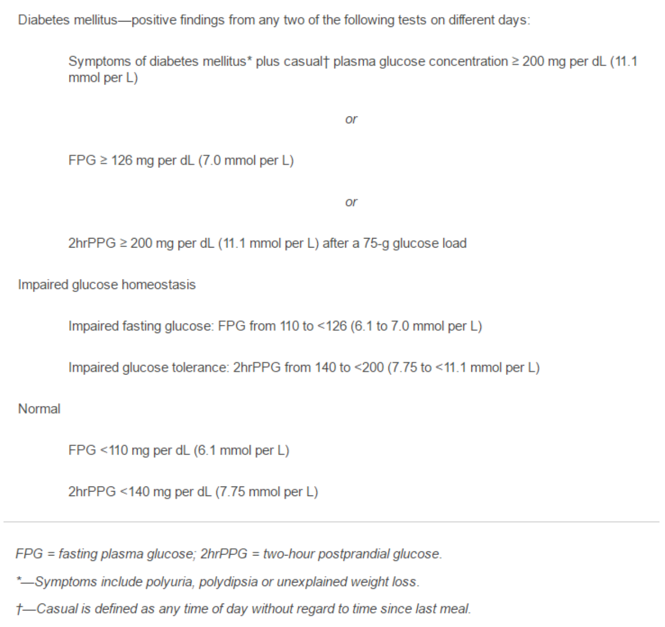


Figure 2.1: Diabetes examination test results [52]



Diabetes mellitus—positive findings from any two of the following tests on different days:

Symptoms of diabetes mellitus* plus casual† plasma glucose concentration ≥ 200 mg per dL (11.1 mmol per L)

or

FPG ≥ 126 mg per dL (7.0 mmol per L)

or

2hrPPG ≥ 200 mg per dL (11.1 mmol per L) after a 75-g glucose load

Impaired glucose homeostasis

Impaired fasting glucose: FPG from 110 to <126 (6.1 to 7.0 mmol per L)

Impaired glucose tolerance: 2hrPPG from 140 to <200 (7.75 to <11.1 mmol per L)

Normal

FPG <110 mg per dL (6.1 mmol per L)

2hrPPG <140 mg per dL (7.75 mmol per L)

FPG = fasting plasma glucose; 2hrPPG = two-hour postprandial glucose.

**—Symptoms include polyuria, polydipsia or unexplained weight loss.*

†—Casual is defined as any time of day without regard to time since last meal.

Figure 2.2:Criteria for the diagnosis of diabetes mellitus and impaired glucose homeostasis [53]

Those criteria are mostly in paper based form and physician can fail to obtain the available documentation. Thus, make them not widely used. Additionally, it takes time for the physician to consult these hard documents when making diagnosis. As result, there is a need to turn these criteria into computer interpretable form and use them in clinical decision support system to generate patient-specific recommendations. The challenge lies on how to represent diagnostic criteria in computer understandable and processable form. There are various standard and language that can be used for that purpose and are discussed in **[54]**. During this thesis, we used an ontological approach to encode key concepts and relationship in diabetes diagnostic criteria to facilitate clinical knowledge sharing, reuse, retrieval and update. Furthermore, production rules were created using jena syntax to enable inference of diagnosis, recommendation and decision explanation. Table 2-1 describes the diabetes laboratory tests with their threshold measurements (i.e blood glucose level). It also describes the decision that can be taken by a clinician based on the obtained results.

Table 2-1: description of diagnostic criteria of diabetes

Test	Result(mg/dl)	Decision1	Another test?	What's next?
A1c	≥6.5	Diabetes	Yes	Ask a physician to select another test and provide a result.
	≥5.7<6.5	Prediabetes	No	Give final diagnosis.
	<5.7	Normal	No	Give final diagnosis.
FPG	≥126	Diabetes	Yes	Ask a physician to select another test and provide a result.
	≥110 <126	Impaired fasting glucose	No	Give final diagnosis.
	<110	Normal	No	Give final diagnosis.
OGT	≥200	Diabetes	Yes	Ask a physician to select another test and provide a result.
	≥140 <200	Impaired Glucose tolerance	No	Give final diagnosis.
	<140	Normal	No	Give final diagnosis.
RPG	≥200	Diabetes	Yes	Ask a doctor to select another test and provide a result.
	<200	Normal	No	Give final diagnosis

2.2 Semantic web

Semantic web (web of data) is the idea that was conceptualized by Tim Berners-Lee in 2001. Its intention is not to replace the existing world wide web (web of documents) rather to extend it by adding the semantic (meta-data) to the web contents so that software agents and people can understand, infer the web resources and work in cooperation. This is achieved by the use of a set of technologies and standards that enable the presentation and publication of resources on the web in machine readable and processable form **[55]**. It has been a promising solution in the development of many applications including medical reasoning and clinical decision support systems as it facilitates an easy retrieval of biomedical vocabularies, terminologies and taxonomies **[56]**. **Figure 2.3** shows seven layers of semantic web stack that contribute to its success in knowledge integration, querying and sharing. Each layer is built on top of each other where the layer above exploits the features and uses capabilities of layer below. The two bottom layers are not new. They are already used in the current web. Semantic web is built on top of them and adds semantic to information contained in a web document to enable reasoning and interoperability between software agents. This section gives the description of new layers defined in semantic web and are the one used during this thesis.

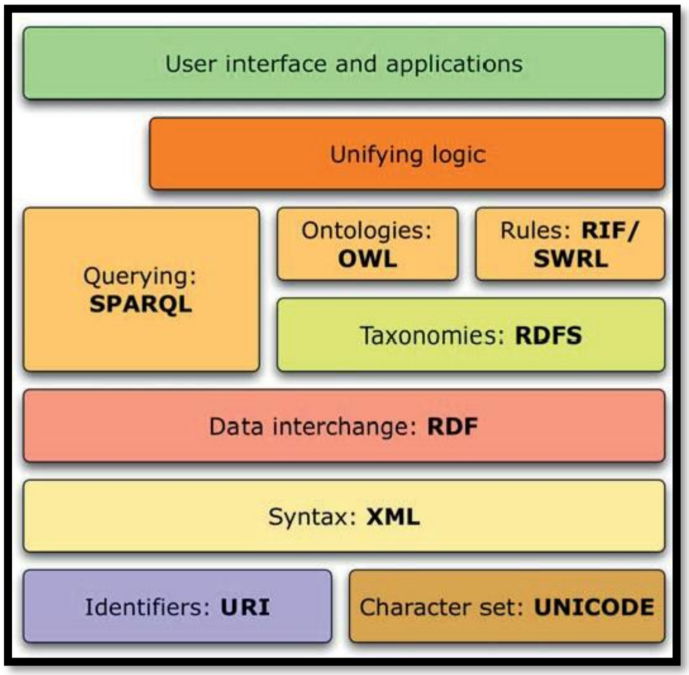


Figure 2.3:Semantic web stack [57]

2.2.1 RDF

RDF stands for Resource Description Framework. It is not a language rather a framework for describing resources into the web and helps to make statement about resource in form of triples; that is, subject, predicate and object. The subject represents resource that is going to be described and can be an IRI or a blank node. Predicate is the property of that resource. More specifically, predicate describes the binary relationship between subject and object and is always an IRI. The object is the value of that resource which corresponds to the intersection of row and column in the traditional relational database table. It can be an IRI, blank node or literal value. The set of triples form graph where nodes are subject and object while predicate presents the edge of the graph. Subject, object and predicate in RDF are identified by unified resource identifier (URI) **[58]**.

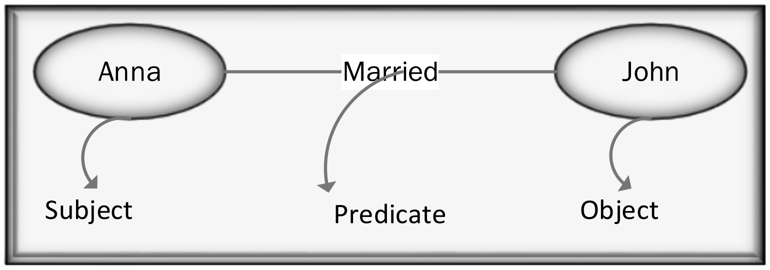


Figure 2.4: RDF example

Example of RDF triple is shown in **Figure 2.4** and says that Anna married John. In this case, Anna is subject, married is predicate and John is object. The predicate here shows a relationship between Anna and John. In addition, subject of a triple can be an object of one or more triples. Similarly, an object can be the subject of one or more triples, in this case, RDF is viewed as directed graph.

2.2.2 RDFS [58]

With RDF one can make statement about anything from any domain. So, there is a need for distinguishing resources. For instance, some resources may have common characteristic or may be the same**.** W3C has recommended RDFS for this purpose**.** RDFS stands for RDF Schema and defines the semantic vocabularies for RDF resources. It allows the definition of simple ontologies by defining classes, taxonomies (i.e superclass-subclass), properties and sub-properties, thereby providing a simple inference capability.

2.2.3 OWL

RDFS is suitable for defining simple ontologies and simple inference capabilities. When more detailed ontology is needed, a language that can provide additional standardized vocabularies to define concepts, properties and so on is required. That’s why Owl has been recommended by W3C to provide richer vocabularies and a stronger inference capability. It is written in XML and is built on top of RDF. OWL is a semantic markup language designed for a developing semantic web application and is based on the description logic. It allows a user to create/edit the ontologies which behave like other web documents but with semantic added to them so that the machine can understand, process and infer new knowledge from the asserted one with the help of different reasoners. These reasoners are the tools equipped with capability of inferring a new knowledge that was not asserted in ontology and provide inconsistency checking in the ontology, automatic classification of classes, and instance checking **[60]**. With OWL, one can represent a domain knowledge as a set of concepts (i.e classes), individuals, properties and defines the relationship between concepts in a way that a computer can understand **[61]**.

Writing ontology in XML format is difficult and time consuming because of the language syntax and complexity. There are various tools that allow users to intuitively create and edit OWL ontology without having to worry about complexity and syntax of the language. They get support different reasoners for automatic reasoning. Those tools include protégé, swoop, among others.

Owl has three specifications: OWL Lite, OWL DL, and OWL full. They differ from each other depend on the level of expressivity and decidability. The simplest one is OWL Lite which defines taxonomies and simple constraints. OWL DL is based on description logic and OW full provides maximum expressiveness.

2.2.4 SPARQL

SPARQL stands for Simple Protocol and RDF Query Language. It is a query language recommended by W3C and is used to retrieve information from an RDF data model as opposed to SQL which is used to query relational data. As we have discussed earlier, we have seen that the RDF data are represented in form of a triple, that is, subject, predicate, and object and that a set of triples form a graph. These triples are the one used in the SPARQL query and when it contains variable, it is referred as triple pattern. **Figure 2.5** at point 1 shows the informal representation of triple asserted in RDF model (Alpha hasDiagnosis Types2Diabetes) and point2 of this figure shows how the SPARQL allows a user to form a query (i.e question) that retrieves information from RDF model based on the already defined triples in the model (who hasDiagnosis of Type2Diabetes?).

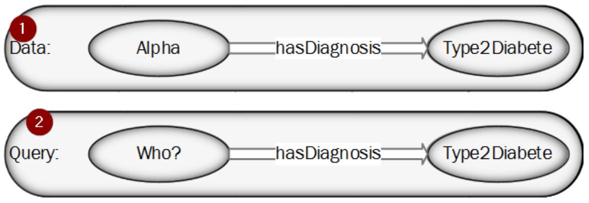
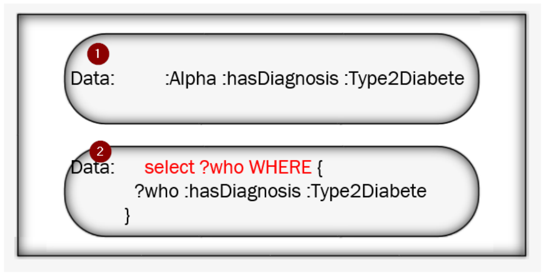


Figure 2.5:SPARQL query example

SPARQL supports different forms of a query such as select, insert, delete, construct, ask and describe that allows the user to manipulate the data stored in RDF format **[62]**. **Figure 2.6** at point2 shows the example of SPARQL select query that is used to query the data shown at point1 of **Figure 2.6**. Basically, SPARQL query may have one or more triple patterns (referred as basic graph pattern), that match a subgraph of the RDF data. Note that triple patterns are like RDF triples except that triple patterns may use variable in subject, predicate and object **[63]**. The example below consists of one triple pattern. SELECT statement denotes the subset of the selected data to be returned (e.g ?who) while WHERE clause consists of the graph pattern to be matched against the RDF graph **[64]**. This query will match any node which is related to type2Diabete through hasDiagnosis property and returns Alpha as result.



```
:Alpha :hasDiagnosis :Type2Diabete
```

```
select ?who WHERE {
  ?who :hasDiagnosis :Type2Diabete
}
```

Figure 2.6: SPARQL query example1

2.2.5 SWRL

SWRL stands for Semantic Web Rule Language. It is the rule language that was proposed by World Wide Web Consortium in 2004 and was designed to be used in semantic web. It was recommended to overcome the expressivity limitation of Owl, since OWL language is not able to represent complex relationship (e.g property chaining). It does so by including horn-like rule into OWL ontology. In other words, it combines the OWL sublanguages, that is, OWLDL and OWL lite with ruleML sublanguages, that is, unary/binary Datalog RuleML **[65]**.

Rules consist of two parts: antecedent part (i.e body) and consequent part (i.e head), each of which consist of a (possibly empty) set of atoms connected with conjunctions (i.e "^") and have the implication symbol (i.e "→") to imply logical relationship between body part and head part. This means that if all statements which are in body part are determined to be true, then all the statements that are in the head part must also be true. In this case, new property can be assigned to individual or new subsumption can be inferred.

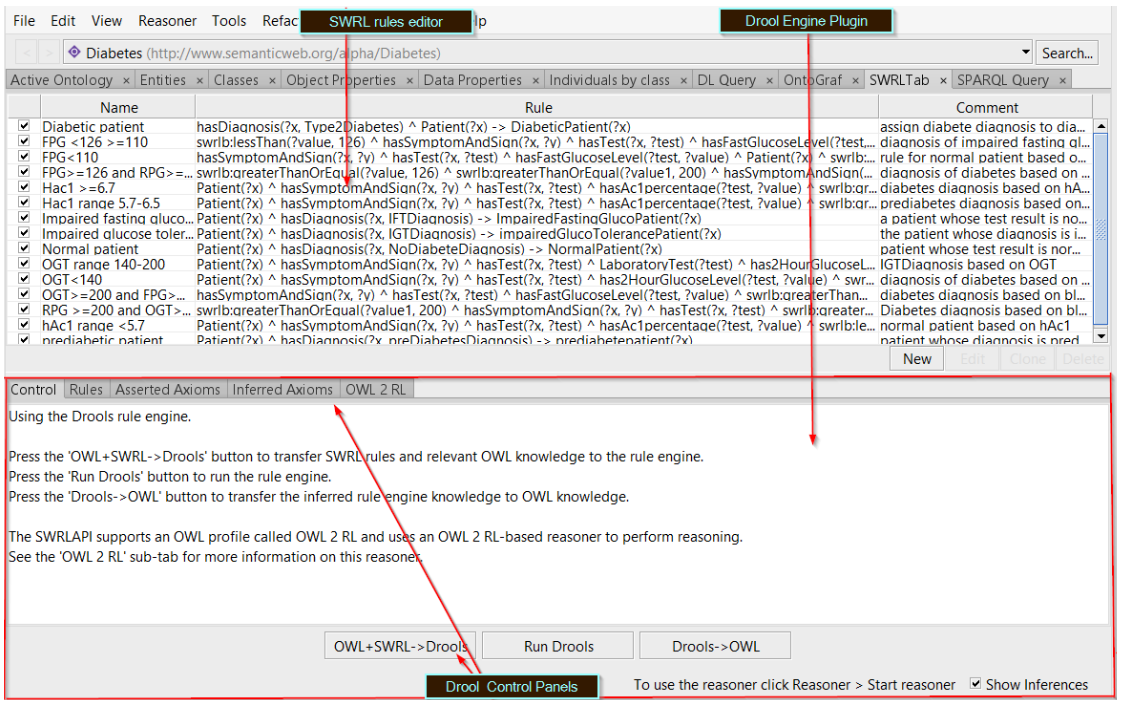
The rules are expressed in term of owl concepts such as classes, properties and individuals already defined in owl ontology and the reasoning is performed on individuals. Additionally, SWRL provides many built-ins functions that enable mathematical and string operation and allows a user to define their own built-in while developing an application. The most known example of SWRL rules is uncle relationship which is expressed as follows: Parent (?x,?y)^hasBrother (?y,?z)->HasUncle (?x,?z). This rule state that if person's parent has brother, it implies that this brother is person's uncle, that is, if Teddy has parent Anna and Anna has brother Kalisa, then Teddy has uncle Kalisa.

2.2.5.1 SWRL editor

Protégé-OWL offers SWRL editor and can be accessed as a tab within it. This SWRL editor lets developer interactively create SWRL rules, editing and read the existing one. It gets supporting different reasoners which are also part of protégé plugin. The reasoner that support SWRL includes pellet, hermit and so on. Protégé allows users to create/edit both owl ontology and SWRL rule and can intuitively switch from one editor to another.

2.2.5.2 SWRL rule engine bridge

SWRL editor provides SWRL rule engine bridge that supports the interoperability between an OWL knowledge base with SWRL rules and the third part rule engine (e.g Jess or Drool engine) **[65]**. **Figure 2.8** shows the interoperability between OWL ontology, SWRL rules, drool rule engine and the way the implicit knowledge is inferred from the asserted knowledge based on the SWRL rules. Both rules and OWL knowledge are created using protégé. After rule and ontology creation, they all transferred to the Drool engine through SWRL Drool tabs. This interaction is user driven. This means that a user chooses the action to be performed based on his need (e.g when to transfer OWL knowledge and SWRL rules, when the inference process is going to start and when the inferred knowledge is going to be transferred back to OWL ontology).



	Name	Rule	Comment
✓	Diabetic patient	hasDiagnosis(?x, Type2Diabetes) ^ Patient(?x) -> DiabeticPatient(?x)	assign diabete diagnosis to dia...
✓	FPG <126 >=110	swrlb:lessThan(?value, 126) ^ hasSymptomAndSign(?x, ?y) ^ hasTest(?x, ?test) ^ hasFastGlucoseLevel(?test,...	diagnosis of impaired fasting gl...
✓	FPG<110	hasSymptomAndSign(?x, ?y) ^ hasTest(?x, ?test) ^ hasFastGlucoseLevel(?test, ?value) ^ Patient(?x) ^ swrlb:...	rule for normal patient based o...
✓	FPG>=126 and RPG>=...	swrlb:greaterThanOrEqual(?value, 126) ^ swrlb:greaterThanOrEqual(?value1, 200) ^ hasSymptomAndSign(...	diagnosis of diabetes based on ...
✓	Hac1 >=6.7	Patient(?x) ^ hasSymptomAndSign(?x, ?y) ^ hasTest(?x, ?test) ^ hasAc1percentage(?test, ?value) ^ swrlb:gr...	diabetes diagnosis based on hA...
✓	Hac1 range 5.7-6.5	Patient(?x) ^ hasSymptomAndSign(?x, ?y) ^ hasTest(?x, ?test) ^ hasAc1percentage(?test, ?value) ^ swrlb:gr...	prediabetes diagnosis based on...
✓	Impaired fasting gluco...	Patient(?x) ^ hasDiagnosis(?x, IFTDiagnosis) -> ImpairedFastingGlucoPatient(?x)	a patient whose test result is no...
✓	Impaired glucose toler...	Patient(?x) ^ hasDiagnosis(?x, IGTDiagnosis) -> impairedGlucoTolerancePatient(?x)	the patient whose diagnosis is i...
✓	Normal patient	Patient(?x) ^ hasDiagnosis(?x, NoDiabeteDiagnosis) -> NormalPatient(?x)	patient whose test result is nor...
✓	OGT range 140-200	Patient(?x) ^ hasSymptomAndSign(?x, ?y) ^ hasTest(?x, ?test) ^ LaboratoryTest(?test) ^ has2HourGlucoseL...	IGTDiagnosis based on OGT
✓	OGT<140	Patient(?x) ^ hasSymptomAndSign(?x, ?y) ^ hasTest(?x, ?test) ^ has2HourGlucoseLevel(?test, ?value) ^ swr...	diagnosis of diabetes based on ...
✓	OGT>=200 and FPG>...	hasSymptomAndSign(?x, ?y) ^ hasTest(?x, ?test) ^ hasFastGlucoseLevel(?test, ?value) ^ swrlb:greaterThan...	diabetes diagnosis based on bl...
✓	RPG >=200 and OGT>...	swrlb:greaterThanOrEqual(?value1, 200) ^ hasSymptomAndSign(?x, ?y) ^ hasTest(?x, ?test) ^ swrlb:greater...	Diabetes diagnosis based on bl...
✓	hAc1 range <5.7	Patient(?x) ^ hasSymptomAndSign(?x, ?y) ^ hasTest(?x, ?test) ^ hasAc1percentage(?test, ?value) ^ swrlb:le...	normal patient based on hAc1
✓	prediabetic patient	Patient(?x) ^ hasDiagnosis(?x, preDiabetesDiagnosis) -> prediabetepatient(?x)	patient whose diagnosis is pred

Using the Drools rule engine.

Press the 'OWL+SWRL->Drools' button to transfer SWRL rules and relevant OWL knowledge to the rule engine.
Press the 'Run Drools' button to run the rule engine.
Press the 'Drools->OWL' button to transfer the inferred rule engine knowledge to OWL knowledge.

The SWRLAPI supports an OWL profile called OWL 2 RL and uses an OWL 2 RL-based reasoner to perform reasoning.
See the 'OWL 2 RL' sub-tab for more information on this reasoner.

Figure 2.7:The screenshot of SWRL rules editor and drool engine interaction

Figure 2.7 shows the screenshot for SWRL rule editor with example of rules and Drool engine plugin with different control buttons. User can create and edit existing rules within this editor and the interoperability with third-party rule engine is performed through different Drool's tab. SWRL Drool engine has several tabs including control, rules, asserted axioms, inferred axioms, OWL2 RL) which allows the user to see what's going on while interacting with Drool engine. Control tab contains three control buttons, that is, OWL+SWRL->Drools, run Drools and Drools->OWL [67]. When a user presses OWL+SWRL->Drools button, all SWRL rules and relevant OWL knowledge are transferred to the Drool engine. The transferred knowledge is represented as Drool facts and Drool rules respectively as shown on the **Figure 2.8**. Now, user can see the number of things exported to rule engine (e.g the number of rules, number of owl axioms…). When run Drools button is pressed, the Drool engine invokes its inference engine and new facts are generated as shown on **Figure 2.8**. Note that SWRL reasons only about OWL named individual. In this case, the asserted

OWL property or OWL class can be assigned to the named individual within OWL ontology **[68]**. The inferred axioms can be seen through inferred axioms tab. When Drools->OWL is pressed**,** the inferred axioms are translated to OWL syntax and transfer them back to OWL knowledge base. SWRL was a good approach to use in this thesis, however, it would require the use of OWL API and pellet reasoner that we were not familiar with. Our project works with jena API. Different rules were created based on jena rule syntax and jena rule engine was used to infer new knowledge based on facts represented in ontology.

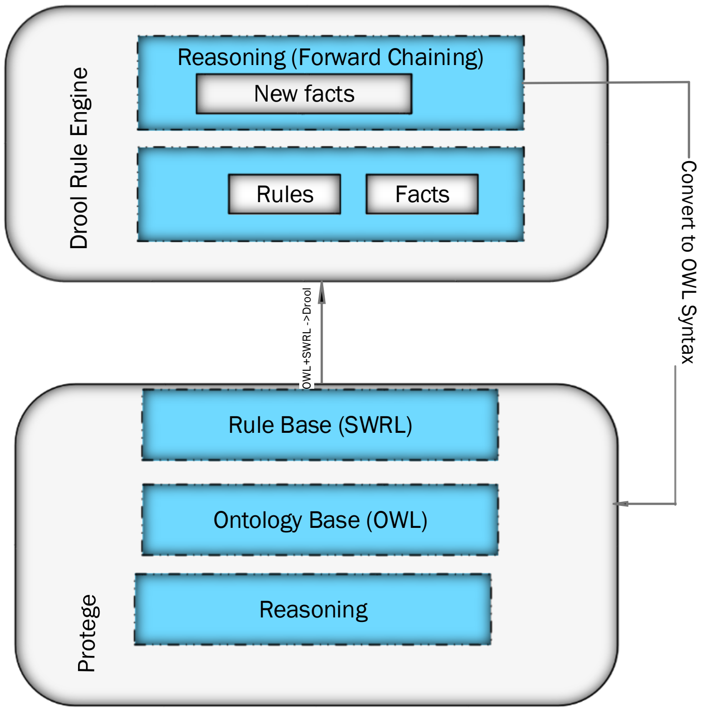


Figure 2.8: SWRL conversion

2.2.6 Jena

Jena is a free and open source a java framework that is used to build semantic web and linked data applications **[69]**. It gives a programmatic environment for RDF, RDFS, OWL, and provides ARQ engine that supports SPARQL query **[69]**. With jena, a user can create, read, write and manipulate RDF model and save it in memory or persistent storage (i.e database). Jena can read/write the file in a different format such as RDF/XML, N3, and N-triple and this file can be located in either a file system or be download from a given URL. Additionally, Jena provides several reasoners such as the RDFS reasoner, OWL reasoner, Transitive reasoner and provides the general-purpose rule-based reasoner that performs reasoning on both RDFS and OWL knowledge base and is available for general use. General purpose rule engine reasons with user defined rules over RDF graphs and provides forward chaining, tabled backward chaining and hybrid execution strategy **[69] [70]**.

2.2.6.1 Writing jena rules

The jena rules can be created using text file, or in application using a String data type and is defined by java rule objects. It consists of condition (i.e premise/if clause/body), conclusion (i.e head/then clause), optional rule name and direction **[71]**. Like SWRL, Jena rule consists of a set of atoms which are implicitly connected by conjunction. Implication symbol ("→" or" ←") between condition and conclusion is used to indicate the type of inference engine that will be used (forward or backward).

Forward chaining of inference is known as data-driven approach, that is, it starts with facts from the model and tries to find out the rule whose condition part satisfies the current content of the model. If the match is found, then the rule is fired and its conclusion part is executed. In this case, new facts can be added or deleted to the model. Backward chaining of inference on the other hand, is known as goal-driven approach, that is, it starts with goal and tries to find out the rule whose conclusion part matches the desired goal. If a match is found, it attempts to satisfy the goal by matching the condition part of the rule with any triple stored in the model **[72]**.

Both forward and backward rule can be used within single rule when hybrid chaining is used. This implication can be defined as "whenever the condition is satisfied, then execute the head". As result, a new statement can be added, removed/update into the model. The collection of rules is referred as rule set. The informal description of rule syntax and structure can be found in **[73]**.It consists of a list of terms in both conditions and conclusion part where a term can be a triple pattern, extended triple pattern or invoke procedural primitive. All terms in condition part must match so that the conclusion can be executed. By default, the conclusion is considered to be true when the condition part is empty while empty conclusion is treated as false. Jena provides several built-in functions that serve different purposes including addition, string concatenation, comparison and so on. Disjunction and negation are not supported in jena. To express negation noValue (?x, ?p) built-in can be used.

3 Solution design and development

The aim of this project is to design and implement an ontology-based clinical decision support system that will assist Rwandan physician when making diagnosis of diabetes mellitus. This section describes the approaches, methods and tools that were used during the period of this thesis in order to achieve the above-mentioned objective. The system's functionalities, design, and implementation are also described in detail in this section.

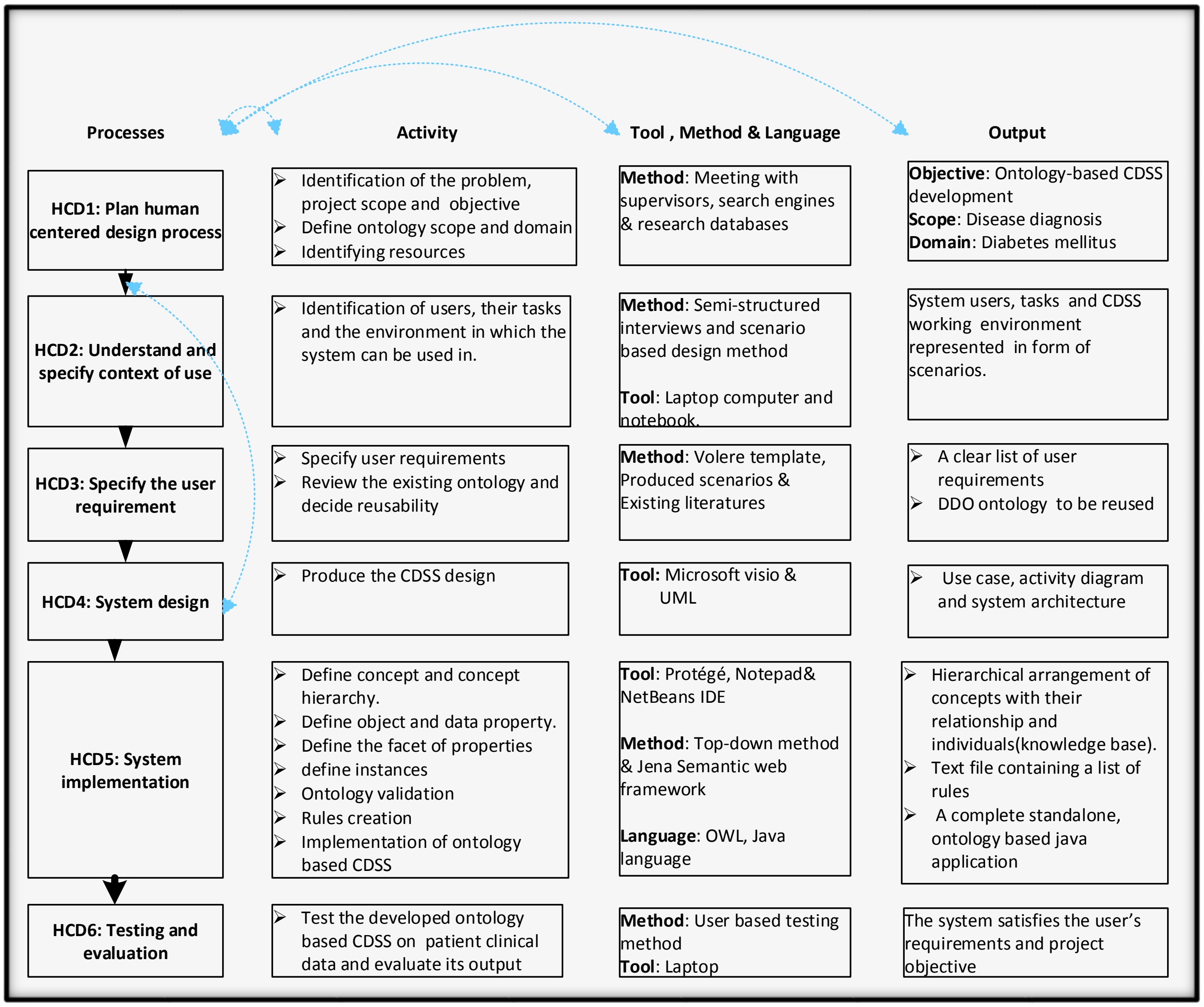
3.1 Methods

In this thesis, human centered design approach was adopted in which both qualitative and quantitative user studies have been used. In addition, semantic web based approach was adopted in which ontology and its generic tools were used to develop a reliable, extensible, reusable and sharable knowledge base for our proposed CDSS. Our knowledge base extends Diabetes Mellitus Diagnosis Ontology(DDO) **[30]** which was built based on BFO **[74]** and OGMS **[75]**. In addition, jena rule engine with forward chaining mode was used to perform reasoning on patient clinical parameters based on production rules to infer diagnosis, recommendation and decision explanation. Production rules were created based on diabetes mellitus diagnostic criteria and formally represented using jena rule syntax.

The design and development of proposed system follows a human centered design principle and activities **[76]** and the ontology building guidelines proposed by Noy Natalya et al **[77]** in which top-down approach was adopted. The ontology was developed in an iterative process that consists of 7 steps, that is, determination of ontology domain and scope, consider reusing existing ontologies, enumeration of ontology terms, defining concepts and concept hierarchies, defining object and data properties, defining the facets of the slots and creation of individuals **[77]**. These steps were mapped into human centered design activities as shown on **Figure 3.1**.

We chose human centered design approach, because it is a promising solution in producing high usable and acceptable system in both technical and commercial aspects. It has several advantages including reduction of cost spent on staff training, human errors reduction, productivity and acceptance improvement as well as reputation enhancement **[78]**. Additionally, semantic web technology was selected as a mean of developing our knowledge base as it provides a knowledge representation technologies and standards that facilitate knowledge sharing, reusing and ensure interoperability across several software agents and people as described in section 2.2.

Figure 3.1 shows a brief description of what was done throughout this chapter including processes, activities, methods/tools, and outcome. Processes at the left side represent the human centered design activities. The two middle columns represent the tasks that were performed, methods and tools that was used within each process. The right column represents the obtained results after completing each activity. Note that each step was done in an iterative process and improvement was made based on user's feedback.



Processes	Activity	Tool , Method & Language	Output
HCD1: Plan human centered design process	➢ Identification of the problem, project scope and objective ➢ Define ontology scope and domain ➢ Identifying resources	**Method**: Meeting with supervisors, search engines & research databases	**Objective**: Ontology-based CDSS development **Scope**: Disease diagnosis **Domain**: Diabetes mellitus
HCD2: Understand and specify context of use	➢ Identification of users, their tasks and the environment in which the system can be used in.	**Method**: Semi-structured interviews and scenario based design method **Tool**: Laptop computer and notebook.	System users, tasks and CDSS working environment represented in form of scenarios.
HCD3: Specify the user requirement	➢ Specify user requirements ➢ Review the existing ontology and decide reusability	**Method**: Volere template, Produced scenarios & Existing literatures	➢ A clear list of user requirements ➢ DDO ontology to be reused
HCD4: System design	➢ Produce the CDSS design	**Tool:** Microsoft visio & UML	➢ Use case, activity diagram and system architecture
HCD5: System implementation	➢ Define concept and concept hierarchy. ➢ Define object and data property. ➢ Define the facet of properties ➢ define instances ➢ Ontology validation ➢ Rules creation ➢ Implementation of ontology based CDSS	**Tool**: Protégé, Notepad& NetBeans IDE **Method**: Top-down method & Jena Semantic web framework **Language**: OWL, Java language	➢ Hierarchical arrangement of concepts with their relationship and individuals(knowledge base). ➢ Text file containing a list of rules ➢ A complete standalone, ontology based java application
HCD6: Testing and evaluation	➢ Test the developed ontology based CDSS on patient clinical data and evaluate its output	**Method**: User based testing method **Tool**: Laptop	The system satisfies the user's requirements and project objective

Figure 3.1: Solution approaches and workflow processes

3.2 HCD1: Planning human-centered design process

This is the first activity in human centered design. During this phase, several meetings with project supervisors were performed. The purpose of these meeting was to clearly identify the problem to be solved, set clear project objective and scope along with methods and tools that can be used to achieve project objective and goal. Hence, the objective was to develop an application that can assist Rwandan clinicians in their daily decision-making task and project scope is disease diagnosis and patient registration. Similarly, since this system will be connected to the knowledge base, ontology scope and domain was identified. Therefore, the scope of ontology is the same as project scope with diabetes mellitus as ontology domain.

3.3 HCD2: Understand and specify the context of use

After defining the project objective, scope and domain, the next step was to identify users, their needs, environment that our system will work in, and tasks to be performed.

3.3.1 Research field

Our project aims to develop an application that can be used in Rwanda's hospital to facilitate in diagnosis of diabetes mellitus. Better understanding of Rwanda healthcare structure and the current health information technology is vital important in identifying the gap and working environment for our proposed CDSS. This section gives a brief introduction of healthcare and health information technology in Rwanda.

3.3.1.1 Overview of healthcare in Rwanda

Rwanda is a small, landlocked, East African country with 12 064 318 populations and land area of about 26,338 square kilometers. Its populations are expected to increase by 792 persons daily in 2017 and is at higher density compared to other African countries. Its healthcare is a decentralized, multi-tied system organized based on geographical structure, that is, from village, sector, district up to national level and composed by 18 dispensaries, 16 prison dispensaries, more than 442 health centers, 48 district hospitals and 4 national referrals hospitals. In each level, there is specific people with certain level of knowledge and skills depend on the level of care being delivered. The lowest level of care is provided at village by Community Health Workers (more than 60000 well trained people that mainly focus on nutrition, HIV/AIDS prevention and pre and postnatal maternity care) while the highest level of care is provided at nation level by specialist. When a patient seeks for medical care, he/she first visit healthcare center (i.e at sector level) and when the health center fails to provide needed care, a patient is referred to the district hospital (i.e district level) or referral hospital (i.e national level) depend on the patient case.

3.3.1.2 Health information technology in Rwanda

Rwanda recognizes the importance of integrating ICT in health sector for improving health care delivery and building a great and sustainable health system infrastructure. They do so by cooperating with public, private health sector and different partners from all over the world. They all focus on building the information systems that collect, integrate, process and report patient related information with the intention of improving and managing health services at all level. Thus, makes easier for health planning and programmatic decision.

Rwanda health information technology is classified into three categories: Health management information system, mobile e-health and electronic Medical record. Mobile e-health uses the mobile phone to communicate patient related information. For instance, RapidSMS which is an SMS based tool that is designed to track pregnant women to ensure a timely and efficient follow up, report on danger signs during pregnancy so that the necessary action can be taken early and deal with before leading to complication or death and provide an alert when an emergency event occurs **[79]**.

Electronic Medical record is designed to collect and store patient related information within hospital. In Rwanda, they are currently using 2 EMRs, that is, openMRS which an open source, scalable and flexible Electronic Medical Record designed to reliably store the patient data related to HIV and tuberculosis treatment and provide a reporting tool that generates report and patient summary and OpenClinic which is an open source, web based hospital information management system that was designed to automatically manage health information within a hospital.

Health information management systems are systems designed to collect, aggregate, analyze and disseminate data related to all health programs at all health facilities. Rwanda is using DHIS-2 for such purpose. Each health facility from each level uses this application (i.e through a web browser) to submit patient related information at Rwanda warehouse (RHMIS) located at national level. This thesis mainly focus on electronic medical record (EMR). All these systems were implemented independent from each other and intended to be used for specific purpose. This imposes a big challenge on interoperability. No system can exchange and uses information from another. Additionally, these systems were designed with data collection in mind as opposed to diagnostic support. Data are firstly collected by using paper based form and data entry is performed by data manager.

3.3.2 Interview

One of the principle of human centered design is to involve user in the design and development process so that the product will meet their needs and get used in their working routine. That’s why we collaborated with nurses and physicians throughout the design and development process.

Semi structured interview with open ended questions were performed. Different physicians and nurses from both public and private health sector in Rwanda were contacted mostly via skype. In total, 6 people were interviewed. Two nurses were from private clinic while 4 medical doctors were from public hospital. Skype was chosen because we were at school (University of Agder, Norway). So, it was not possible to perform face to face interview with physician living in Rwanda. The intention of this interviews was to understand their needs, clinical diagnosis process and their reasoning procedures to come up with diagnosis. Furthermore, the information regarding existing health information systems such as the usability, the way they feel while using them and the area of improvement was gathered.

3.3.3 Data transcription, analysis and output

The interviews were performed so that the developed CDSS should meet the user’s needs and be accepted and used in clinical practice. During the interview, the interviewees was free to answer and ask questions at any time. while conducting the interview, the key points and relevant information was taken using notebook. After that, the written notes were copied into microsoft word for later analysis. Upon completion of all planned interviews, the collected data were organized and analyzed. Finally, the relevant information was taken and represented in a form of conceptual scenarios. The next section shows two scenarios: existing conceptual scenario based on the existing workflow and improved conceptual scenario for the proposed system that we have come up after reviewing and analyzing the collected information. These scenarios were discussed with interviewees, supervisors and then was refined before specifying the requirement of the system.

3.3.4 Existing conceptual scenario based on existing workflow

Mr. Karangwa Leandre is a 30-year-old farmer. In December,2016, He woke up in the morning and felt he is having a serious sickness. He went to Mwenzi health center to seek for medical care. Upon arriving to the hospital, Nurse Donatile receives him and started ask Karangwa different questions regarding his personal identification including name, father, mother, health status, insurance information, among others, symptom and signs. Donatile was writing all informations on the paper (i.e patient file) while asking Mr. Karangwa.

Nurse Donatile was having a big book of more than 300 pages containing the treatments of common health problems (i.e clinical signs and symptoms, probable causes, investigation and complication associated with each health problems). After listening to Karangwa's complains, she went through that book and wrote down a list diseases that Mr. Karangwa is probably suffering from. Diabetes was on the list of suspected disease. He gave Mr. Karangwa a small paper containing his medical record number and requests him to keep it in a safe place and should bring it next time he comes to the hospital. Furthermore, Donatile sends Mr. Karangwa to the laboratory office so that they can perform blood test in order to rule out some diseases and remains with the candidate one.

Upon arriving to the examination office, Dr. Moses warmly welcomes him to the laboratory office. Karangwa gives his file to Mr. Moses. Moses reviews the information contained in the file, draws blood from Karangwa and starts performing Random Plasma Glucose (RPG) test. Few hours later, the test result was ready and is written to Karangwa file along with diabetes as final diagnosis. He recommends that Karangwa should start diabetes medication, take healthy diets, do physical activities and invite him for next check-up after 6 months. Karangwa medical file was kept in archive where other patient records are stored.

After 6 months passed, Karangwa comes back to the hospital. He has lost the paper given by doctor which contains his medical record number and did not even remember that number. Prior to the hospital visit, Moses asks him his medical record number and Karangwa did not show it because it was lost. Moses tries to look for Karangwa patient file but he did not find it as there was a huge number of archived patient files. So, Dr. Moses decides to take a new patient file and takes again Mr. Karangwa history. He remembers that Karangwa was having diabetes and RPG test but forgets the obtained results. He performs the test again and found out that Karangwa is still having diabetes.

However, he was not sure if the obtained result is higher or less than the previous one. Karangwa was not satisfied with Mr. Moses decision because he has tried all the best to follow the doctor's advice in order to decrease his blood glucose level.

3.3.5 Improved conceptual scenario for the proposed solution

Alpha is a nurse at Hehe Health Center, she is extremely happy about the fact that, after much time, effort and motivation, her office has installed a new computer system with a clinical decision support for all of her patient.

When Mr. Karangwa, a farmer comes for his first visit, Alpha having a computer on his table and already logged in while waiting to receive the patient, starts registering Karangwa in the system by asking him different questions regarding his personal identification, address, health status, among others. After fill in the form, she started carefully listening to Karangwa's complains while at the same time select it from system's proposed list and add it to another empty list. While making selection, she accidently selects the wrong symptom and uses remove button to remove it from her choice. She does the same when examining the patient. Upon completing and reviewing her choice, she just clicks the diagnose button to get suggestion about the most probable diseases that Karangwa is suffering from. Within a second, the system suggests that Mr. Karangwa was having Diabetes along with a list of 4 exams recommended for him and requests Alpha to select one exam based on her hospital laboratory settings.

Alpha selects Fasting Plasma Glucose because 8 hours was last after Karangwa's last meal. She draws blood sample from him and start performing test. In a few hours, the test result was ready and was 150 mg/dl. Alpha inputs the result value into the system and click submit button. The system automatically asks her to select another result because one result is not enough to confirm the existence of diabetes.

Alpha selects OGT test because she wants Mr. Karangwa to go home with the final diagnosis. She gives him foods containing glucose and after 2 hours, she draws another blood samples, performs tests and obtains the glucose level of 210 mg/dl. After entering the result into the system and clicking done button, the system automatically confirms diabetes diagnosis along with the explanation on how the decision was reached and ask her to accept or reject the decision. She accepts the final diagnosis. Upon accepting the suggested decision, the system saves all the information related to Karangwa.

3.4 HCD3: Specify user requirements

After formulating the abovementioned scenarios, we have used them to specify the requirements of proposed CDSS. Requirement specifications is the document that specify the system's functions, constraints and usability based on the user's needs. It is critical important for every software, hardware or system that was designed to be interacted with user to specify the requirements before starting the design and implementation. This will help the developer to develop a product that meet the customer's needs, reduce implementation time and minimize errors.

This project adopted Volere requirement specification shell approach to specify the requirement of the system with the aim of improving the quality of health care and individual patient outcome **[78]**. All the Volere's suggested categories were not used during this project requirement specification but some of them was used based on our CDSS context of use and structure.

Additionally, existing journals (PubMed, NCBI, Springer, National institutes of Diabetes & Digestive and kidney diseases, American Diabetes Association, to name a few) was consulted to gain some medical knowledge specifically on diabetes diagnosis and to review existing diabetes ontologies so that we can reuse it instead of starting from scratch.

3.4.1 User characteristics

Our system is intended to be used by only physician having sufficient knowledge on diabetes mellitus and should have a basic computer knowledge. In this project, the terms physician, nurse, health care staff, healthcare professional and clinicians are used interchangeably and means any person who is responsible for taking care of patient, that is, examining patient, taking the patient's history, diagnosing patient, deciding treatment, advising patient on health plan, among others. He can use this application at the point of care. More specifically, he/she should be able to record patient related information including personal identification, home address, health status, among others into the system, select sign/symptom based on his observation, and provide the result of test whenever asked. He should also be able to search Patient's complete set of medical data when the patient comes back to the hospital seeking for the medical care. Furthermore, he should be able to update or delete the patient.

3.4.2 Product function

The physician ability to determine a condition that explains person's signs and symptoms can sometimes be challenging and can make clinical decision difficult. This project want to give immediate and complete guidance to the physicians during their diagnosis process based on their clinical workflow and provide registration and archiving of patient information.

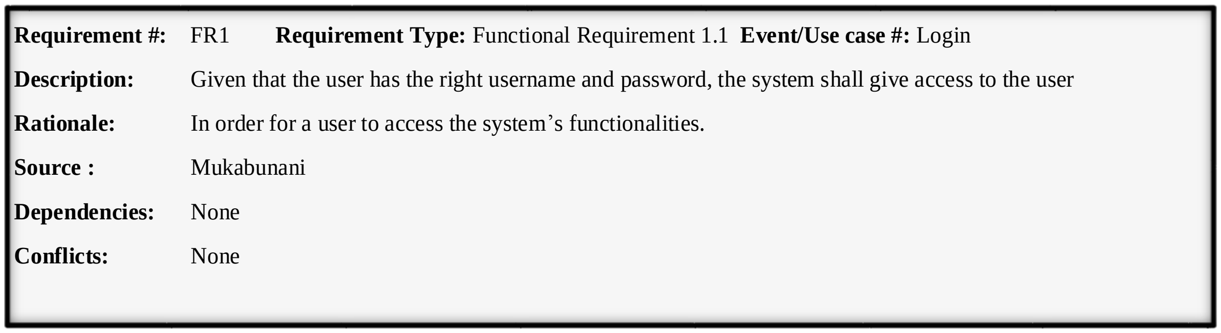
3.4.3 Constraints

The system will be constrained by the size of the knowledge base and available rules. Since it collects and store the information related to each individual patient case, the reasoning process become an issue as the number of patient gets increasing. Additionally, computer storage capacity and processing speed are also a constraint for this application. Since the medical knowledge evolve quickly and update is needed in knowledge base, the storage and processing speed will be needed to reason over those huge amounts of information.
This application is standalone application, which means that the data cannot be shared on the internet. More specifically, all the processes involved in diagnosis must be performed on the same computer.

3.4.4 Functional requirements

This section includes the requirements that specify all actions that the system is expected to do.

Functional requirement 1.1



Requirement #:	FR1 **Requirement Type:** Functional Requirement 1.1 **Event/Use case #:** Login
Description:	Given that the user has the right username and password, the system shall give access to the user
Rationale:	In order for a user to access the system's functionalities.
Source :	Mukabunani
Dependencies:	None
Conflicts:	None

Figure 3.2: Functional requirement1.1

Functional requirement 1.2

Requirement #: FR2 **Requirement Type:** Functional Requirement 1.2 **Event/Use case #:** Register a patient

Description: When a patient visit a hospital for the first time, the system shall allow the physician to take a patient history. This include patient personal identification, home address, health status, health insurance and so on.

Rationale: To better identify a patient and deliver high quality care.

Source : Mukabunani

Dependencies: FR1

Conflicts: None

Figure 3.3: Functional requirement 1.2

Functional requirement 1.3

Requirement #: FR3 **Requirement Type:** Functional Requirement 1.3 **Event/Use case #:** Select signs/symptoms

Description: After taking the patient’s history, the system shall provide a list of available symptoms/signs that a physician should select from.

Rationale: In order to get the list of probable diseases or any other support from the system.

Source : Mukabunani

Dependencies: FR1 and FR2

Conflicts: None

Figure 3.4: Functional requirement 1.3

Functional requirement 1.4

Requirement #: FR4 **Requirement Type:** Functional Requirement 1.4 **Event/Use case #:** Obtain differential diagnosis

Description: when the physician’s choice is received (i.e list of symptoms and signs that a physician has chosen), a system shall suggest a list of probable diseases that a patient is suffering from.

Rationale: In order to get the most probable diseases.

Source : Mukabunani

Dependencies: FR1, FR2 and FR3

Conflicts: None

Figure 3.5: Functional requirement 1.4

Functional requirement 1.5

Requirement #: FR5 **Requirement Type**: Functional Requirement 1.5 **Event/Use case #:** Select laboratory test

Description:	Based on differential diagnosis, the system shall suggest the test exam if applicable
Rationale:	In order for the physician to rule out some diseases and remain with the best candidate one
Source :	Mukabunani
Dependencies:	FR1, FR2, FR3 and FR4
Conflicts:	None

Figure 3.6: Functional requirement 1.5

Functional requirement 1.6

Requirement #: FR6 **Requirement Type:** Functional Requirement 1.6 **Event/Use case #:** Obtain final diagnosis

Description: Upon receipt of test result, the system shall display the final diagnosis along with explanation on how the decision was made and recommendation if applicable.

Rationale:	In order for the physician to make a treatment plan and be able to accept or reject decision
Source :	Mukabunani
Dependencies:	FR1, FR2, FR3, FR4 and FR5
Conflicts:	None

Figure 3.7: Functional requirement 1.6

Functional requirement 1.7

Requirement #: FR7 **Requirement Type:** Functional Requirement 1.7 **Event/Use case #:** Search for patient

Description: When the physician enters patient's name, the system should display the information related to the patient. This include personal identification, signs/symptoms, laboratory tests and their result, among others

Rationale: In order for a user to generate a report, make decision regarding strategic plan and measure the quality of health care provided to the patient.

Source :	Mukabunani
Dependencies:	FR1
Conflicts:	None

Figure 3.8: Functional requirement 1.7

Functional requirement 1.8

Requirement #: FR8 **Requirement Type:** Functional Requirement 1.8 **Event/Use case #: Accept/reject diagnosis**	
Description:	Based on the user decision (accept/reject), the system should definitively associate a patient related information to him or not.
Rationale:	In order for a user to keep an accurate and useful patient information
Source :	Mukabunani
Dependencies:	FR1, FR2, FR3, FR4, FR5, FR6 and FR7
Conflicts:	None

Figure 3.9: Functional requirement 1.8

3.4.5 Non-functional requirements

Usability and humanity

The system should be easy to use by a physician

The system shall be easy to learn and to remember the steps performed while carrying out the task.

The system shall give the user the way to recover from error and provide the feedback on their progress.

Hardware interfaces

The system can run on laptop /desktop computer

Software interfaces

The system will run on window/Mac/Linux operating system with JDK1.8.

3.4.6 Review of existing diabetes ontology and output

After gathering the requirements, existing literatures (PubMed, NCBI, Springer, National institutes of Diabetes & Digestive and kidney diseases, American Diabetes Association, to name a few) was consulted to gain some medical knowledge specifically on diabetes diagnosis and to review existing diabetes ontologies. One of advantage of using ontology to encode knowledge is reusability, that is, instead of representing a knowledge from scratch, we can reuse the existing one and extend it based on our application features and requirements. Moreover, that knowledge should contain standardized concepts. In other words, that ontology should obtains its concepts from standard biomedical ontologies such as SCT, UMLS, LOINC, Gene Ontology(GO), Human Disease ontology(DOID), RxNorm, BFO, and OGMS to ensure interoperability between people and software agents **[30]**. In this regards, our intention was to find out the ontology that can satisfy the above-mentioned user requirements and conditions. More specifically, the one that can answer physician's questions such as what are signs/symptoms associated to this diabetes mellitus? what are diseases affecting a patient given sign/symptom? what are the laboratory tests that are needed for screening and diagnosing diabetes mellitus. As results, DDO: a diabetes mellitus diagnosis ontology has been selected **[30]**.

The reason behind choosing this ontology: firstly, it accurately covers every aspect of diabetes mellitus related to diagnosis (i.e., its clinical manifestation, diagnosis, treatment, course of development and laboratory test). Secondly, it is built based on top level ontologies (i.e BFO and OGMS) and follows the principle of the OBO Foundry consortium (e.g., open, common format, etc.). Lastly, its terms were extracted from existing standard biomedical ontologies such as SCT, DOID, Symptom ontology(SYMP), and so on). The part of their ontology in relation with OGMS and BFO is shown on **Figure 3.10**.

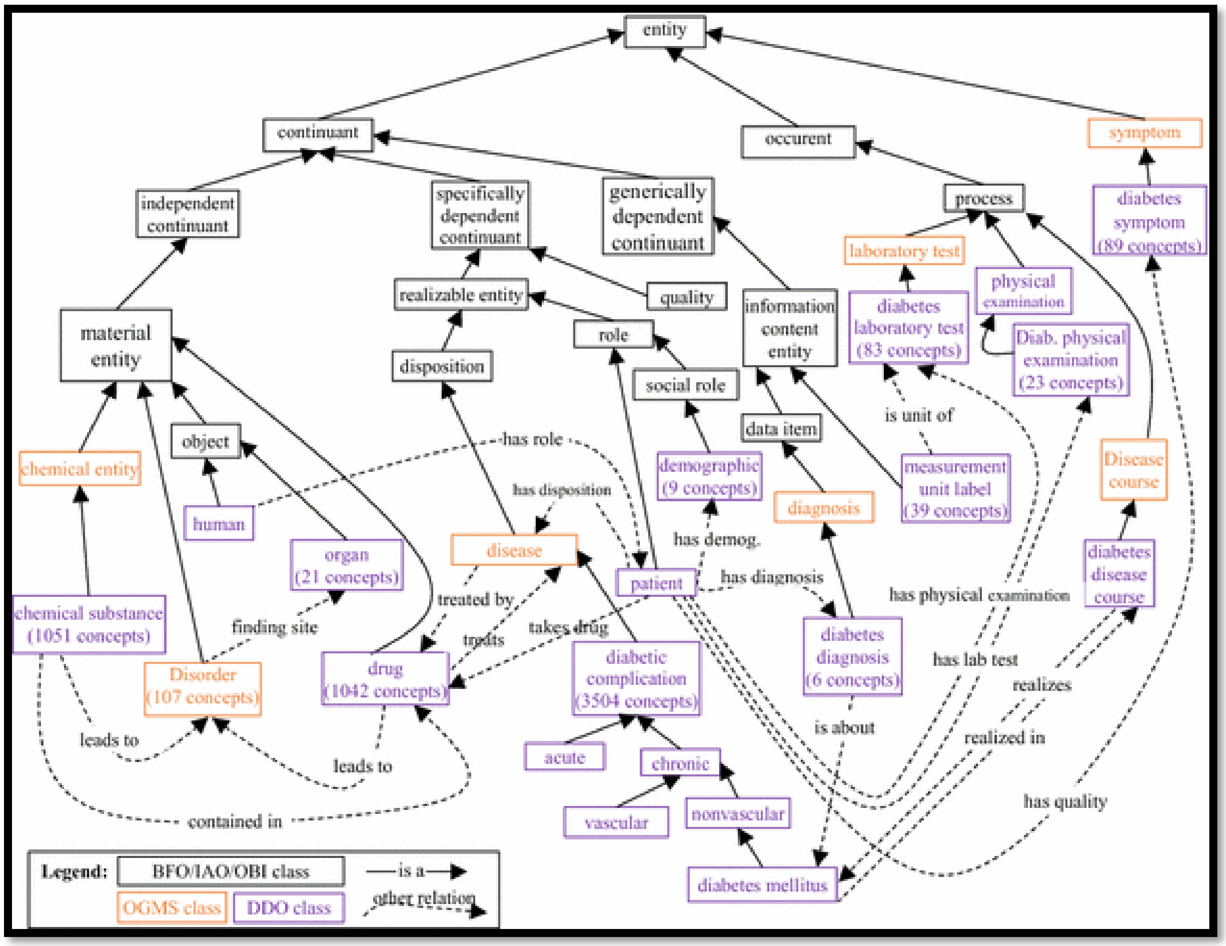


Figure 3.10: A portion of DDO in relation with OGMS and BFO [30]

3.5 System design

Based on the user requirements, the system was designed and its interaction with intended users (i.e physician) was shown. UML was used as modeling language of our proposed system. The reason behind choosing UML is that it is based on object oriented technology, flexibility, standardized for software design and helps us to clearly specify, visualize, document and communicate the behavior of our CDSS to the users **[81]**. All UML diagram was not used. Only use case diagram, activity diagram was selected. Microsoft visio professional 2016 was used to create the above-mentioned diagrams. It was selected because of its capability of creating professional, graphical diagrams that can be easily shared and accessed by various entities involved in project. Additionally, it offers a wide range of built-in shapes, objects stencils and allows us to import and customize our own shapes **[82]**.

3.5.1 Use case for proposed CDSS

Use case diagram is used to model high-level functions, scope of system and show relationship with its intended users referred as actors so that users can understand the features of the system before implementation and serves as foundation for other diagrams such as activity diagram **[83]**) as shown in **Figure 3.11**.

It was designed based functional requirements of the system and other informations described in section 3.4. The intention was to create and communicate the proposed system’s model to the users so that we can receive feedback and redesign it before we start real implementation. It is made of four main components, that is, actor, system, use case and relationship. Actors are defined as people or any external entity interacting with the system and are shown on the left side of **Figure 3.11** (i.e., physician). A system is shown in the middle and is represented as rectangle to show our system’s boundary. Inside the system there is use cases which are the functions that is carried out by physician and line between physician and use case represents relationship between them.

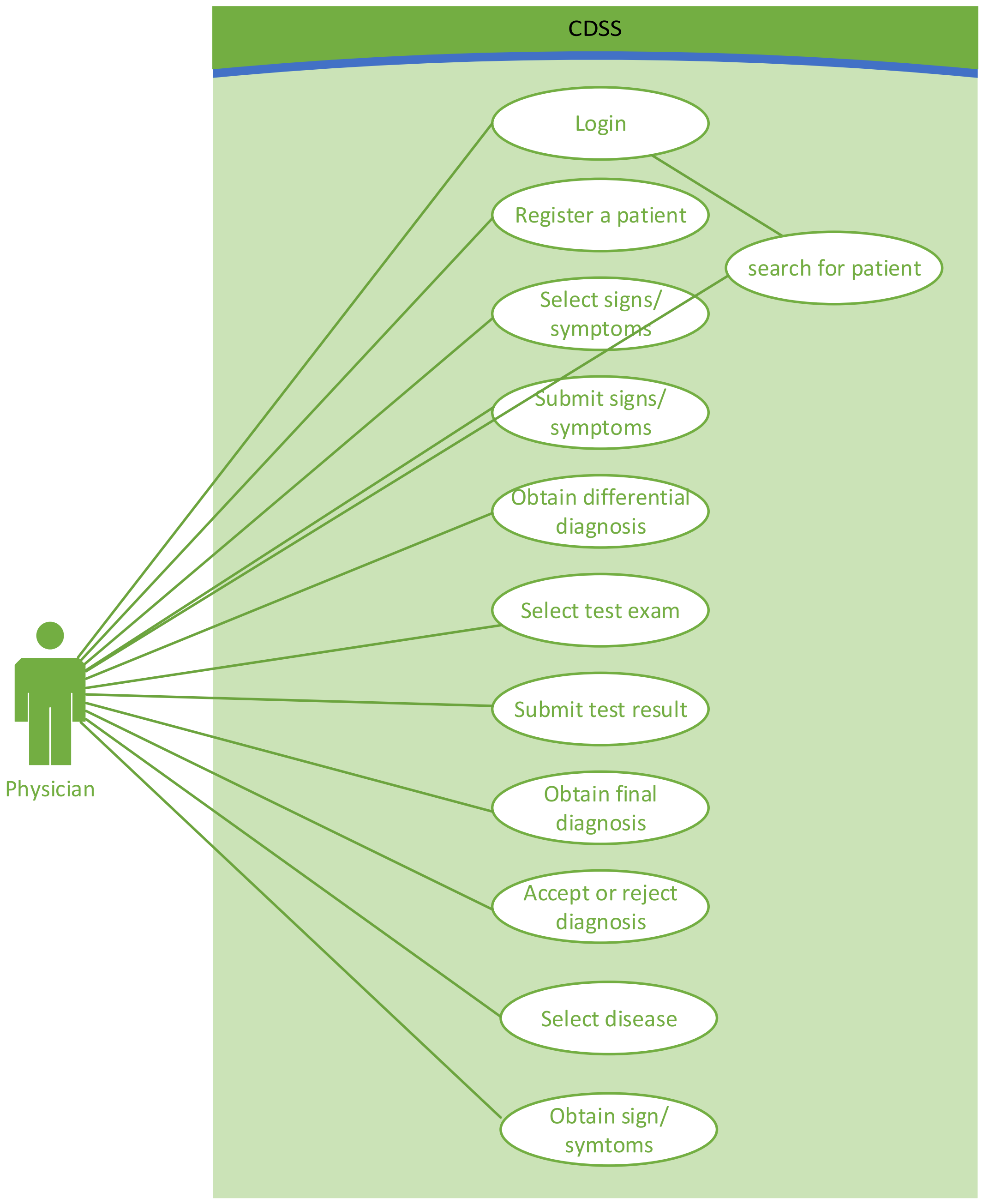


Figure 3.11: Use case diagram of proposed system

3.5.2 Activity diagram

Based on use case diagram, the activity diagram is presented to describe a workflow involved in use case diagram along with all decisions and branching logics as shown in **Figure 3.12.** The process is shown in rectangle while diamond shape represents decision that should be made either by physician or system.

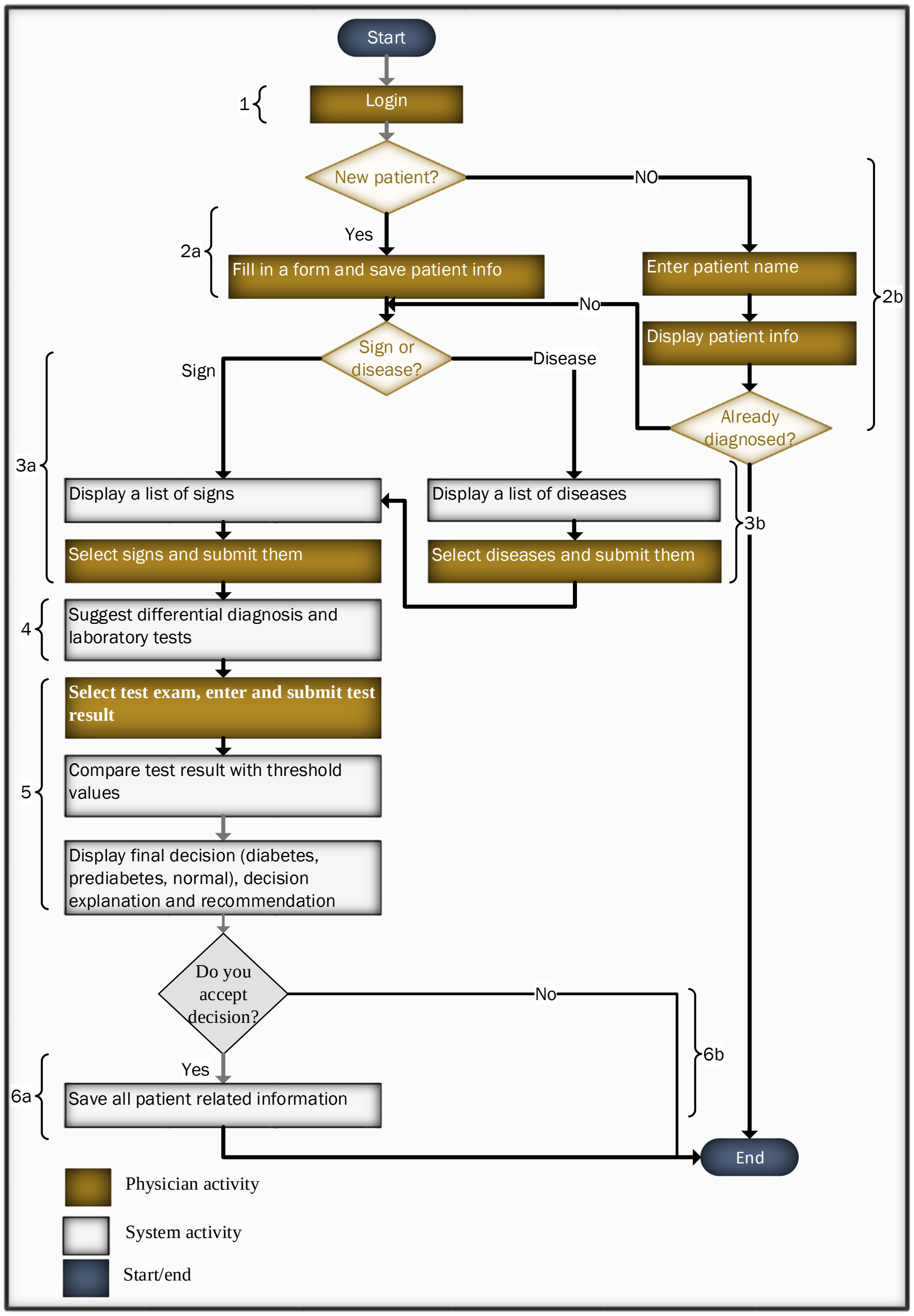

Start
1
Login
New patient?
NO
Yes
2a
Fill in a form and save patient info
Enter patient name
No
2b
Display patient info
Sign
Sign or disease?
Disease
Already diagnosed?
3a
Display a list of signs
Display a list of diseases
3b
Select signs and submit them
Select diseases and submit them
4
Suggest differential diagnosis and laboratory tests
Select test exam, enter and submit test result
5
Compare test result with threshold values
Display final decision (diabetes, prediabetes, normal), decision explanation and recommendation
Do you accept decision?
No
6b
Yes
6a
Save all patient related information
End
Physician activity
System activity
Start/end

Figure 3.12: Activity diagram

Table 3-1: Description of activity diagram

Activity no	Description
1	Before accessing the system's functionalities, physician must login into the system by providing the right credentials (i.e username and password). The system will check the username and password against the information stored in database. When a match is found, the access will be given.
2	After login, a physician will be given different options. For instance, physician can select diagnose option when a patient visits a hospital and could specify if the patient is new or if he already exist in the system. If physician selects new patient, a patient form will be displayed and he can start fill in the form. The information that can be filled out include patient personal details, allergies, blood group and so on. After completing the form, he/she should save patient information. In this case, system will formalize patient's information in ontology language and store it in the knowledge base. If a patient already exists in the system, the physician will enter a patient's name and obtain the information about the patient.
3	In this step, this is where the real diagnosis task starts. A physician will be given two options: start with sign/symptom or start with disease. If the first option is selected, the system will display a list of all signs/symptoms that is represented in the knowledge base and he/she is free to select them based on his observation and patient complains. His choice will be added to an empty list and he can remove the wrong selected items from the list at any time. He could then submit his/her selections after reviewing and refining them. Similarly, when a physician select suspect some diseases from a patient, he/she can select the second option and the system will display a list of all diseases (in this system we do have only diabetes mellitus) represented in knowledge base. In this case, physician will select diseases and submit them. The system will display a list of signs/symptoms associated to these selected diseases and then physician can continue with step4.

4	Upon receival of signs/symptoms associated to the patient, the system will formalize this information and associate them to the intended patient based on the property already defined in ontology and run logical inference to derive differential diagnosis and give it back to the physician and suggest examination test that are needed in order to confirm diagnosis.
5	A physician should select a test exam based on their clinical setting and provides the result of examination. After submitting result, the system will run logical inference and compare this result with threshold value presented in production rule and generate the final diagnosis along with explanation on how decision was made. If there is any recommendation associated with this decision, the system will also generate it.
6	Physician does not have to rely on the result of CDSS, sometimes, he/she should agree with CDSS or not. That's why at this step, he/she will be given an option to agree or accept the result. When the decision is accepted, all information related to a patient will be bound to him/her and saved into knowledge base for later retrieval or usage.

3.6 HCD5: System implementation

After designing and evaluating the design, the system was implemented. This section describes the architecture of the system, tool and language used for development of both application and its knowledge base. The system implementation and ontology development is described in detail along with the screenshots and codes. After implementation, the system was tested and evaluated with clinical patient information.

3.6.1 Proposed system architecture

Our systems consist of several components that works together to assist the clinician when making the diagnosis of diabetes mellitus. It was designed based on the human processes and reasoning when making the diagnosis. This is a multistep process that a physician goes through to come up with a right diagnosis. For example, taking patient history, physical examination, test result, and derive conclusion about disease affecting the patient after analyzing those patient data. The proposed components include database, CDSS user interface, inference engine, data loading and saving engine, search engine and knowledge base as shown on **Figure 3.13**

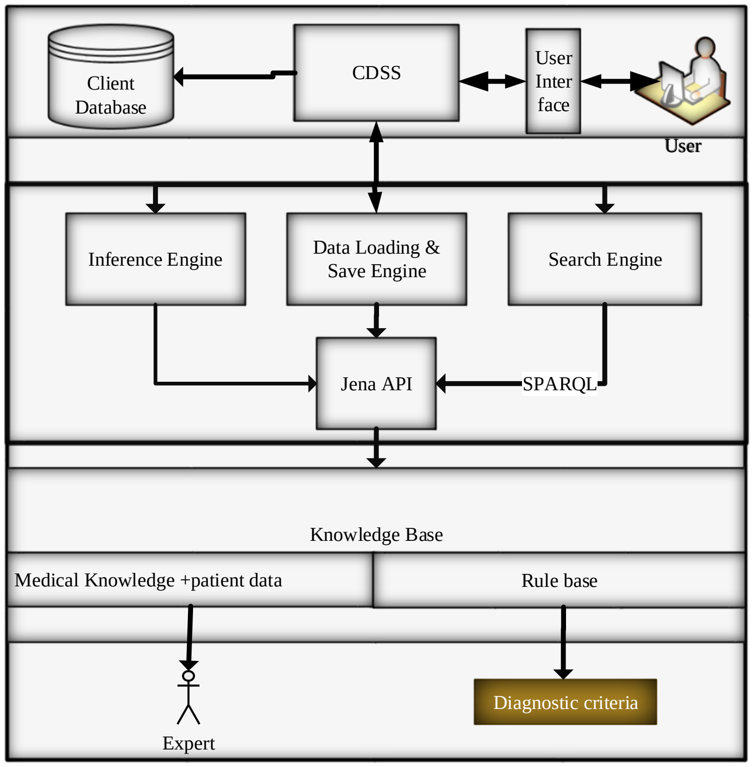


Figure 3.13: Overview of the system architecture

Database: the database is one of the element of this system. Its role is to store the username and password of intended user of the system. The administrative staff could create the user's credentials and give them to the users so that they could login and get access to the system's functionalities. This database was used just to separate medical knowledge base and staff informations. Thus, allowing this knowledge base to be shared and reused by other people or health organizations.

CDSS user interface: User interface serves as a bridge between physician and CDSS application. Physician will interact with the application through this user interface. He/she should be given a series of interfaces based on the task that need to be performed. More specifically, a physician will use the user interface to input patient data and will receive the result through it.

Data loading and saving engine: this engine uses jena API and loads an ontology file and rule file in memory so that other system components can access and manipulate data stored in these files. It is also used to save or update the ontology file when some changes are made or when patient information is received. For instance, this engine helps the physician to save signs/symptoms and test result associated with a patient. More specifically, this engine takes the inputs from the physician, saves and associates them to the intended patient in the knowledge base. To do so, the name of patient should be given to the engine so that it can identify him/her from another existing patient.

Inference engine: Inference engine is the central component of proposed CDSS. It is where all reasoning task is performed and the result is made available to jena API. Inference engine requires access to ontology file containing signs/symptoms, patient data, diseases description, and so on and the rule file. Then, it applies rules to the facts stored in ontology in order to derive diagnosis or any other kind of decision presented in the rule decision. The result of inference engine can be stored in memory or saved in the ontology for further usage or for answering user queries. We have selected jena rule engine that works in forward chaining mode to reason with individual patient clinical data.

Search engine: search engine is responsible for answering all user's requests. More specifically, this engine uses SPARQL query to retrieve information from knowledge base. It should have access to the knowledge base and use pattern matching to produce the result.

Knowledge base: knowledge base is another main component of proposed CDSS. It is a big database that contain all information related to diabetes diagnosis. These include diseases (i.e Diabetes mellitus in this case), its physical manifestation, laboratory test, properties, restriction, patient and patient related information. In this project rules are stored in text files and are formally presented based on diabetes mellitus diagnostic criteria while knowledge related to diabetes mellitus and patients are stored in Owl file. Owl file can be in different format such as turtle, RDF/XML and so on.

Figure 3.14 shows the internal working structure of our CDSS, that is, the process that is performed to satisfy user's request. When the user first interacts with CDSS application, CDSS application consults Jena API. The first thing that Jena API does is to load the ontology file. The rule file is loaded based on the physician's request. For instance, when a physician requests to save patient information, only ontology file will be loaded and updated accordingly. When the physician requests for patient diagnosis, both ontology and rule file will be loaded and transferred to the inference engine. Then inference engine performs reasoning and derive the diagnosis, inference explanation, etc. In this case, SPARQL query can be used to retrieve the inferred information and give them back to the physician.

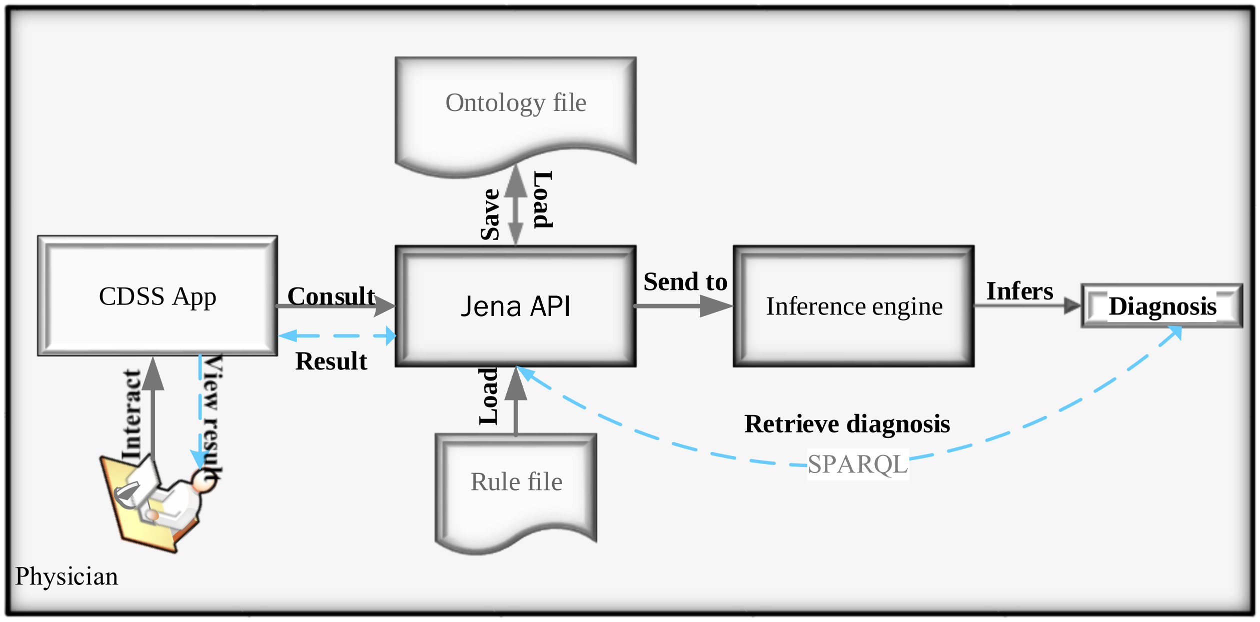


Figure 3.14: System's component interactions

Figure 3.15 shows the workflow process of proposed CDSS in term of programming logic. Input and output is received through user interface and reasoning process is performed in so called inference model. Patient demographic information, insurance status and health status will be collected through registration form and formally represented into ontology file. Moreover, patient vital signs and test result will be inputted into the system through user interface. Upon receiving these information, SPARQL update will be used to formally represent and associate them to intended patient. After that, reasoner will be associated to the rule file and bound to the ontology file to create inference model. The result returned by inference Model will be queried by using SPARQL select query and give them back to the user through user interface.

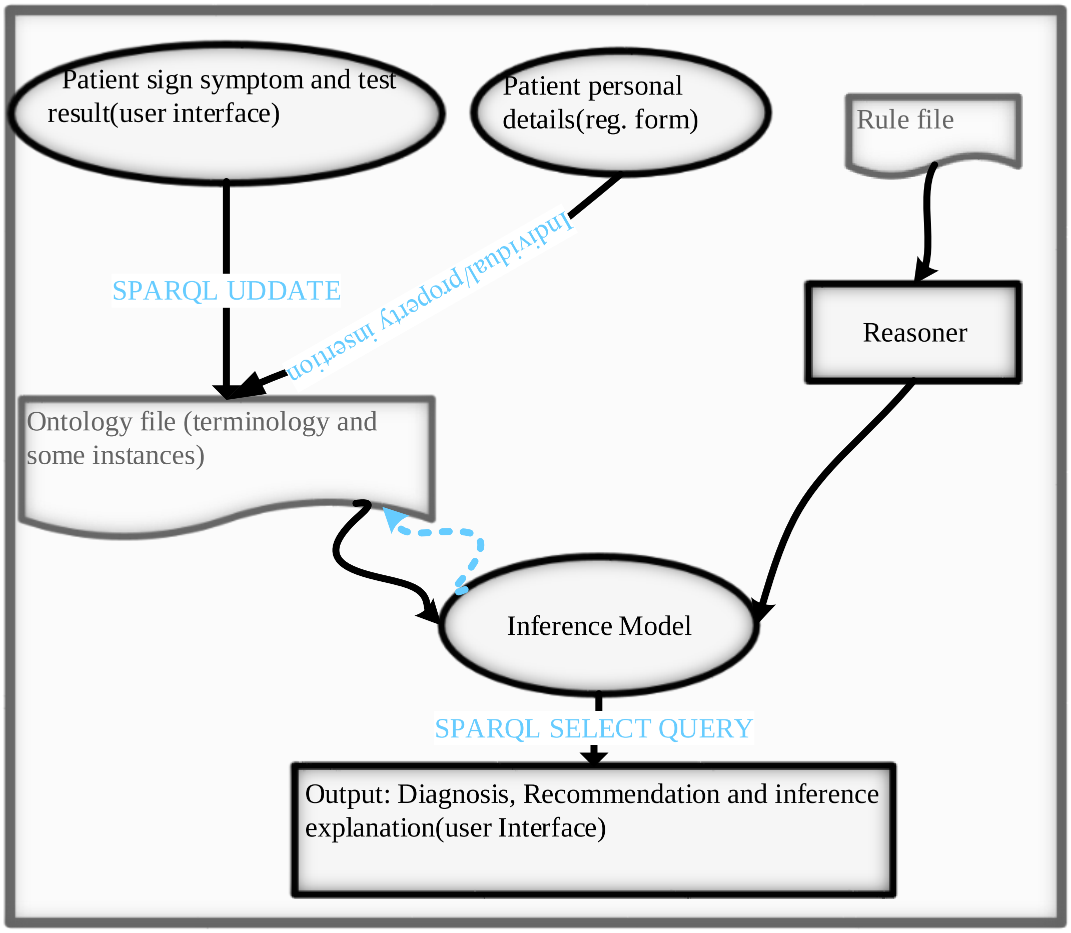

Patient sign symptom and test result(user interface)
Patient personal details(reg. form)
Rule file
SPARQL UDDATE
Individual/property insertion
Reasoner
Ontology file (terminology and some instances)
Inference Model
SPARQL SELECT QUERY
Output: Diagnosis, Recommendation and inference explanation(user Interface)

Figure 3.15: System's workflow architecture

3.6.2 Ontology development tool and language

There are various standards language for building an ontology, however, their features differ from each other. We have chosen the OWL language as it is the most recent developed ontology language recommended by W3C for building ontologies. The OWL language has advanced features that allow users to create concepts (class), organized them into hierarchy, define the binary relationship between concepts and create instances(individuals) as well as impose different restrictions on the property. Additionally, OWL offers inference capability which acquires new knowledge from the ontology (automatic subsumption or classification) and consistency checking between hierarchy.
Similarly, many ontology development tools have been proposed (webOnto, OntoStudio, onto Edit, WebODE, Protégé ...,) and they differ depending on type of application being developed, W3C standards support, API support and the way data is represented. Some of them are used for building PHP and web- based ontologies and are commercial; for example, POWL while other can be used for building the ontologies which are large and complex; for example, OntoStudio **[84]** . Protégé 5.0.0 desktop was used as ontology editing tool in this thesis. The reason behind choosing this tool:

- ✓ It is an open source
- ✓ It is easy to use
- ✓ It is standalone, that is, we do not need an internet connection to use it.
- ✓ It has user friendly interface
- ✓ It is suitable for academic ontology
- ✓ It fully supports OWL ontologies and
- ✓ It offers plugins for graph visualization of ontology and has a large user community.

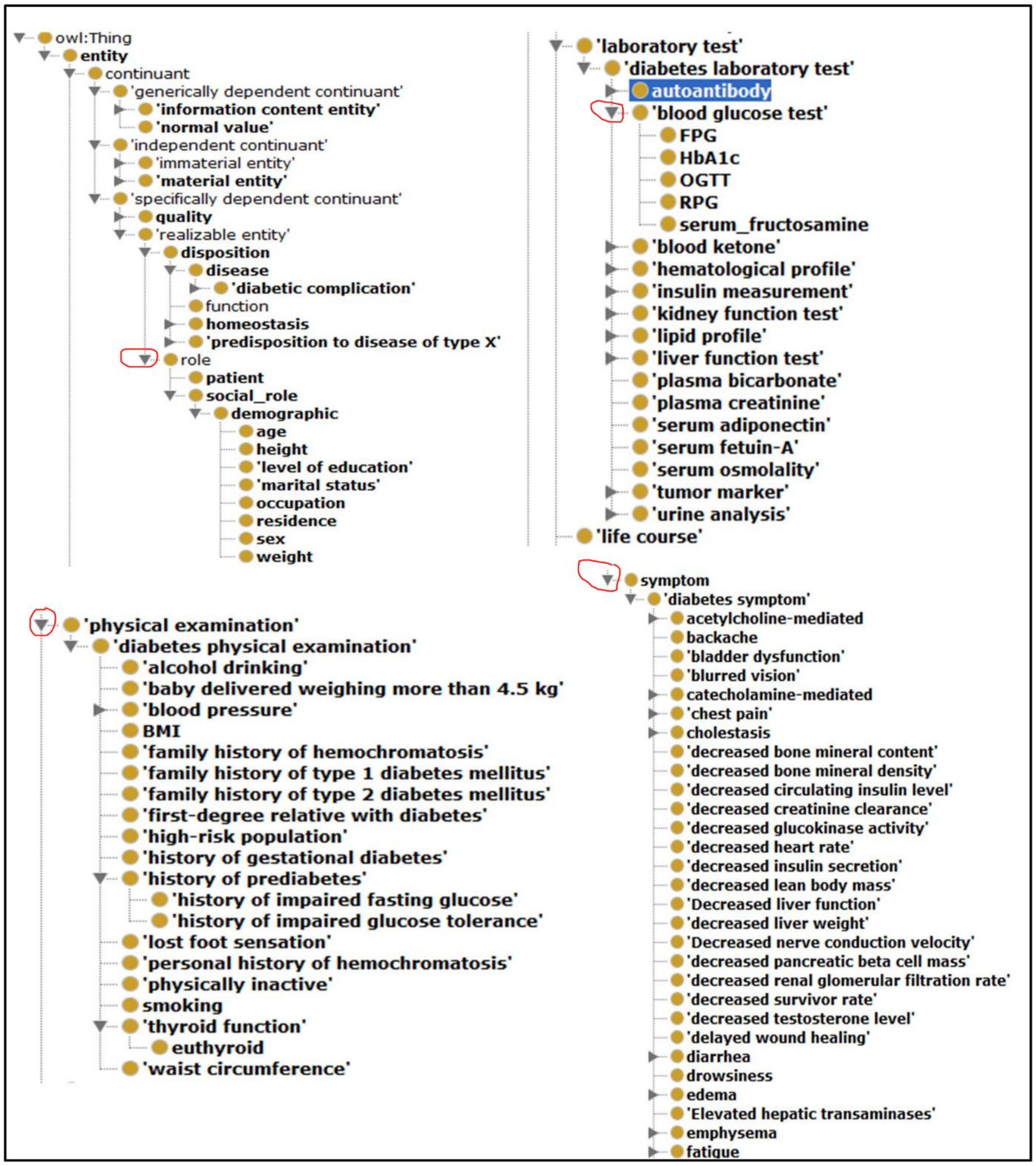
3.6.3 Ontology-based application development tool and language

The ontology-based CDSS was implemented by using java language and jena API. The reason behind choosing jena is that it provides different classes and interfaces that allow us to access and manipulate directly OWL ontologies **[85]**. More specifically, jena provides a class to create a model. After creating model, jena can parse an OWL file and store it in memory or in persistent storage. At this moment, we should access classes, properties and individuals created in ontology, manipulate them as well creating the new one and save it to the model. Besides, jena provides inference engine that supports OWL inference and is available for general use. This means it can reason with user defined rule and generate suitable decision. It also has an ARQ engine that help us to use SPARQL query and retrieve information stored in inference model.

This API can be downloaded at (**http://jena.sourceforge.net/),** put it into your java development environment and start developing a semantic web application with java. In this project, NetBeans was used as java development environment. There are other environments that are available such as eclipse but we have chosen NetBeans because we are familiar with it and offer several features that was needed such as graphic designer and so on. Any other java environment can work with jena and the choice depends on your needs and familiarity with the IDE.

3.6.4 Knowledge base development

As described in section 3.4.6, our ontology reuses DDO ontology which is an extension of BFO and OGMS. It was developed by using both top-down (development process starts with general concepts up to more specific one) and bottom-up (the development process starts with more specific concepts up to general concept) approach. The extension of this ontology adopted the top-down approach. We did not use all concepts represented in this ontology. We focused on concepts related to physical examinations, symptoms, role, diagnosis, laboratory test (e.g., blood glucose test), etc and the interested concepts are highlighted in red including their subclasses as shown in **Figure 3.16**.

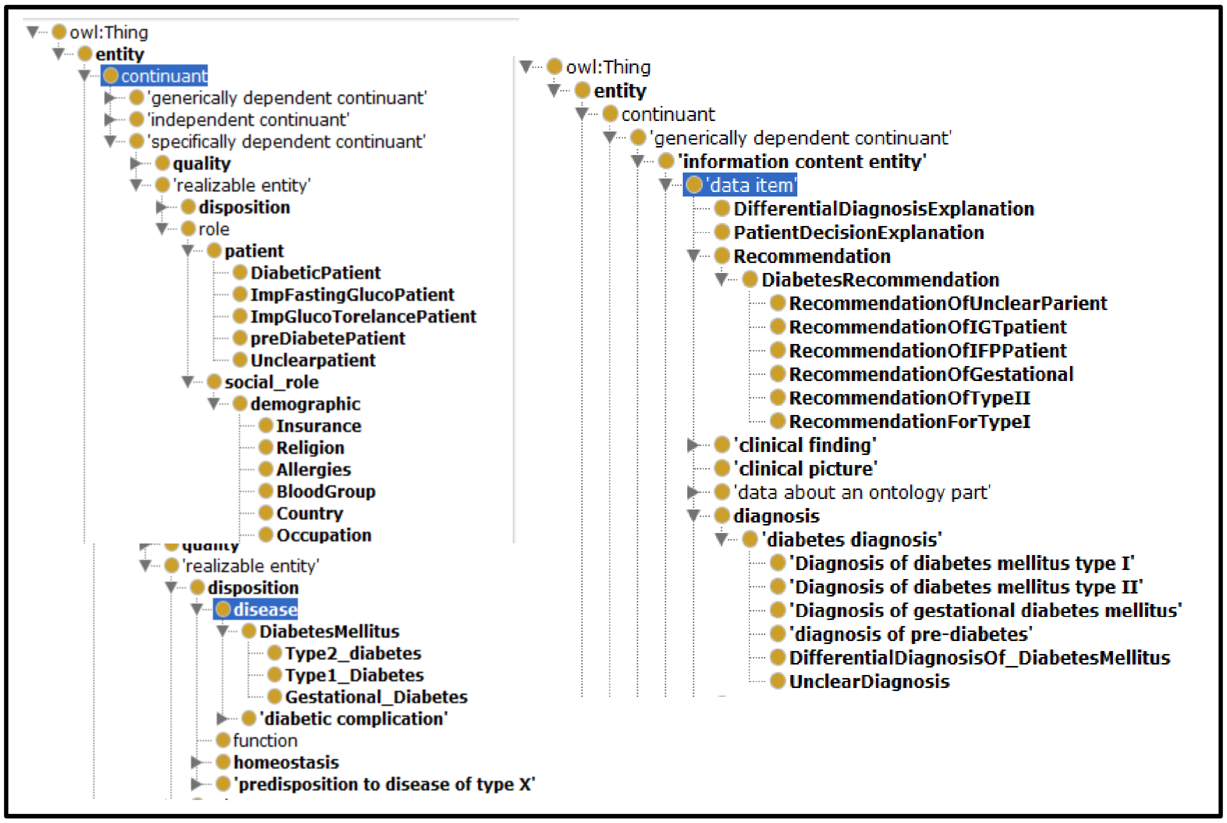


```
owl:Thing
  entity
    continuant
      'generically dependent continuant'
        'information content entity'
        'normal value'
      'independent continuant'
        'immaterial entity'
        'material entity'
      'specifically dependent continuant'
        quality
        'realizable entity'
          disposition
            disease
              'diabetic complication'
            function
            homeostasis
            'predisposition to disease of type X'
          role
            patient
            social_role
              demographic
                age
                height
                'level of education'
                'marital status'
                occupation
                residence
                sex
                weight

'physical examination'
  'diabetes physical examination'
    'alcohol drinking'
    'baby delivered weighing more than 4.5 kg'
    'blood pressure'
    BMI
    'family history of hemochromatosis'
    'family history of type 1 diabetes mellitus'
    'family history of type 2 diabetes mellitus'
    'first-degree relative with diabetes'
    'high-risk population'
    'history of gestational diabetes'
    'history of prediabetes'
      'history of impaired fasting glucose'
      'history of impaired glucose tolerance'
    'lost foot sensation'
    'personal history of hemochromatosis'
    'physically inactive'
    smoking
    'thyroid function'
      euthyroid
    'waist circumference'

'laboratory test'
  'diabetes laboratory test'
    autoantibody
    'blood glucose test'
      FPG
      HbA1c
      OGTT
      RPG
      serum_fructosamine
    'blood ketone'
    'hematological profile'
    'insulin measurement'
    'kidney function test'
    'lipid profile'
    'liver function test'
    'plasma bicarbonate'
    'plasma creatinine'
    'serum adiponectin'
    'serum fetuin-A'
    'serum osmolality'
    'tumor marker'
    'urine analysis'
'life course'

symptom
  'diabetes symptom'
    acetylcholine-mediated
    backache
    'bladder dysfunction'
    'blurred vision'
    catecholamine-mediated
    'chest pain'
    cholestasis
    'decreased bone mineral content'
    'decreased bone mineral density'
    'decreased circulating insulin level'
    'decreased creatinine clearance'
    'decreased glucokinase activity'
    'decreased heart rate'
    'decreased insulin secretion'
    'decreased lean body mass'
    'Decreased liver function'
    'decreased liver weight'
    'Decreased nerve conduction velocity'
    'decreased pancreatic beta cell mass'
    'decreased renal glomerular filtration rate'
    'decreased survivor rate'
    'decreased testosterone level'
    'delayed wound healing'
    diarrhea
    drowsiness
    edema
    'Elevated hepatic transaminases'
    emphysema
    fatigue
```

Figure 3.16: Portion of DDO ontology

Defining concept and concept hierarchy

The concepts represented in DDO were not enough to accomplish our project objective. Some of them were removed and others were added based on our needs. For instance, in DDO, age and sex are defined as class. But in our new ontology, they are not represented as class instead they are used as data type property. Initially, all patients are classified as instance of patient class and we want to classify them based on their diagnosis after reasoning process. That's why, we defined the subclasses of patient class. We wanted to explain to our users how conclusion was reached so that they can trust the system's decision and provides the physician with recommendation on how patient can be followed up. In this regard, we added PatientDecisonExplanation and Recommendation classes. **Figure 3.17** shows some new concepts added in DDO ontology.



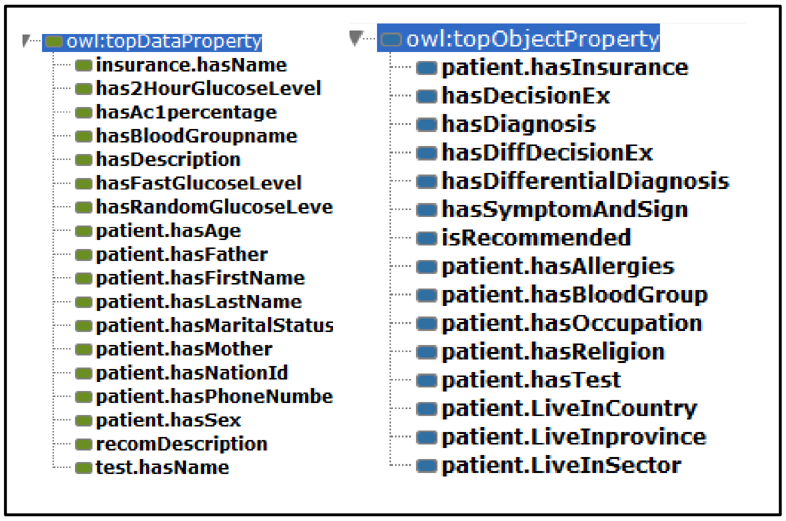
```
owl:Thing
  entity
    continuant
      'generically dependent continuant'
      'independent continuant'
      'specifically dependent continuant'
        quality
        'realizable entity'
          disposition
          role
            patient
              DiabeticPatient
              ImpFastingGlucoPatient
              ImpGlucoTorelancePatient
              preDiabetePatient
              Unclearpatient
            social_role
              demographic
                Insurance
                Religion
                Allergies
                BloodGroup
                Country
                Occupation
        quality
        'realizable entity'
          disposition
            disease
              DiabetesMellitus
                Type2_diabetes
                Type1_Diabetes
                Gestational_Diabetes
              'diabetic complication'
            function
            homeostasis
            'predisposition to disease of type X'
```

```
owl:Thing
  entity
    continuant
      'generically dependent continuant'
        'information content entity'
          'data item'
            DifferentialDiagnosisExplanation
            PatientDecisionExplanation
            Recommendation
              DiabetesRecommendation
                RecommendationOfUnclearParient
                RecommendationOfIGTpatient
                RecommendationOfIFPPatient
                RecommendationOfGestational
                RecommendationOfTypeII
                RecommendationForTypeI
            'clinical finding'
            'clinical picture'
            'data about an ontology part'
            diagnosis
              'diabetes diagnosis'
                'Diagnosis of diabetes mellitus type I'
                'Diagnosis of diabetes mellitus type II'
                'Diagnosis of gestational diabetes mellitus'
                'diagnosis of pre-diabetes'
                DifferentialDiagnosisOf_DiabetesMellitus
                UnclearDiagnosis
```

Figure 3.17: Part of some concepts added in new ontology

3.6.4.1 Defining data and object property

Property defines the binary relationship between individuals or between individuals to xml data type such as string integer, integer, to name a few. OWL has two types of properties, that is, object property and datatype property. The object property (owl: ObjectProperty) relates individuals from two classes while Datatype property (Owl: DatatypeProperty) is used to relate an individual to a data value. Our ontology consists of several object and data properties and most of them are owned by patient. **Figure 3.18** shows an example of properties defined in our ontology. Data properties are shown at the left while object properties are shown at the right.



owl:topDataProperty	owl:topObjectProperty
insurance.hasName	patient.hasInsurance
has2HourGlucoseLevel	hasDecisionEx
hasAc1percentage	hasDiagnosis
hasBloodGroupname	hasDiffDecisionEx
hasDescription	hasDifferentialDiagnosis
hasFastGlucoseLevel	hasSymptomAndSign
hasRandomGlucoseLeve	isRecommended
patient.hasAge	patient.hasAllergies
patient.hasFather	patient.hasBloodGroup
patient.hasFirstName	patient.hasOccupation
patient.hasLastName	patient.hasReligion
patient.hasMaritalStatus	patient.hasTest
patient.hasMother	patient.LiveInCountry
patient.hasNationId	patient.LiveInprovince
patient.hasPhoneNumbe	patient.LiveInSector
patient.hasSex	
recomDescription	
test.hasName	

Figure 3.18: Datatype and object property

Starting from the left side of the **Figure 3.18**, there several data properties. These properties are used to link the individual of class to a data value. For instance, each blood glucose test is associated with blood glucose level. In order to related test to its blood glucose level, data type property was created. Our system will only focus on blood glucose test (i.e only 4 tests will be used) which means that 4 data type properties are needed, that is, has2HourGlucoseLevel for IGT test, hasAc1percentage for hemoglobin test, hasFastGlucoseLevel for FPG test and hasRandomGlucoseLevel for RPG test. Moreover, some patient information will be added as data type property, for instance, patient name, age, sex, father and so on.

On the right side of **Figure 3.18**, there are several object properties. These properties were created to related individuals of two classes. For instance, a patient should have a test, symptom, diagnosis and so on. To relate instance of patient class to another class's instance, different object properties were created. For example, **patient.hasTest** is object property owns by patient and is used to relate him/her to an instance of diabetes test class. **hasDiagnosis** is another property owns by patient and is used to relate a patient to an instance of diabetes diagnosis class. **hasSymptomAndSign** is another object property owns by a patient and is used to relate a patient to an instance of diabetes sign/symptom class**.**

3.6.4.2 Defining facet, range and domain of the property

Facets are attribute of a property and are used to impose restrictions on property value, including cardinality restriction (minimum and maximum cardinality restriction), quantifier restriction (existential and universal) and value type for property (e.g, integer, string, date, etc). The latter was used to define the facet of our properties. In addition to the restriction, a property should have a domain and range. For instance, **hasDiagnosis** property has domain of patient and range of DiabetetDiagnosis class. **Figure 3.19** shows value type restriction of patient.hasFirstName **(**the first name of a patient will be treated as string) as String and the domain and range of hasDiagnosis (any individual having hasDiagnosis as predicate will be inferred as instance of patient class and his/her diagnosis should come from diabetesDiagnosis class).

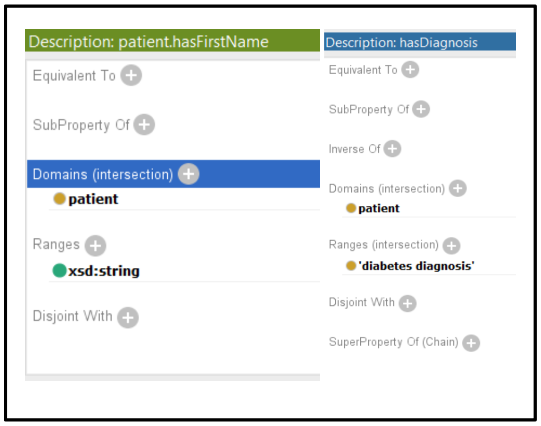
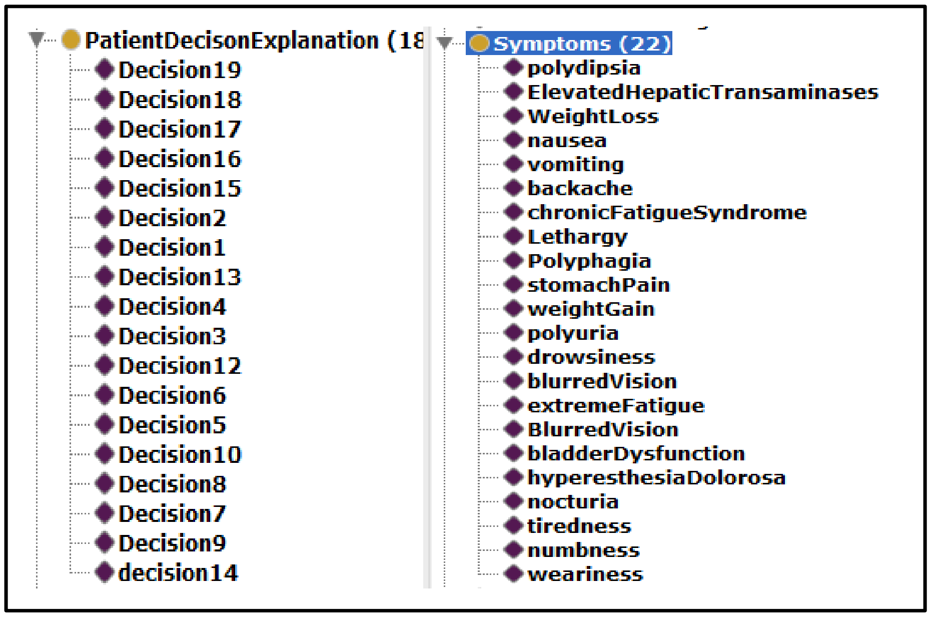


Figure 3.19: Facet, domain and range of properties

3.6.4.3 Creation of individuals

Instances are called individuals in description logic and are defined as concrete objects of classes. They are associated to an owl class and are referred as member of that class. The instances were instantiated in two ways. In the first way, they were created in protégé. In the second one, they were instantiated based on patient conditions and characteristics, thus, facilitates personalized diagnosis and treatment. Individuals created by using first case were used when creating the rules (e.g instances from PatientDecisonExplanation, diabetesDiagnosis and Recommendation class) and when providing a list of signs/symptoms or diseases to the user while others will be instantiated when formalizing patient data (i.e the information that is entered or selected by a physician). **Figure 3.20** shows an example of instances created in ontology.


PatientDecisonExplanation (18
Decision19
Decision18
Decision17
Decision16
Decision15
Decision2
Decision1
Decision13
Decision4
Decision3
Decision12
Decision6
Decision5
Decision10
Decision8
Decision7
Decision9
decision14

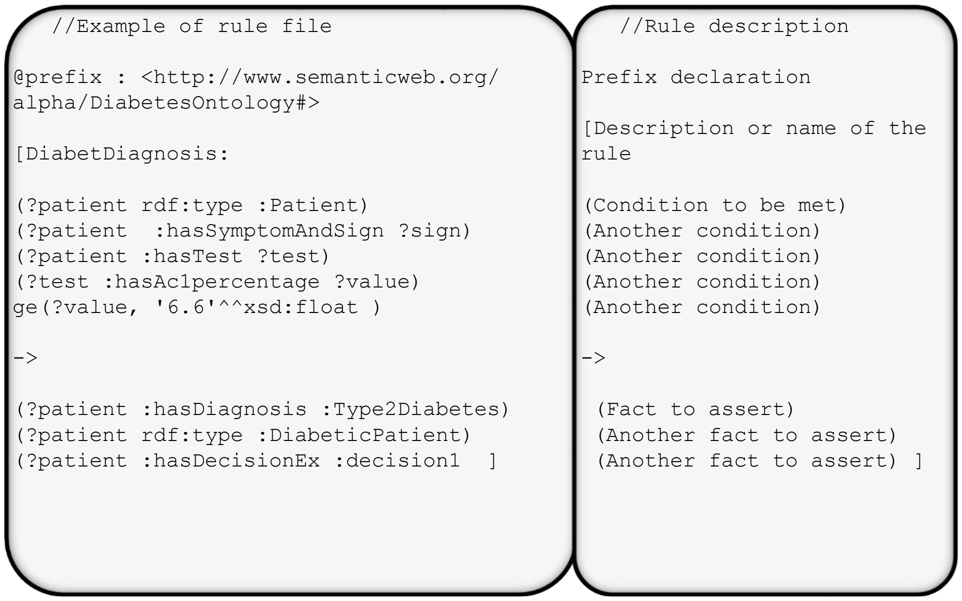
Symptoms (22)
polydipsia
ElevatedHepaticTransaminases
WeightLoss
nausea
vomiting
backache
chronicFatigueSyndrome
Lethargy
Polyphagia
stomachPain
weightGain
polyuria
drowsiness
blurredVision
extremeFatigue
BlurredVision
bladderDysfunction
hyperesthesiaDolorosa
nocturia
tiredness
numbness
weariness

Figure 3.20: Example of instances in ontology

3.6.5 Construction of decision rules

As described in section 2.2.6, jena rule can be written in a text file or in an application and can be used to infer diagnosis, recommendation and inference explanation based on an individual patient case. We have chosen to write rules in text file so that these rules can be easily reused in different applications. Based on the information represented in **Table 2-1** in section **2.1.2** and other information described in section **2.1,** we created 19 rules using jena abstract syntax shown on **Error! Reference source not found.** and forward chaining of inference ("->") was used. These rules use concepts/axioms defined in our ontology. The reason behind choosing forward chaining strategy is that we wanted all statements inferred after the rule execution to be explicitly generated and presented in the model. By saving this model, they will be associated to the individuals which satisfy the statements declared in rule body.

Figure 3.21 shows one of example of jena rule that we created in the text file to infer a patient who suffer from a type2 diabetes, declares that patient a type of DiabeticPatient class and provides explanation on how decision was reached. Other examples of rules are provided in appendix 1. On the left, there is rule with its description on the right side. The first line denotes comment and are ignored when rules are parsed. At the next line, the prefix is declared. After that, the name of the rule (DiabetetDiagnosis), the condition to be met, rule direction (left-right) and the fact to be asserted in the model when the condition is satisfied are shown.



```
   //Example of rule file                    //Rule description

@prefix : <http://www.semanticweb.org/      Prefix declaration
alpha/DiabetesOntology#>
                                            [Description or name of the
[DiabetDiagnosis:                           rule

(?patient rdf:type :Patient)                (Condition to be met)
(?patient  :hasSymptomAndSign ?sign)        (Another condition)
(?patient :hasTest ?test)                   (Another condition)
(?test :hasAc1percentage ?value)            (Another condition)
ge(?value, '6.6'^^xsd:float )               (Another condition)

->                                          ->

(?patient :hasDiagnosis :Type2Diabetes)      (Fact to assert)
(?patient rdf:type :DiabeticPatient)         (Another fact to assert)
(?patient :hasDecisionEx :decision1  ]       (Another fact to assert) ]
```

Figure 3.21: Rule example and description

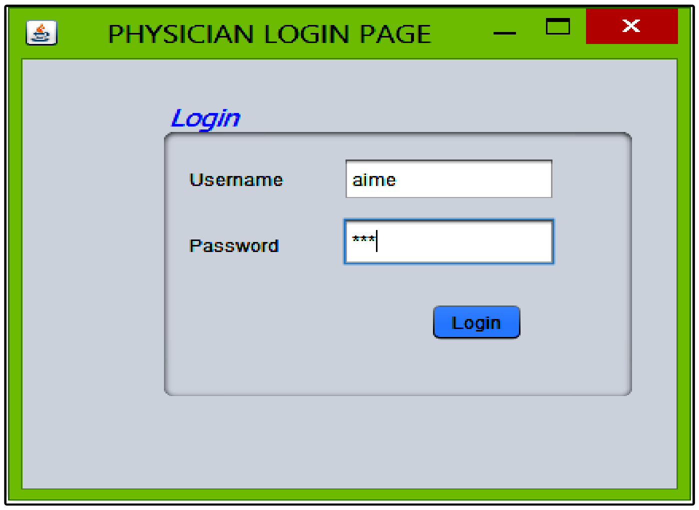
In this example, we have five terms in the condition part and three terms in the conclusion part. The first term in the condition part matches any patient (?patient) having **rdf:type** of patient. The second term matches the statements that contain **:hasSymptomAndSign** property. The third term matches the statements that contains **:hasTest** property. The fourth term matches statement that contains **:hasAc1percentage** property. The fifth term contains a built-in function**, greaterThan (),** to compare the value associated with **:hasAc1percentage** property. If all five terms are true, the head or then portion is executed or fired. In this case, those 3 terms that are in head part are added to the model associated to the reasoner. As result, the patient is associated with :**Type2Diabetes** through **:hasDiagnosis** property in the model, he/she will be classified as instance of DiabeticPatient class and associated with decision explanation through **:hasDecisionEx** property. Then, we can write the SPARQL query to retrieve this information added to the model.

3.6.6 Implementation of ontology based CDSS

Having the ontology and a set of rules, the next step is to develop a java application that a physician could use at the point of care. This application provides a physician a way to input a patient's information and to receive the system's decision such as differential diagnosis, final diagnosis, decision explanation and recommendation.

3.6.6.1 Physician login page

A physician is required to provide the right credentials (username and password) so that he/she should gain access to the system. The system matches username and password against the information stored in database. If a match is found, a user will be given access. We designed a user interface that can be used for that and is shown on **Figure 3.22**.



Username aime

Password ***

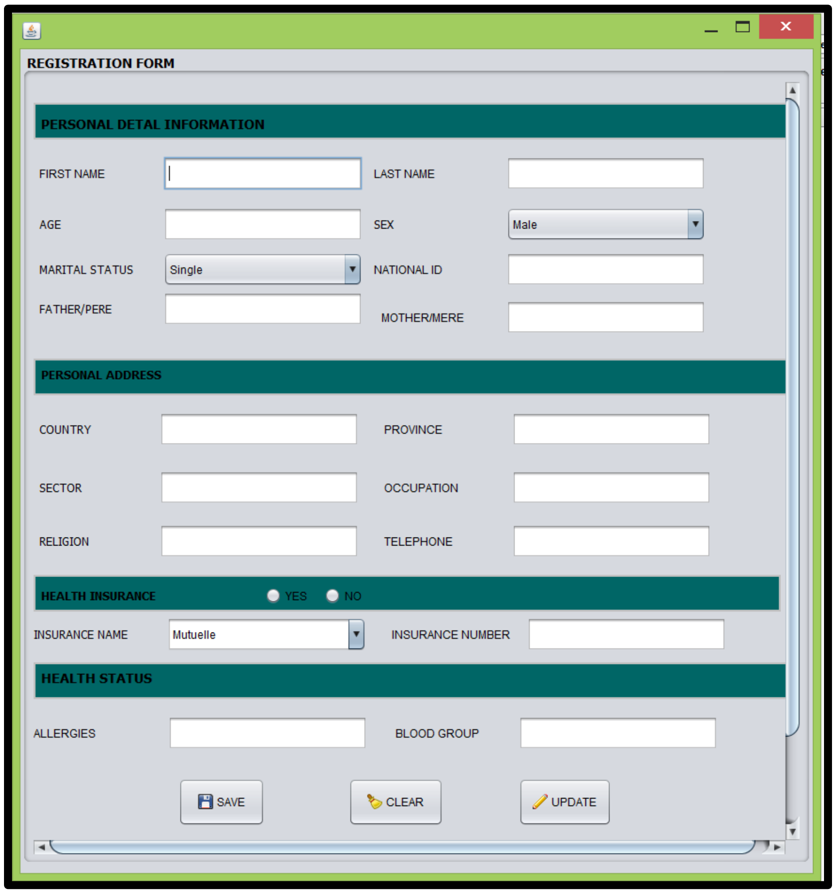
Figure 3.22: Physician login page

3.6.6.2 Formalization of patient information

Our ontology could serve as knowledge management and decision support. This means that patient related information will be recorded and kept in the ontology even after diagnosis so that they can be retrieved when needed and can also help for the patient next visit. In this regard, this information needed to be extracted and formalized in the ontology. These include a patient's personal details (patient identification, address, health status, insurance), sign and symptom and test result. All this information will be collected through a user interface and should be entered by a physician involved in the care. Note that, the diagnostic decision is only based on a patient's sign/symptom and blood glucose level of the test as defined in diagnostic criteria. Other information is there just to complete a patient medical record and for patient identification. In this section, we explain how these patient data were collected and formally represented in the ontology.

1) Formalization of patient data

When a patient visits a hospital, the first thing that a physician does is to record patient's personal details. We have created a registration form shown on **Figure 3.23** that a physician will fill. This file was created based on the outpatient file that is currently used in Rwanda hospital and can be found at Rwanda Ministry of Health website **[86]**. After completing the form, he will click save button. Upon clicking the button, the filled information will be taken so that it can be saved in the ontology.



REGISTRATION FORM

PERSONAL DETAL INFORMATION

FIRST NAME		LAST NAME	
AGE		SEX	Male
MARITAL STATUS	Single	NATIONAL ID	
FATHER/PERE		MOTHER/MERE	

PERSONAL ADDRESS

COUNTRY		PROVINCE	
SECTOR		OCCUPATION	
RELIGION		TELEPHONE	

HEALTH INSURANCE ○ YES ○ NO

INSURANCE NAME	Mutuelle	INSURANCE NUMBER	

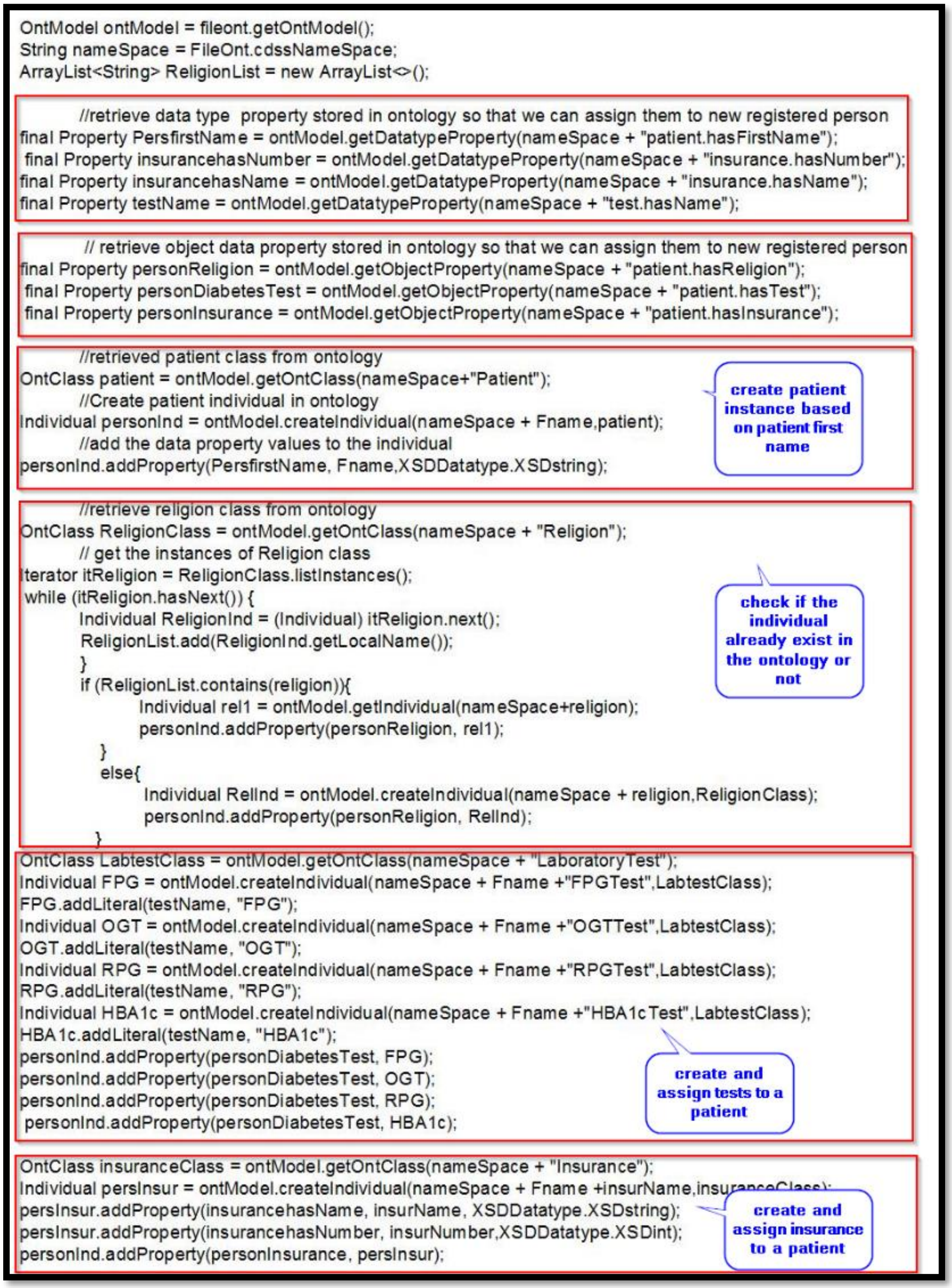
HEALTH STATUS

ALLERGIES		BLOOD GROUP	

Figure 3.23: patient registration form

Based on the information represented on patient form, we retrieved both object and data properties from the model so that they can be associated with the patient. After getting those properties from the model, some classes were also retrieved. Note that registered patient will be an instance of patient class, so, the patient class needs to be also retrieved. Additionally, some properties will be added as either a data type property (age, first name, last name, etc) or object property (test, religion, allergies, etc).

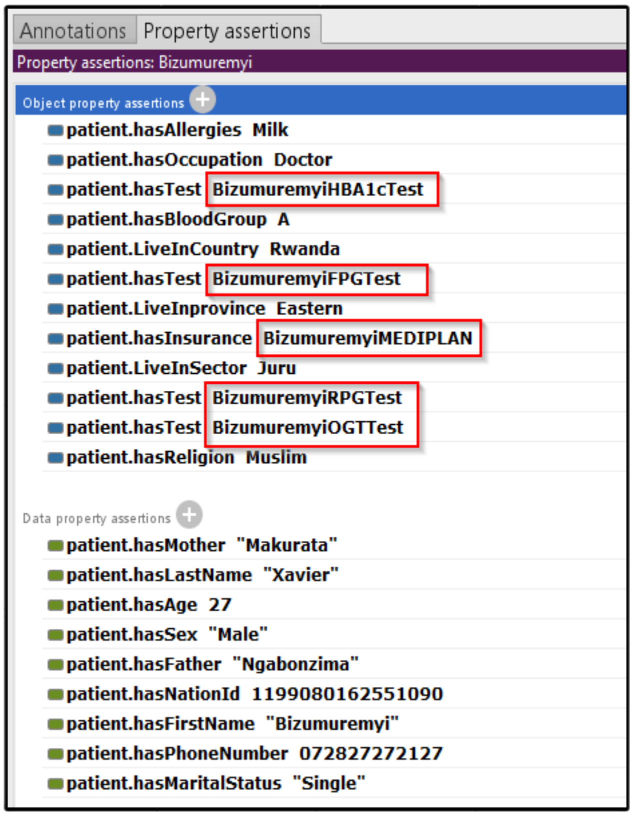
As we have discussed earlier, object property related individuals from two classes. For instance, individual of patient class is related to individual of religion class through **patient.hasReligion** property. To make sure that there is no duplication in our model, the system should check if this individual already exists in the model, if so it gets it and associates it with the patient by using **addProperty** method. If the individual does not exist, it creates a new individual and associates it to the patient. Each patient should have his/her own blood glucose test and its associated value. similarly, he/she should have its own insurance associated with its number. To handle this, the system creates four blood glucose test individuals based on the patient first name and automatically assigns them to the patient. Their value will be assigned based on examination test selected by a physician at next step when needed. A system also creates insurance individual based on the patient first name and insurance name and assigns it to the patient. **Figure 3.24** and **Figure 3.25** show sample code and its ontological representation respectively.



```
OntModel ontModel = fileont.getOntModel();
String nameSpace = FileOnt.cdssNameSpace;
ArrayList<String> ReligionList = new ArrayList<>();

        //retrieve data type  property stored in ontology so that we can assign them to new registered person
final Property PersfirstName = ontModel.getDatatypeProperty(nameSpace + "patient.hasFirstName");
 final Property insurancehasNumber = ontModel.getDatatypeProperty(nameSpace + "insurance.hasNumber");
final Property insurancehasName = ontModel.getDatatypeProperty(nameSpace + "insurance.hasName");
final Property testName = ontModel.getDatatypeProperty(nameSpace + "test.hasName");

         // retrieve object data property stored in ontology so that we can assign them to new registered person
final Property personReligion = ontModel.getObjectProperty(nameSpace + "patient.hasReligion");
 final Property personDiabetesTest = ontModel.getObjectProperty(nameSpace + "patient.hasTest");
 final Property personInsurance = ontModel.getObjectProperty(nameSpace + "patient.hasInsurance");

        //retrieved patient class from ontology
OntClass patient = ontModel.getOntClass(nameSpace+"Patient");
        //Create patient individual in ontology
Individual personInd = ontModel.createIndividual(nameSpace + Fname,patient);
        //add the data property values to the individual
personInd.addProperty(PersfirstName, Fname,XSDDatatype.XSDstring);

        //retrieve religion class from ontology
OntClass ReligionClass = ontModel.getOntClass(nameSpace + "Religion");
        // get the instances of Religion class
Iterator itReligion = ReligionClass.listInstances();
 while (itReligion.hasNext()) {
        Individual ReligionInd = (Individual) itReligion.next();
        ReligionList.add(ReligionInd.getLocalName());
        }
        if (ReligionList.contains(religion)){
                Individual rel1 = ontModel.getIndividual(nameSpace+religion);
                personInd.addProperty(personReligion, rel1);
          }
          else{
                 Individual RelInd = ontModel.createIndividual(nameSpace + religion,ReligionClass);
                 personInd.addProperty(personReligion, RelInd);
          }
OntClass LabtestClass = ontModel.getOntClass(nameSpace + "LaboratoryTest");
Individual FPG = ontModel.createIndividual(nameSpace + Fname +"FPGTest",LabtestClass);
FPG.addLiteral(testName, "FPG");
Individual OGT = ontModel.createIndividual(nameSpace + Fname +"OGTTest",LabtestClass);
OGT.addLiteral(testName, "OGT");
Individual RPG = ontModel.createIndividual(nameSpace + Fname +"RPGTest",LabtestClass);
RPG.addLiteral(testName, "RPG");
Individual HBA1c = ontModel.createIndividual(nameSpace + Fname +"HBA1cTest",LabtestClass);
HBA1c.addLiteral(testName, "HBA1c");
personInd.addProperty(personDiabetesTest, FPG);
personInd.addProperty(personDiabetesTest, OGT);
personInd.addProperty(personDiabetesTest, RPG);
 personInd.addProperty(personDiabetesTest, HBA1c);

OntClass insuranceClass = ontModel.getOntClass(nameSpace + "Insurance");
Individual persInsur = ontModel.createIndividual(nameSpace + Fname +insurName,insuranceClass);
persInsur.addProperty(insurancehasName, insurName, XSDDatatype.XSDstring);
persInsur.addProperty(insurancehasNumber, insurNumber,XSDDatatype.XSDint);
personInd.addProperty(personInsurance, persInsur);
```

create patient instance based on patient first name
check if the individual already exist in the ontology or not
create and assign tests to a patient
create and assign insurance to a patient

Figure 3.24: Sample code for patient registration



Property assertions: Bizumuremyi

Object property assertions

```
patient.hasAllergies  Milk
patient.hasOccupation  Doctor
patient.hasTest  BizumuremyiHBA1cTest
patient.hasBloodGroup  A
patient.LiveInCountry  Rwanda
patient.hasTest  BizumuremyiFPGTest
patient.LiveInprovince  Eastern
patient.hasInsurance  BizumuremyiMEDIPLAN
patient.LiveInSector  Juru
patient.hasTest  BizumuremyiRPGTest
patient.hasTest  BizumuremyiOGTTest
patient.hasReligion  Muslim
```

Data property assertions

```
patient.hasMother  "Makurata"
patient.hasLastName  "Xavier"
patient.hasAge  27
patient.hasSex  "Male"
patient.hasFather  "Ngabonzima"
patient.hasNationId  1199080162551090
patient.hasFirstName  "Bizumuremyi"
patient.hasPhoneNumber  072827272127
patient.hasMaritalStatus  "Single"
```

Figure 3.25: Ontological representation of patient data

2) Formalization of patient signs/symptoms and test result in the ontology

Patient’s clinical information (i.e sign/symptom and test result) are also collected through a user interface. Before collecting patient’s sign/symptom, a physician will be given two options. The first option is to start with signs/symptoms and the second option is to start with a disease. Based on the selected option, another interface will be displayed. Let assume that the physician chooses to start with sign/Symptom, then the interface which contains the list of sign and symptom and another empty list will be displayed. At this moment, the physician would choose among all the existing sign/symptom and add them to the empty list based his observation (patient case) and click diagnose button. **Figure 3.26** shows three interfaces. The first interface is option selection, the second will contain the physician choice and the last interface shows how the second interface looks like after physician’s choice. **Figure 3.27** shows an interface that lets the physician select and enter blood glucose test result.

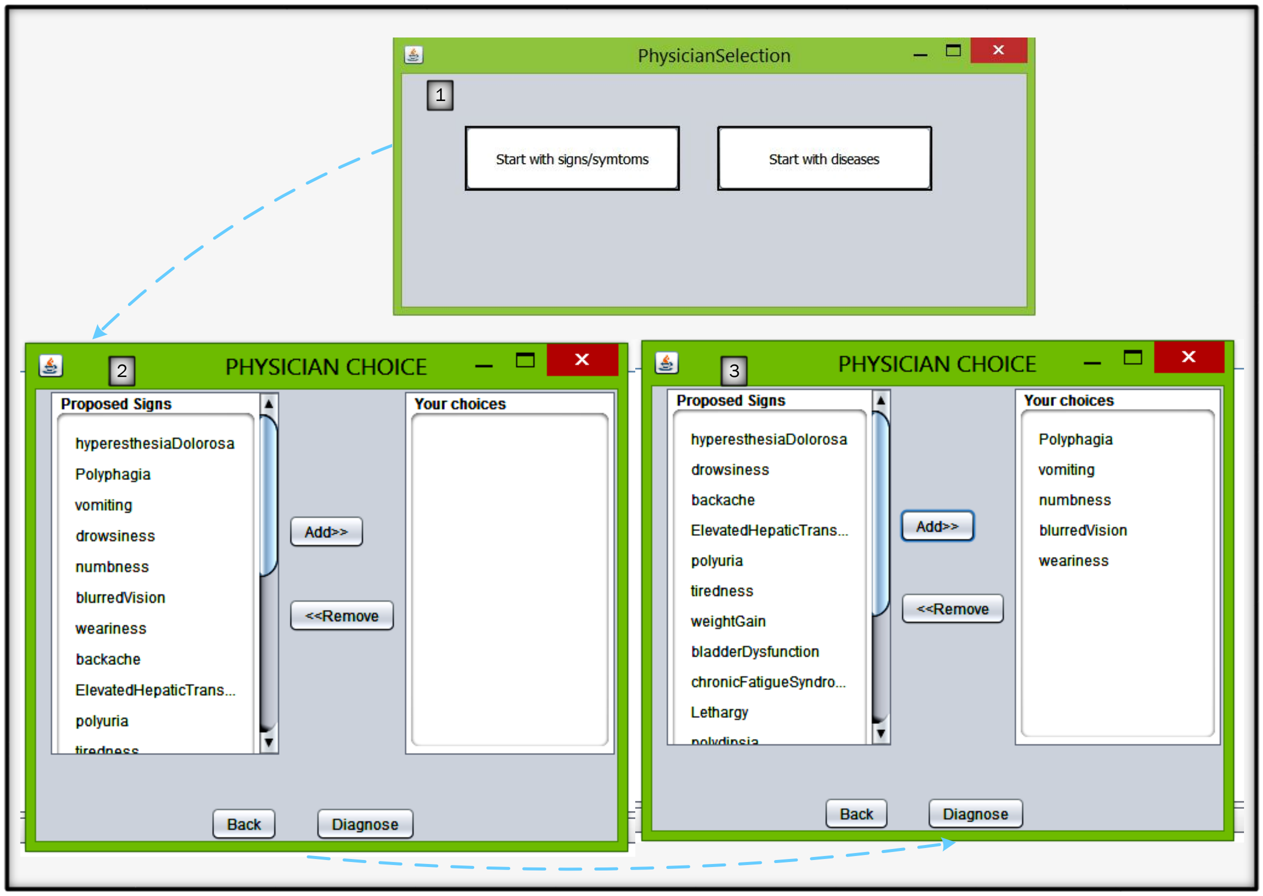
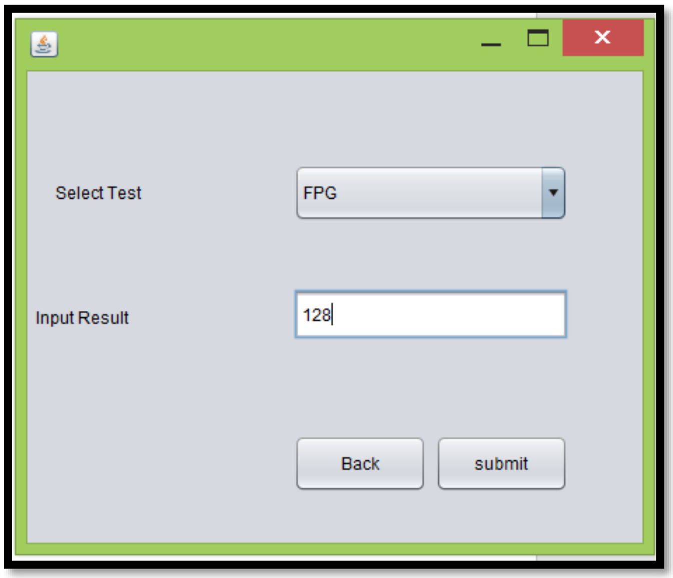


Figure 3.26: Physician selection of signs/symptoms

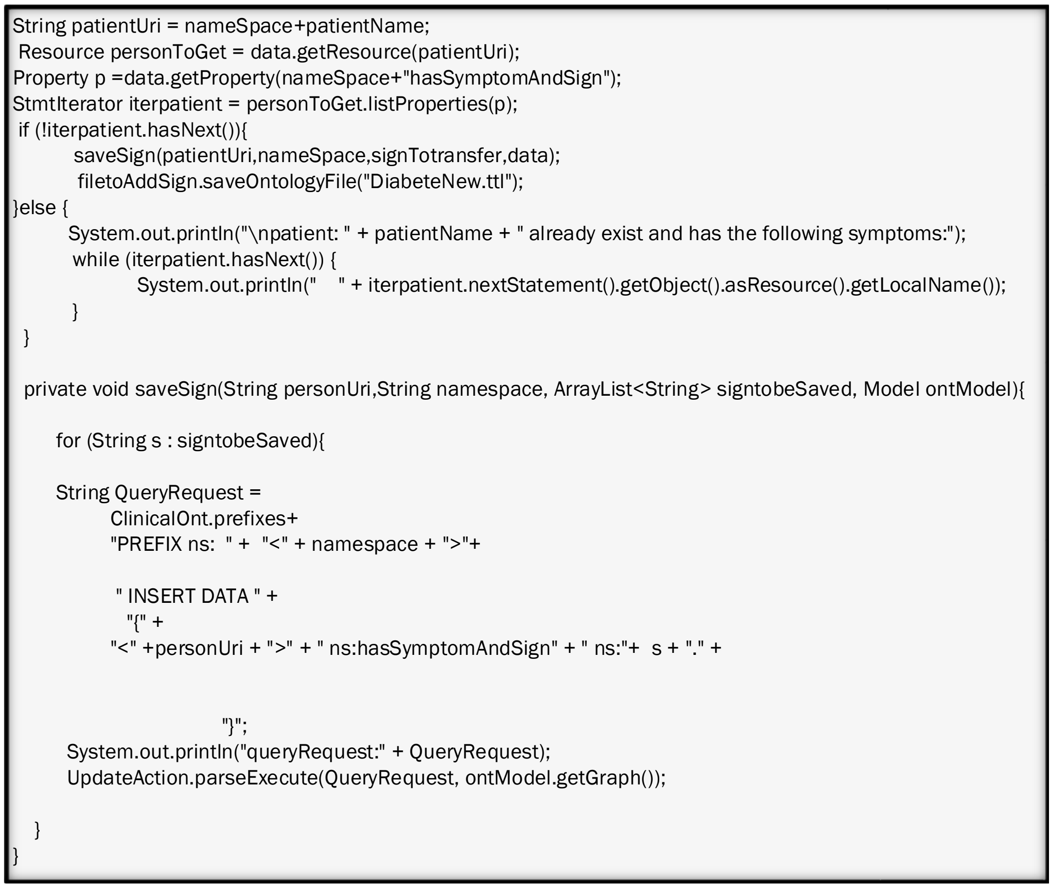


Select Test FPG

Input Result 128

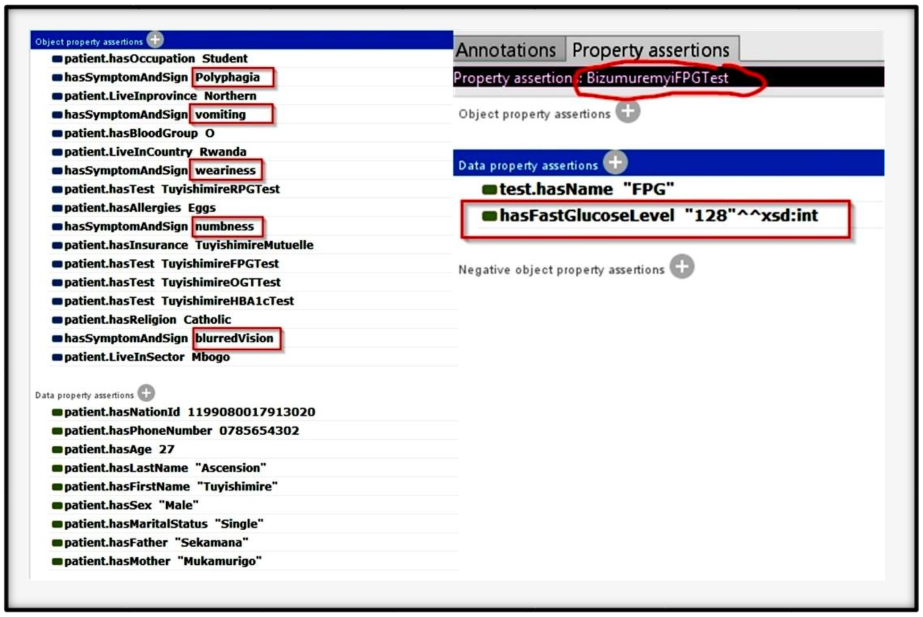
Figure 3.27: User interface for diabetes blood glucose test

After selecting and reviewing his/her choice, the physician will click diagnose button. The system will take the selected signs/symptoms and add them to an array list. It will also check if that patient has already diagnosed. If yes, it will return the signs/symptoms already associated to this patient otherwise it associates them to a patient through h**asSymptomAndSign** property. To save this information in the model, we created a method and passed patient URI created based on the patient first name, array string that contains physician's choice and ontology model in the method's parameter. In the method body, we used SPARQL update query to insert this information and update graph as shown in **Figure 3.28**. The same procedure was used to save patient's test result in the model and the ontological representation is shown in **Figure 3.29**. In total, we have created one SPARQL update query to save patient vital sign and four SPARQL update queries to save patient's test result.



```
String patientUri = nameSpace+patientName;
 Resource personToGet = data.getResource(patientUri);
Property p =data.getProperty(nameSpace+"hasSymptomAndSign");
StmtIterator iterpatient = personToGet.listProperties(p);
 if (!iterpatient.hasNext()){
          saveSign(patientUri,nameSpace,signTotransfer,data);
           filetoAddSign.saveOntologyFile("DiabeteNew.ttl");
}else {
         System.out.println("\npatient: " + patientName + " already exist and has the following symptoms:");
          while (iterpatient.hasNext()) {
                   System.out.println("    " + iterpatient.nextStatement().getObject().asResource().getLocalName());
          }
  }

  private void saveSign(String personUri,String namespace, ArrayList<String> signtobeSaved, Model ontModel){

       for (String s : signtobeSaved){

       String QueryRequest =
               ClinicalOnt.prefixes+
               "PREFIX ns:  " +  "<" + namespace + ">"+

                " INSERT DATA " +
                  "{" +
               "<" +personUri + ">" + " ns:hasSymptomAndSign" + " ns:"+  s + "." +


                                "}";
         System.out.println("queryRequest:" + QueryRequest);
         UpdateAction.parseExecute(QueryRequest, ontModel.getGraph());

    }
}
```

Figure 3.28: Getting and saving patient sign/symptom into the ontology



```
Object property assertions
patient.hasOccupation  Student
hasSymptomAndSign  Polyphagia
patient.LiveInprovince  Northern
hasSymptomAndSign  vomiting
patient.hasBloodGroup  O
patient.LiveInCountry  Rwanda
hasSymptomAndSign  weariness
patient.hasTest  TuyishimireRPGTest
patient.hasAllergies  Eggs
hasSymptomAndSign  numbness
patient.hasInsurance  TuyishimireMutuelle
patient.hasTest  TuyishimireFPGTest
patient.hasTest  TuyishimireOGTTest
patient.hasTest  TuyishimireHBA1cTest
patient.hasReligion  Catholic
hasSymptomAndSign  blurredVision
patient.LiveInSector  Mbogo

Data property assertions
patient.hasNationId  1199080017913020
patient.hasPhoneNumber  0785654302
patient.hasAge  27
patient.hasLastName  "Ascension"
patient.hasFirstName  "Tuyishimire"
patient.hasSex  "Male"
patient.hasMaritalStatus  "Single"
patient.hasFather  "Sekamana"
patient.hasMother  "Mukamurigo"
```

```
Annotations  Property assertions
Property assertion : BizumuremyiFPGTest
Object property assertions
Data property assertions
test.hasName  "FPG"
hasFastGlucoseLevel  "128"^^xsd:int
Negative object property assertions
```

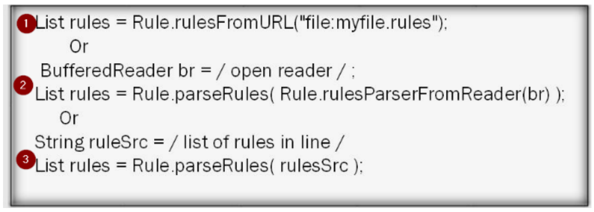
Figure 3.29: Ontological representation after saving vital sign and test result

3.6.6.3 Data loading, reasoning and consistency checking

As we have discussed earlier, for the application to produce the accurate output, it must contain both facts and medical knowledge (in form of if-then clause). This information is then transferred to the rule engine and a rule engine uses algorithm to combine facts and medical knowledge to produce an accurate decision. This section explains how rule, ontology file was loaded and transferred to the jena rule engine, and how the inconsistency can be detected if anything goes wrong.

1) Loading rule file

Jena offers three ways to load rules. **Figure 3.30** shows sample of java code that loads and parses the rule file as well the java code for parsing the rule that is written in the java application. In the first case, the rule file is loaded from URL while the second case, the BufferedReader is used.



```
1 List rules = Rule.rulesFromURL("file:myfile.rules");
      Or
  BufferedReader br = / open reader / ;
2 List rules = Rule.parseRules( Rule.rulesParserFromReader(br) );
     Or
 String ruleSrc = / list of rules in line /
3 List rules = Rule.parseRules( rulesSrc );
```

Figure 3.30:Jena code for loading and parsing rule

In both case, the rule file is preprocessed by a simple processor in order to strip the comment or any other macro commands. The third case, rules are defined within the application. To load our rule from text file, we used the first case.

2) Loading ontology file

To load the ontology file, we created two java classes. First class contains two methods responsible for reading and saving model and return ontModel. Using **ModelFactory** class, we created an empty ontology model (this model uses **OWL full profile**, **in-memory storage** and does not use any reasoner) which will hold all statements asserted in our ontology file after the file is read. **Figure 3.31** shows java code that create ontology model, read and save the ontology file given the file name and return the model.

```
public OntModel getOntModel()  {
                return ontModel;
}
public ClinicalOnt()  {
        // Create an empty model
        ontModel = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
}
public boolean readOntologyFile(String fileName){
                InputStream in = FileManager.get().open( fileName );
                if (in == null) {
                        System.out.println("File: " + fileName + " not found");
                        return false;
                } else {                    ontModel.read(in, null, "Turtle")
                        return true;
                }
        }
public boolean saveOntologyFile(String fileName) {
                // assuming turtle format
                FileOutputStream outStream = null;
                try   {
                        outStream = new FileOutputStream(fileName);
                } catch (FileNotFoundException e)  {
                        System.err.println("Error - can not open file:" + fileName);
                        return false;
                }
                ontModel.write(outStream, "ttl");
                try  {
                        outStream.close();
                } catch (IOException e)   {
                        System.err.println("Error writing to file: " + fileName);
                        return false;
                }
                return true;
        }
}
```

Figure 3.31: Java code for loading and saving an ontology model

The second class is subclass of the first one and is responsible for reading the ontology file

```
public class FileOnt extends ClinicalOnt {
    public FileOnt() {
        super();
    this.readOntologyFile("DiabeteNew.ttl");
    }
```

3) Transfer rule, ontology file to rule engine and consistency checking

After loading the rule and ontology (i.e a file that contain facts) file in our application, we created an instance of reasoner and a ruleset is passed into its constructor. That reasoner was used to create an inference model that associates both reasoner and our ontology model. Query to this model will return both statements asserted in our ontology and additional statements that derived using rules. This model needs to be checked for inconsistency and throw an error if anything is incorrect. Jena offers validation interface (inf.validate()) for such purpose and gives us a validityReport object which comprises a simple pass/fail flag (Validity.isValid()) together with a list of specific reports which detail any detected inconsistencies. **Figure 3.32** shows a sample of java code which associates reasoner to the ontology to create an inference model, check inconsistency against inference model and list any problem if found.

```
List rules = Rule.rulesFromURL("file:ruleTest.txt");
Reasoner reasoner = new GenericRuleReasoner(rules);
FileOnt filetoTest = new FileOnt();
OntModel data = filetoTest.getOntModel();
InfModel inf = ModelFactory.createInfModel(reasoner, data);
ValidityReport validity = inf.validate();
         if (validity.isValid()) {
                  System.out.println("OK");
        } else {
                  System.out.println("Conflicts");
                  for (Iterator i = validity.getReports(); i.hasNext(); ) {
                          System.out.println(" - " + i.next());
                  }
        }
```

Figure 3.32: Java code to create an inference model and consistency checking

3.6.6.4 Using SPARQL to retrieve information from ontology

Having inference Model that containing the asserted and inferred statements, we used SPARQL to search and query the triples stored in this model. The result should be made for specific individual patient. In this way, we created a method that receives a patient's name and inference model. The system should first check if the model is empty. If yes it returns null. **Figure 3.33** shows one example of a query that we have in our system. The aim of this query is to retrieve a diagnosis of a given patient. More specifically, it will match anyone who is an instance of patient; has any sign and symptom; has any decision explanation and that decision explanation has description; has any test and that test has fast glucose level and name; has any diagnosis and that diagnosis has label; has first name that passed in the method's parameter. The same type of queries was also created to retrieve description (i.e which explain how the result was obtained) and recommendation. To be able to deal with several patient cases, 35 SPARQL select queries were created. The result of query could be then displayed in interface shown on **Figure 3.34**.

```
static public String getpatientDiagnosisfpg(String patientName, InfModel
info){
    if(info == null) return null;

    String queryRequest =
        ClinicalOnt.prefixes+
        "SELECT ?diaglabel \n"
        +"WHERE{\n "
        +"?patient rdf:type :Patient. \n"
        +"?patient :hasSymptomAndSign ?sign. \n"
        +"?patient :hasDecisionEx ?Decision. \n"
        +"?Decision :hasDescription ?Description. \n"
        +"?patient :patient.hasTest ?test. \n"
        +"?test :hasFastGlucoseLevel ?value.\n"
        +"?test :test.hasName ?testname. \n"
        +"?patient :hasDiagnosis ?diagnosis.\n "
        +"?diagnosis rdfs:label ?diaglabel. \n"
        +"?patient  :patient.hasFirstName   " + "\"" + patientName + "\" . \
n" +
          "}";
    //Create select
        Query query;
        QueryExecution qexec;
        try {
            query = QueryFactory.create(queryRequest);
            qexec = QueryExecutionFactory.create(query, info);
        } catch(Exception e){
            System.out.println(e);
            return null;
        }

        //Run select
        ResultSet response = null;
        try {
            response = qexec.execSelect();
            while( response.hasNext()){
                QuerySolution QuerySolution = response.nextSolution();

                RDFNode diagnosisNode =
QuerySolution.get("?diaglabel");
                if( diagnosisNode == null ) break;
                return diagnosisNode.toString();
            }
        } finally {qexec.close();}
```

Figure 3.33: Example of SPARQL query to retrieve diagnosis

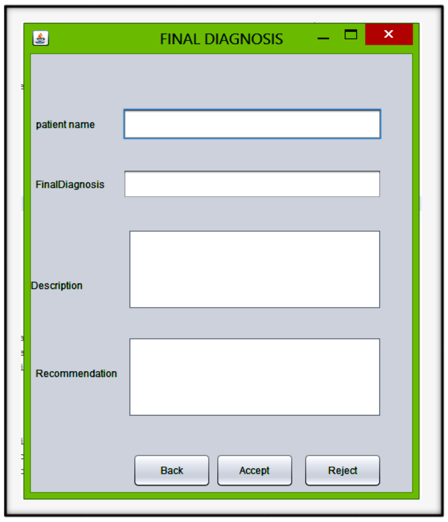
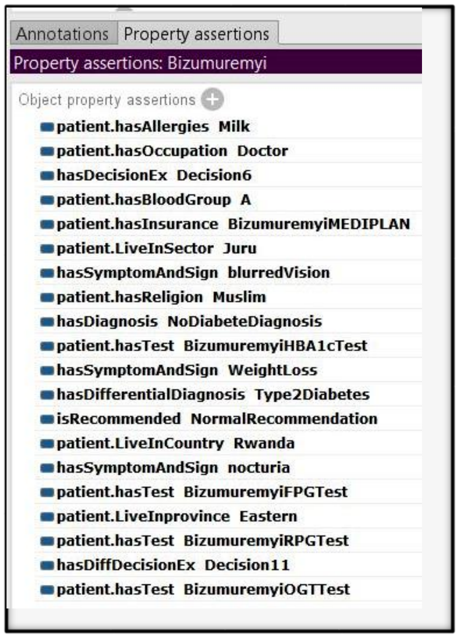


Figure 3.34: Interface for patient final diagnosis

A physician is given a button that allows him/her to accept or reject a decision. When the decision is accepted, the system will take all patient related information and associate them to a patient. This information can be retrieved later, for instance, when a patient comes back to the hospital and when hospital or any other organization ask for information regarding a patient. **Figure 3.35** shows a part of property assertion of patient after a physician accepts the system's decision.



Property assertions: Bizumuremyi

Object property assertions

- patient.hasAllergies Milk
- patient.hasOccupation Doctor
- hasDecisionEx Decision6
- patient.hasBloodGroup A
- patient.hasInsurance BizumuremyiMEDIPLAN
- patient.LiveInSector Juru
- hasSymptomAndSign blurredVision
- patient.hasReligion Muslim
- hasDiagnosis NoDiabeteDiagnosis
- patient.hasTest BizumuremyiHBA1cTest
- hasSymptomAndSign WeightLoss
- hasDifferentialDiagnosis Type2Diabetes
- isRecommended NormalRecommendation
- patient.LiveInCountry Rwanda
- hasSymptomAndSign nocturia
- patient.hasTest BizumuremyiFPGTest
- patient.LiveInprovince Eastern
- patient.hasTest BizumuremyiRPGTest
- hasDiffDecisionEx Decision11
- patient.hasTest BizumuremyiOGTTest

Figure 3.35: Ontological representation of a patient after accepting system's decision

4 Testing and validation

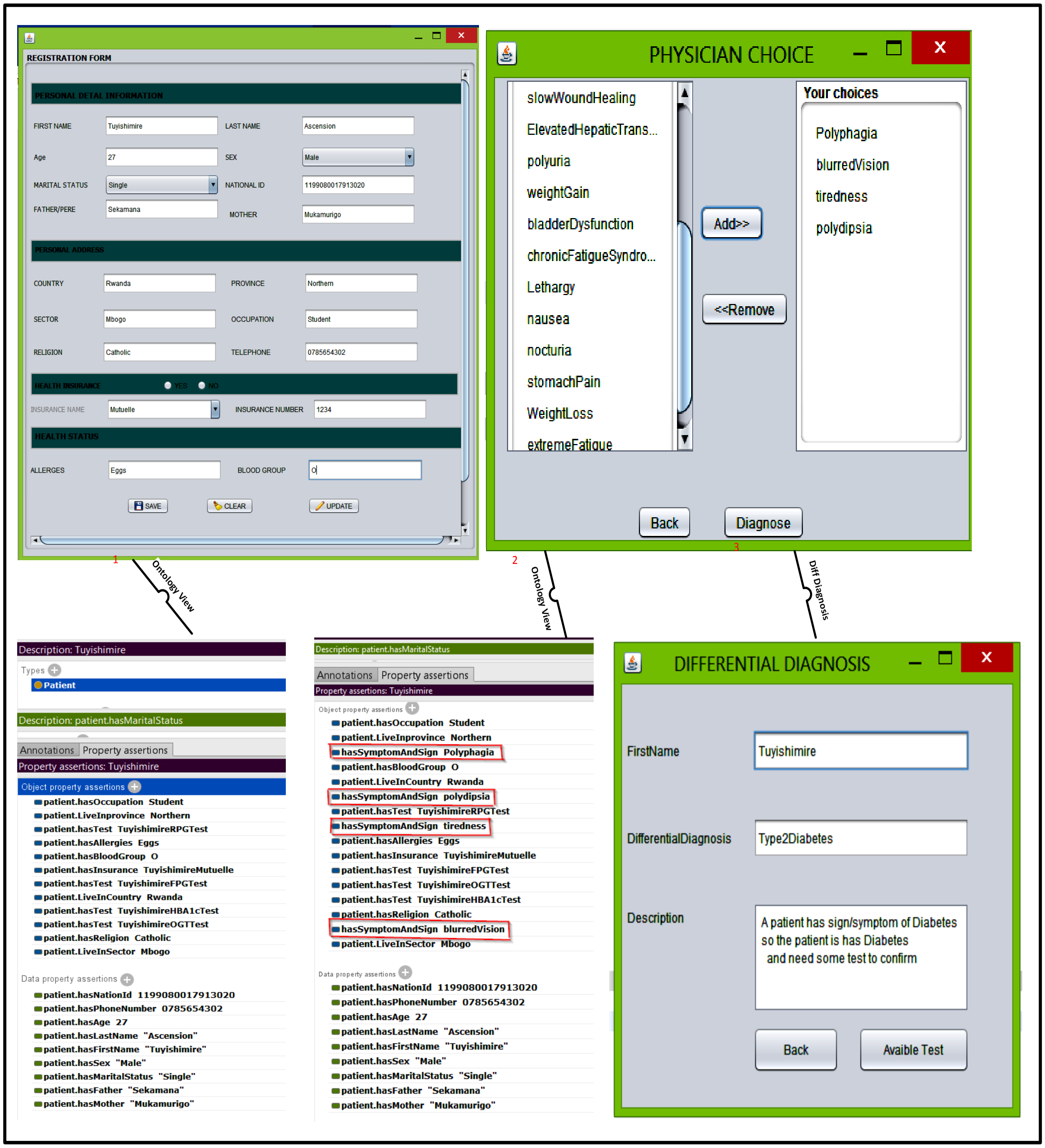
After implementation phase, the next step is to test and evaluate the system in order to check if it really meets the objective and user requirements. Note that the project objective was to develop an application that could assist a physician when making diagnosis of diabetes mellitus. During testing phase, the real patient clinical data are needed, however, we did not find any hospital that can give us these data. In cooperation with one of the student from faculty of health and sport sciences at the University of Agder, an imaginary patient data was formulated and used into the system to check the accuracy of system output. This section presents the results obtained after testing the system with 4 patients.

4.1 Diagnosing patient1 case

The first patient that was tested is Tuyishimire Ascension. He is from Rwanda, Northern province and Mbogo sector. He is a 27-year-old, single man. He is a student and uses Mutuelle insurance with 1234 number. He is allergic to eggs and has O blood group. The remaining patient information is shown in the filled form in **Figure 4.1**. He presents polyphagia, BlurredVision, tiredness and polydipsia. He has 2 tests: HBA1c and FPG with blood glucose level 6.9 % and 128 mg/dl respectively.

4.1.1 Patient1 data and vital sign extraction, formalization and differential diagnosis retrieval.

Nurse inputs the patient details and vital signs into the system. Upon clicking diagnose button, the application infers that Tuyishimire suffers from Type2Diabete because he is having sign/symptom of diabetes and suggests that some tests are needed before confirming the diagnosis. The way the patient details and vital signs was inputted into the system, their ontological representation and the obtained result are shown in **Figure 4.1**.



REGISTRATION FORM

PERSONAL DETAL INFORMATION

FIRST NAME	Tuyishimire	LAST NAME	Ascension
Age	27	SEX	Male
MARITAL STATUS	Single	NATIONAL ID	1199080017913020
FATHER/PERE	Sekamana	MOTHER	Mukamurigo

PERSONAL ADDRESS

COUNTRY	Rwanda	PROVINCE	Northern
SECTOR	Mbogo	OCCUPATION	Student
RELIGION	Catholic	TELEPHONE	0785654302

HEALTH INSURANCE YES NO

INSURANCE NAME	Mutuelle	INSURANCE NUMBER	1234

HEALTH STATUS

ALLERGES	Eggs	BLOOD GROUP	O

SAVE CLEAR UPDATE

PHYSICIAN CHOICE

slowWoundHealing
ElevatedHepaticTrans...
polyuria
weightGain
bladderDysfunction
chronicFatigueSyndro...
Lethargy
nausea
nocturia
stomachPain
WeightLoss
extremeFatigue

Add>> <<Remove

Your choices

Polyphagia
blurredVision
tiredness
polydipsia

Back Diagnose

Description: Tuyishimire

Types

Patient

Description: patient.hasMaritalStatus

Annotations Property assertions

Property assertions: Tuyishimire

Object property assertions

```
patient.hasOccupation  Student
patient.LiveInprovince  Northern
patient.hasTest  TuyishimireRPGTest
patient.hasAllergies  Eggs
patient.hasBloodGroup  O
patient.hasInsurance  TuyishimireMutuelle
patient.hasTest  TuyishimireFPGTest
patient.LiveInCountry  Rwanda
patient.hasTest  TuyishimireHBA1cTest
patient.hasTest  TuyishimireOGTTest
patient.hasReligion  Catholic
patient.LiveInSector  Mbogo
```

Data property assertions

```
patient.hasNationId  1199080017913020
patient.hasPhoneNumber  0785654302
patient.hasAge  27
patient.hasLastName  "Ascension"
patient.hasFirstName  "Tuyishimire"
patient.hasSex  "Male"
patient.hasMaritalStatus  "Single"
patient.hasFather  "Sekamana"
patient.hasMother  "Mukamurigo"
```

Description: patient.hasMaritalStatus

Annotations Property assertions

Property assertions: Tuyishimire

Object property assertions

```
patient.hasOccupation  Student
patient.LiveInprovince  Northern
hasSymptomAndSign  Polyphagia
patient.hasBloodGroup  O
patient.LiveInCountry  Rwanda
hasSymptomAndSign  polydipsia
patient.hasTest  TuyishimireRPGTest
hasSymptomAndSign  tiredness
patient.hasAllergies  Eggs
patient.hasInsurance  TuyishimireMutuelle
patient.hasTest  TuyishimireFPGTest
patient.hasTest  TuyishimireOGTTest
patient.hasTest  TuyishimireHBA1cTest
patient.hasReligion  Catholic
hasSymptomAndSign  blurredVision
patient.LiveInSector  Mbogo
```

Data property assertions

```
patient.hasNationId  1199080017913020
patient.hasPhoneNumber  0785654302
patient.hasAge  27
patient.hasLastName  "Ascension"
patient.hasFirstName  "Tuyishimire"
patient.hasSex  "Male"
patient.hasMaritalStatus  "Single"
patient.hasFather  "Sekamana"
patient.hasMother  "Mukamurigo"
```

DIFFERENTIAL DIAGNOSIS

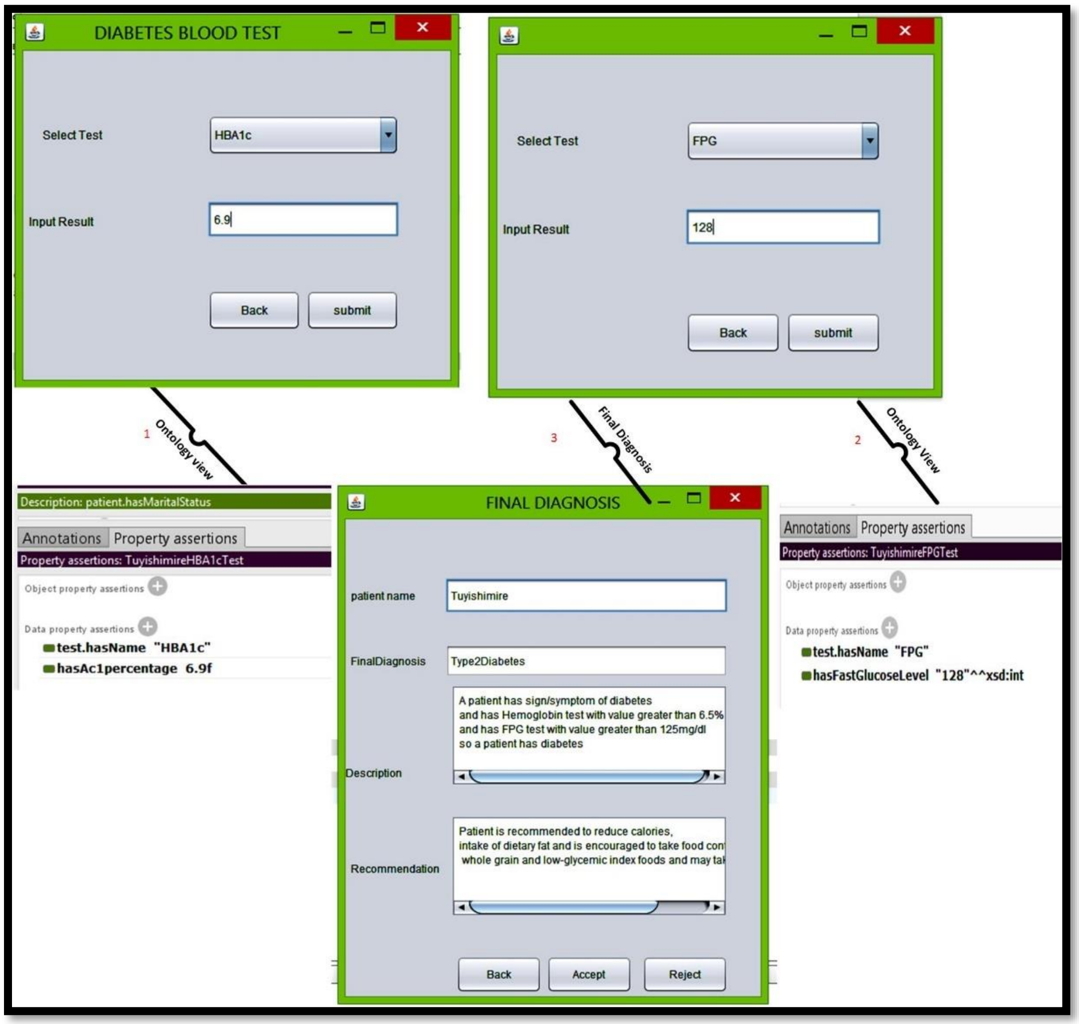
FirstName	Tuyishimire
DifferentialDiagnosis	Type2Diabetes
Description	A patient has sign/symptom of Diabetes so the patient is has Diabetes and need some test to confirm

Back Avaible Test

Figure 4.1: Extracting, formalizing Tuyishimire's data and retrieval of differential diagnosis

4.1.2 Patient1 test result extraction, formalization and final decision retrieval

Nurse uses the available test button to check laboratory tests required for diabetes diagnosis. She selects HBA1c test and inputs 6.9% into the system. The application asks her for another test. At the second time, she selects FPG and inputs 128 mg/dl into the system. Upon submission, the application confirms that Tuyishimire suffers from Type2Diabetes. This is because Tuyishimire was having sign/symptom of diabetes and hemoglobin test with value greater than 6.5% and FPG test with value greater than 125mg/dl which is true. Furthermore, the system recommends that Tuyishimire should reduce calories, intake of dietary fat and is encouraged to take food containing whole grain and low-glycemic index foods and may take medication. The selected test, its value, its ontological representation and the obtained result is shown in **Figure 4.2**.


DIABETES BLOOD TEST

Select Test: HBA1c

Input Result: 6.9

Select Test: FPG

Input Result: 128

Description: patient.hasMaritalStatus

Property assertions: TuyishimireHBA1cTest

Object property assertions

Data property assertions
- test.hasName "HBA1c"
- hasAc1percentage 6.9f

FINAL DIAGNOSIS

patient name: Tuyishimire

FinalDiagnosis: Type2Diabetes

Description: A patient has sign/symptom of diabetes
and has Hemoglobin test with value greater than 6.5%
and has FPG test with value greater than 125mg/dl
so a patient has diabetes

Recommendation: Patient is recommended to reduce calories,
intake of dietary fat and is encouraged to take food con
whole grain and low-glycemic index foods and may ta

Property assertions: TuyishimireFPGTest

Object property assertions

Data property assertions
- test.hasName "FPG"
- hasFastGlucoseLevel "128"^^xsd:int

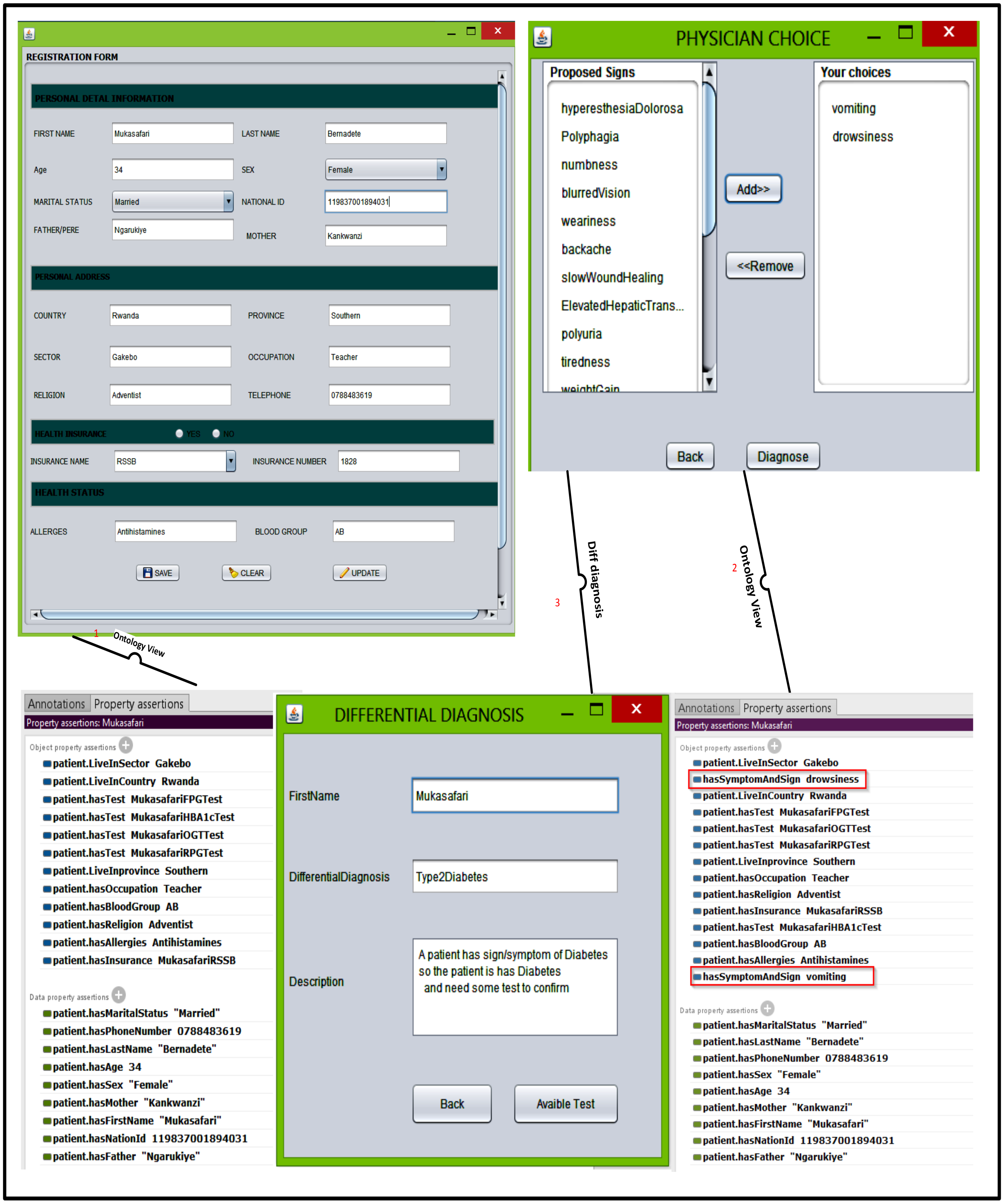
Figure 4.2: Patient1 test result extraction, formalization and final decision retrieval

4.2 Diagnosing patient2 case

The second patient was Mukasafari Bernadete. She is a 34-year-old, married woman who is from Rwanda, southern province and Gakebo sector. She is a teacher and uses RSSB insurance. She is allergic to Antihistamines and has AB blood group. The remaining information is shown in filled form in **Figure 4.3**. She presents vomiting and drowsiness. She has two tests: FPG and OGT with blood glucose level of 130 mg/dl and 205 mg/dl respectively.

4.2.1 Patient2 data and vital sign extraction, formalization and differential diagnosis retrieval.

Nurse inputs the patient details and vital signs into the system. Upon clicking diagnose button, the system infers that Mukasafari suffers from type2 diabetes because she is presenting sign/symptom of diabetes and that tests are needed to confirm the diagnosis. The way the patient details and vital signs was inputted into the system, their ontological representation and the obtained result are shown in **Figure 4.3**.



REGISTRATION FORM

PERSONAL DETAIL INFORMATION

FIRST NAME	Mukasafari	LAST NAME	Bernadete
Age	34	SEX	Female
MARITAL STATUS	Married	NATIONAL ID	119837001894031
FATHER/PERE	Ngarukiye	MOTHER	Kankwanzi

PERSONAL ADDRESS

COUNTRY	Rwanda	PROVINCE	Southern
SECTOR	Gakebo	OCCUPATION	Teacher
RELIGION	Adventist	TELEPHONE	0788483619

HEALTH INSURANCE YES NO

INSURANCE NAME	RSSB	INSURANCE NUMBER	1828

HEALTH STATUS

ALLERGES	Antihistamines	BLOOD GROUP	AB

SAVE CLEAR UPDATE

PHYSICIAN CHOICE

Proposed Signs	Your choices
hyperesthesiaDolorosa	vomiting
Polyphagia	drowsiness
numbness	
blurredVision	
weariness	
backache	
slowWoundHealing	
ElevatedHepaticTrans...	
polyuria	
tiredness	
weightGain	

Add>> <<Remove Back Diagnose

Annotations Property assertions

Property assertions: Mukasafari

Object property assertions
- patient.LiveInSector Gakebo
- patient.LiveInCountry Rwanda
- patient.hasTest MukasafariFPGTest
- patient.hasTest MukasafariHBA1cTest
- patient.hasTest MukasafariOGTTest
- patient.hasTest MukasafariRPGTest
- patient.LiveInprovince Southern
- patient.hasOccupation Teacher
- patient.hasBloodGroup AB
- patient.hasReligion Adventist
- patient.hasAllergies Antihistamines
- patient.hasInsurance MukasafariRSSB

Data property assertions
- patient.hasMaritalStatus "Married"
- patient.hasPhoneNumber 0788483619
- patient.hasLastName "Bernadete"
- patient.hasAge 34
- patient.hasSex "Female"
- patient.hasMother "Kankwanzi"
- patient.hasFirstName "Mukasafari"
- patient.hasNationId 119837001894031
- patient.hasFather "Ngarukiye"

DIFFERENTIAL DIAGNOSIS

FirstName	Mukasafari
DifferentialDiagnosis	Type2Diabetes
Description	A patient has sign/symptom of Diabetes so the patient is has Diabetes and need some test to confirm

Back Avaible Test

Annotations Property assertions

Property assertions: Mukasafari

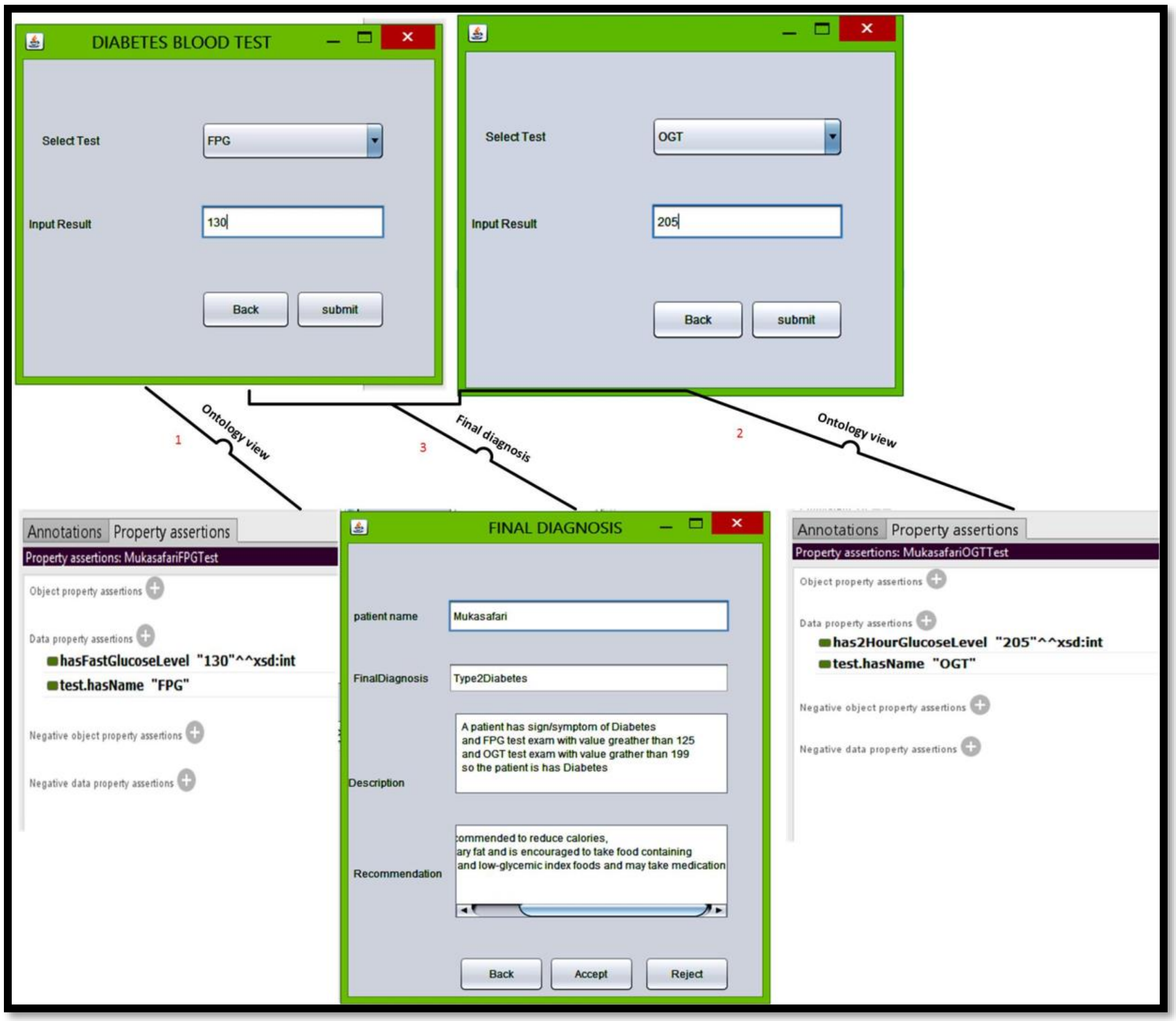
Object property assertions
- patient.LiveInSector Gakebo
- hasSymptomAndSign drowsiness
- patient.LiveInCountry Rwanda
- patient.hasTest MukasafariFPGTest
- patient.hasTest MukasafariOGTTest
- patient.hasTest MukasafariRPGTest
- patient.LiveInprovince Southern
- patient.hasOccupation Teacher
- patient.hasReligion Adventist
- patient.hasInsurance MukasafariRSSB
- patient.hasTest MukasafariHBA1cTest
- patient.hasBloodGroup AB
- patient.hasAllergies Antihistamines
- hasSymptomAndSign vomiting

Data property assertions
- patient.hasMaritalStatus "Married"
- patient.hasLastName "Bernadete"
- patient.hasPhoneNumber 0788483619
- patient.hasSex "Female"
- patient.hasAge 34
- patient.hasMother "Kankwanzi"
- patient.hasFirstName "Mukasafari"
- patient.hasNationId 119837001894031
- patient.hasFather "Ngarukiye"

Figure 4.3: patient2 data collection, formalization and differential diagnosis

4.2.2 Patient2 test result extraction, formalization and final decision retrieval

Nurse uses an available test button to see the test available in the system. she selects FPG and inputs 130 mg/dl the system. Upon submitting the result, the system asks her to select another test because one test is not enough to confirm diabetes. She selects OGT and input 205 mg/dl into the system. Upon submission, the system confirms that Mukasafari has Diabetes. This is because she has sign/symptom of diabetes, FPG test exam with value greater than 125 and OGT test exam with value greater than 199. Furthermore, the system recommends that Mukasafari should reduce calories, intake of dietary fat and is encouraged to take food containing whole grain and low-glycemic index foods and may take medication. The selected test, its value, its ontological representation and the obtained result is shown in **Figure 4.4**.



Select Test: FPG
Input Result: 130

Select Test: OGT
Input Result: 205

Property assertions: MukasafariFPGTest
Object property assertions
Data property assertions
hasFastGlucoseLevel "130"^^xsd:int
test.hasName "FPG"
Negative object property assertions
Negative data property assertions

patient name: Mukasafari
FinalDiagnosis: Type2Diabetes
Description:
A patient has sign/symptom of Diabetes
and FPG test exam with value greather than 125
and OGT test exam with value grather than 199
so the patient is has Diabetes
Recommendation:
:ommended to reduce calories,
ary fat and is encouraged to take food containing
and low-glycemic index foods and may take medication

Property assertions: MukasafariOGTTest
Object property assertions
Data property assertions
has2HourGlucoseLevel "205"^^xsd:int
test.hasName "OGT"
Negative object property assertions
Negative data property assertions

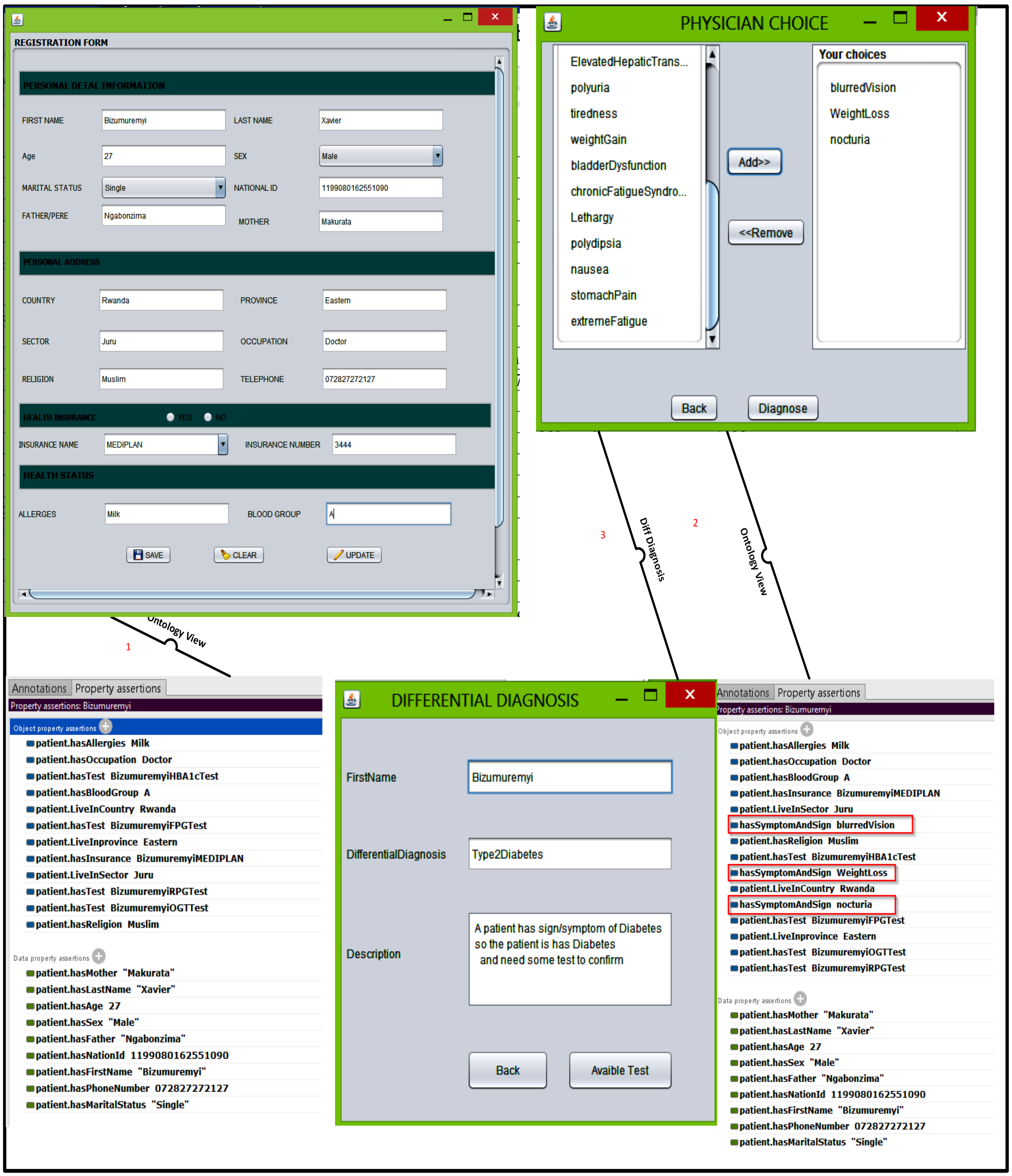
Figure 4.4: Patient2 test result extraction, formalization and final decision retrieval

4.3 Diagnosis of patient3 case

The third patient is Bizumuremyi Xavier, a 27-year-old, single man. He is from Rwanda, Eastern province, Juru sector. He is a doctor and uses MEDIPLAN insurance. He is allergic to Milk and has A blood group. The remaining information is shown in the filled form shown in **Figure 4.5**. He presents BlurredVision, WeightLoss and nocturia. He has OGT test with blood glucose level of 139 mg/dl.

4.3.1 Patient3 data and vital sign extraction, formalization and differential diagnosis retrieval.

Nurse inputs patient details and vital signs into the system. Upon clicking diagnose button, the system infers that Bizumuremyi suffers from type2 diabetes because she is presenting sign/symptom of diabetes and suggests that some tests are needed before confirmation. The way the patient details and vital signs was inputted into the system, their ontological representation and the obtained result are shown in **Figure 4.5**.



REGISTRATION FORM

PERSONAL DETAIL INFORMATION

FIRST NAME	Bizumuremyi	LAST NAME	Xavier
Age	27	SEX	Male
MARITAL STATUS	Single	NATIONAL ID	1199080162551090
FATHER/PERE	Ngabonzima	MOTHER	Makurata

PERSONAL ADDRESS

COUNTRY	Rwanda	PROVINCE	Eastern
SECTOR	Juru	OCCUPATION	Doctor
RELIGION	Muslim	TELEPHONE	072827272127

HEALTH INSURANCE ○ YES ○ NO

INSURANCE NAME	MEDIPLAN	INSURANCE NUMBER	3444

HEALTH STATUS

ALLERGES	Milk	BLOOD GROUP	A

SAVE CLEAR UPDATE

PHYSICIAN CHOICE

ElevatedHepaticTrans...
polyuria
tiredness
weightGain
bladderDysfunction
chronicFatigueSyndro...
Lethargy
polydipsia
nausea
stomachPain
extremeFatigue

Add>> <<Remove

Your choices
blurredVision
WeightLoss
nocturia

Back Diagnose

Annotations Property assertions
Property assertions: Bizumuremyi
Object property assertions
patient.hasAllergies Milk
patient.hasOccupation Doctor
patient.hasTest BizumuremyiHBA1cTest
patient.hasBloodGroup A
patient.LiveInCountry Rwanda
patient.hasTest BizumuremyiFPGTest
patient.LiveInprovince Eastern
patient.hasInsurance BizumuremyiMEDIPLAN
patient.LiveInSector Juru
patient.hasTest BizumuremyiRPGTest
patient.hasTest BizumuremyiOGTTest
patient.hasReligion Muslim
Data property assertions
patient.hasMother "Makurata"
patient.hasLastName "Xavier"
patient.hasAge 27
patient.hasSex "Male"
patient.hasFather "Ngabonzima"
patient.hasNationId 1199080162551090
patient.hasFirstName "Bizumuremyi"
patient.hasPhoneNumber 072827272127
patient.hasMaritalStatus "Single"

DIFFERENTIAL DIAGNOSIS

FirstName	Bizumuremyi
DifferentialDiagnosis	Type2Diabetes
Description	A patient has sign/symptom of Diabetes so the patient is has Diabetes and need some test to confirm

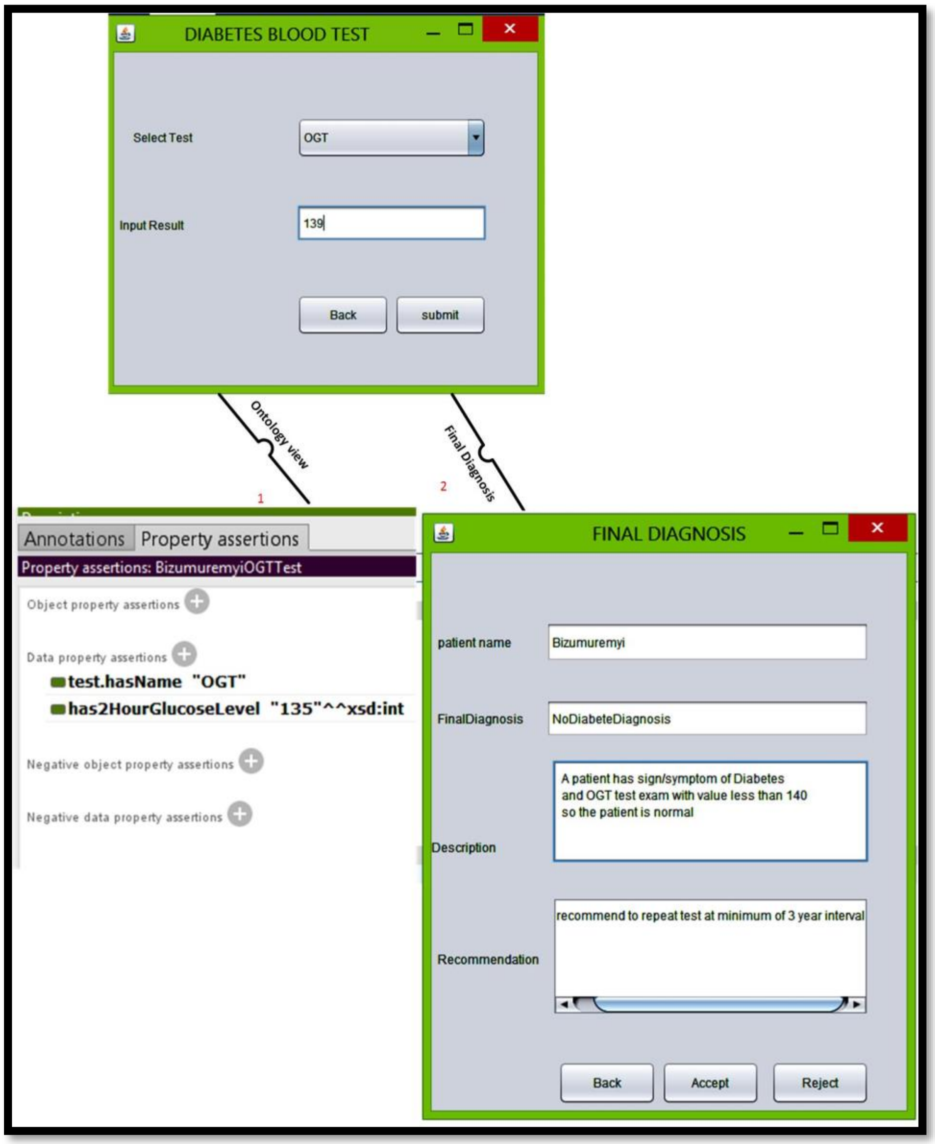
Back Avaible Test

Annotations Property assertions
Property assertions: Bizumuremyi
Object property assertions
patient.hasAllergies Milk
patient.hasOccupation Doctor
patient.hasBloodGroup A
patient.hasInsurance BizumuremyiMEDIPLAN
patient.LiveInSector Juru
hasSymptomAndSign blurredVision
patient.hasReligion Muslim
patient.hasTest BizumuremyiHBA1cTest
hasSymptomAndSign WeightLoss
patient.LiveInCountry Rwanda
hasSymptomAndSign nocturia
patient.hasTest BizumuremyiFPGTest
patient.LiveInprovince Eastern
patient.hasTest BizumuremyiOGTTest
patient.hasTest BizumuremyiRPGTest
Data property assertions
patient.hasMother "Makurata"
patient.hasLastName "Xavier"
patient.hasAge 27
patient.hasSex "Male"
patient.hasFather "Ngabonzima"
patient.hasNationId 1199080162551090
patient.hasFirstName "Bizumuremyi"
patient.hasPhoneNumber 072827272127
patient.hasMaritalStatus "Single"

Figure 4.5: patient3 data collection, formalization and differential diagnosis

4.3.2 Patient3 test result extraction, formalization and final decision retrieval

Nurse uses available test button to see the test available in the system and then selects OGT and inputs 135 mg/dl into the system. Upon submitting the result, the system suggests that Bizumuremyi has no diabetes. This is because Bizumuremyi has sign/symptom of Diabetes and OGT test exam with value less than 140. The system recommends that the test should be repeated at minimum of 3 years' interval. The selected test, its value, its ontological representation and the obtained result is shown in **Figure 4.6**.


DIABETES BLOOD TEST

Select Test: OGT

Input Result: 139

Property assertions: BizumuremyiOGTTest

Data property assertions
- test.hasName "OGT"
- has2HourGlucoseLevel "135"^^xsd:int

FINAL DIAGNOSIS

patient name: Bizumuremyi

FinalDiagnosis: NoDiabeteDiagnosis

Description: A patient has sign/symptom of Diabetes and OGT test exam with value less than 140 so the patient is normal

Recommendation: recommend to repeat test at minimum of 3 year interval

Back
submit
Ontology view
Final Diagnosis
1
2
Annotations
Property assertions
Object property assertions
Negative object property assertions
Negative data property assertions
Back
Accept
Reject

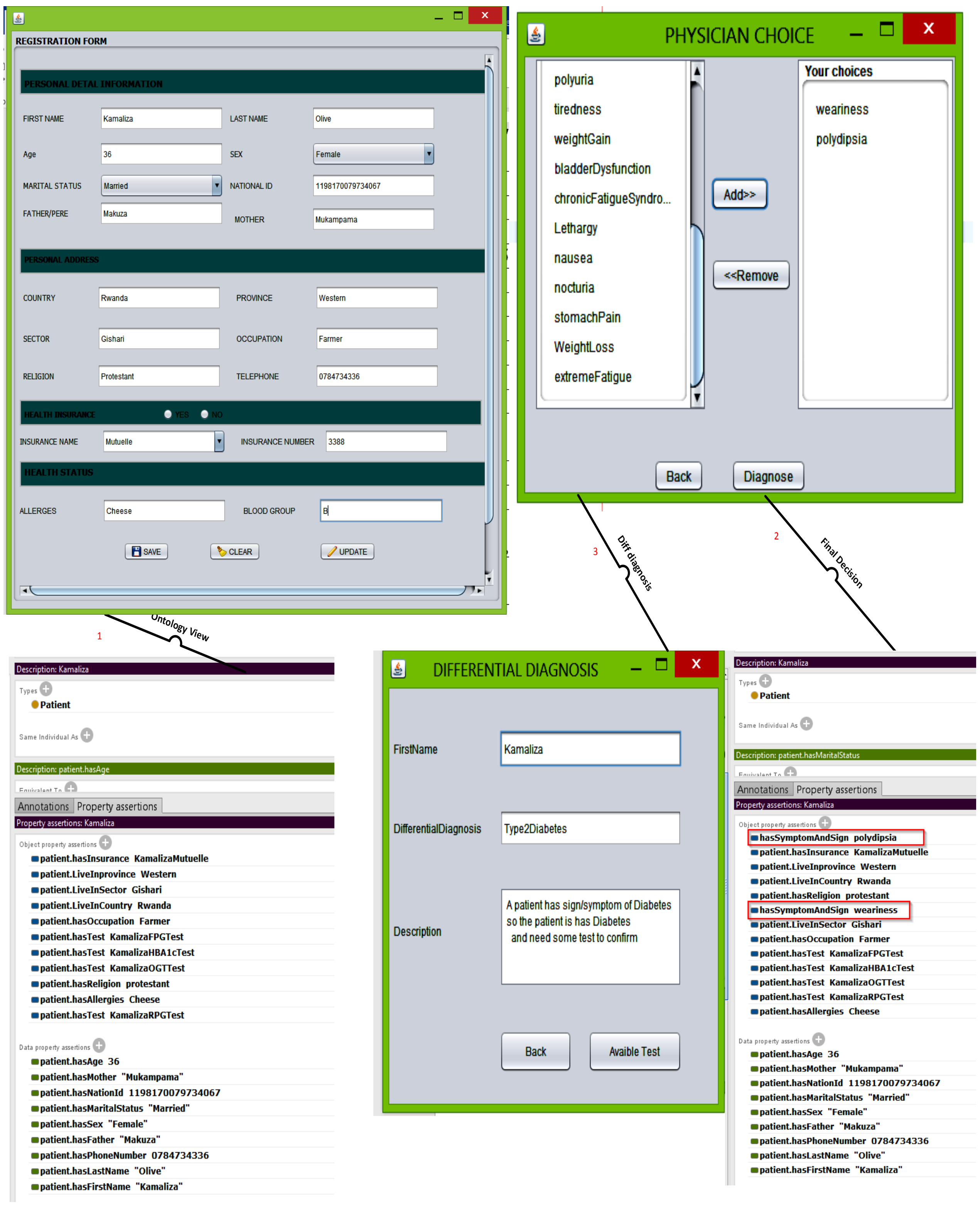
Figure 4.6: Patient3 test result extraction, formalization and final decision retrieval

4.4 Diagnosis of patient4 case

The last patient is kamaliza olive. She is a 36-year-old, Married woman. She is from Rwanda, Western province and Gishari sector. She is a farmer and uses Mutuelle insurance. She is allergic to Cheese and has B blood group. The remaining information is shown in patient filled form in **Figure 4.7**. She presents weariness and polydipsia. She has FPG test with blood glucose level of 115 mg/dl.

4.4.1 Patient4 data and vital sign extraction, formalization and differential diagnosis retrieval.

Nurse inputs patient details and vital signs into the system. Upon clicking diagnose button, the system infers that Kamaliza suffers from type2 diabetes because she is presenting sign/symptom of diabetes and suggests that some tests are needed before confirmation. The way the patient details and vital signs was inputted into the system, their ontological representation and the obtained result are shown in **Figure 4.7**.



REGISTRATION FORM

PERSONAL DETAL INFORMATION

FIRST NAME	Kamaliza	LAST NAME	Olive
Age	36	SEX	Female
MARITAL STATUS	Married	NATIONAL ID	1198170079734067
FATHER/PERE	Makuza	MOTHER	Mukampama

PERSONAL ADDRESS

COUNTRY	Rwanda	PROVINCE	Western
SECTOR	Gishari	OCCUPATION	Farmer
RELIGION	Protestant	TELEPHONE	0784734336

HEALTH INSURANCE YES NO

INSURANCE NAME	Mutuelle	INSURANCE NUMBER	3388

HEALTH STATUS

ALLERGES	Cheese	BLOOD GROUP	B

PHYSICIAN CHOICE

polyuria
tiredness
weightGain
bladderDysfunction
chronicFatigueSyndro...
Lethargy
nausea
nocturia
stomachPain
WeightLoss
extremeFatigue

Your choices

weariness
polydipsia

Description: Kamaliza

Types
Patient

Same Individual As

Description: patient.hasAge

Equivalent To

Annotations | Property assertions

Property assertions: Kamaliza

Object property assertions

- patient.hasInsurance KamalizaMutuelle
- patient.LiveInprovince Western
- patient.LiveInSector Gishari
- patient.LiveInCountry Rwanda
- patient.hasOccupation Farmer
- patient.hasTest KamalizaFPGTest
- patient.hasTest KamalizaHBA1cTest
- patient.hasTest KamalizaOGTTest
- patient.hasReligion protestant
- patient.hasAllergies Cheese
- patient.hasTest KamalizaRPGTest

Data property assertions

- patient.hasAge 36
- patient.hasMother "Mukampama"
- patient.hasNationId 1198170079734067
- patient.hasMaritalStatus "Married"
- patient.hasSex "Female"
- patient.hasFather "Makuza"
- patient.hasPhoneNumber 0784734336
- patient.hasLastName "Olive"
- patient.hasFirstName "Kamaliza"

DIFFERENTIAL DIAGNOSIS

FirstName	Kamaliza
DifferentialDiagnosis	Type2Diabetes
Description	A patient has sign/symptom of Diabetes so the patient is has Diabetes and need some test to confirm

Description: Kamaliza

Types
Patient

Same Individual As

Description: patient.hasMaritalStatus

Equivalent To

Annotations | Property assertions

Property assertions: Kamaliza

Object property assertions

- hasSymptomAndSign polydipsia
- patient.hasInsurance KamalizaMutuelle
- patient.LiveInprovince Western
- patient.LiveInCountry Rwanda
- patient.hasReligion protestant
- hasSymptomAndSign weariness
- patient.LiveInSector Gishari
- patient.hasOccupation Farmer
- patient.hasTest KamalizaFPGTest
- patient.hasTest KamalizaHBA1cTest
- patient.hasTest KamalizaOGTTest
- patient.hasTest KamalizaRPGTest
- patient.hasAllergies Cheese

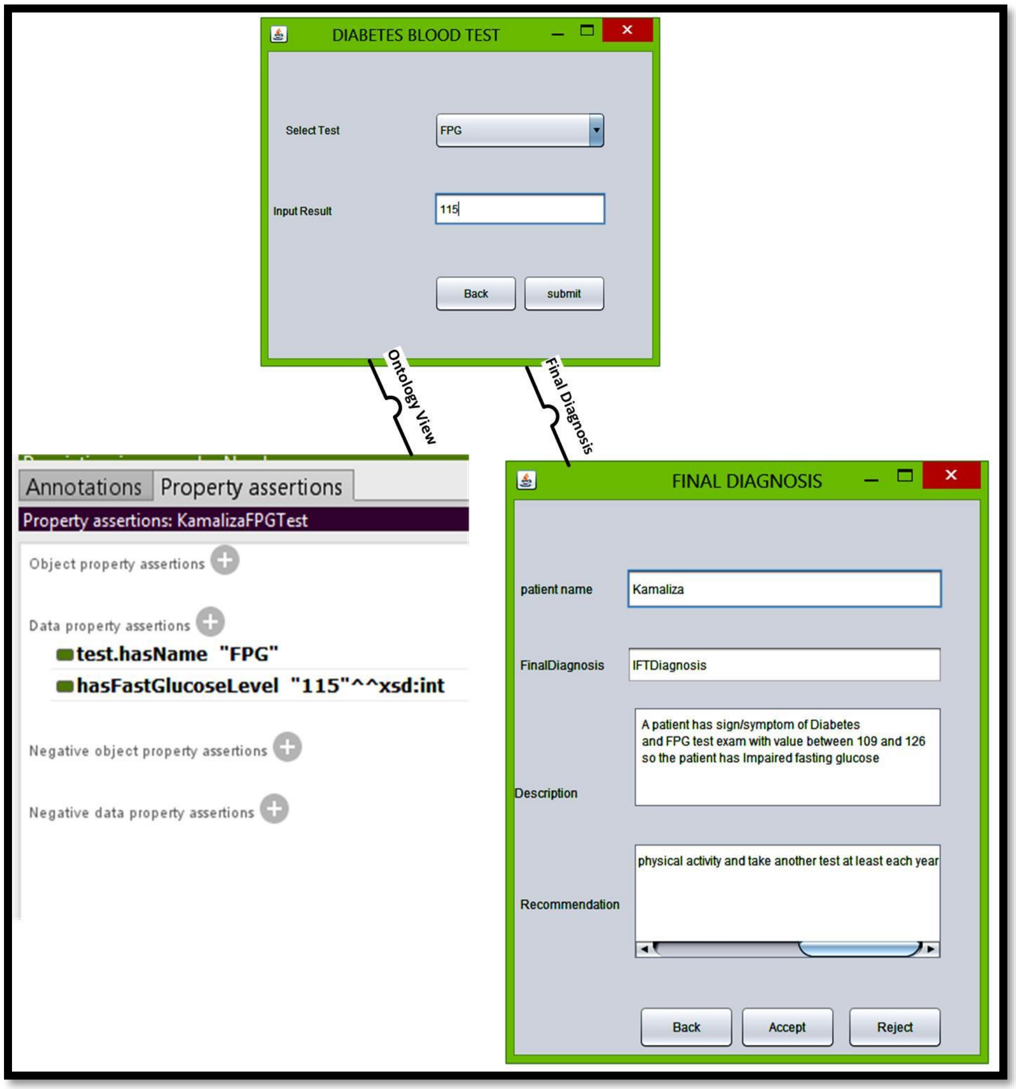
Data property assertions

- patient.hasAge 36
- patient.hasMother "Mukampama"
- patient.hasNationId 1198170079734067
- patient.hasMaritalStatus "Married"
- patient.hasSex "Female"
- patient.hasFather "Makuza"
- patient.hasPhoneNumber 0784734336
- patient.hasLastName "Olive"
- patient.hasFirstName "Kamaliza"

Figure 4.7: Patient4 data collection, formalization and differential diagnosis

4.4.2 Patient4 test result extraction, formalization and final decision retrieval

Nurse uses available test button to see the test available in the system and then selects FPG and inputs 115 mg/dl into the system. Upon submitting the result, the system suggests that Kamaliza has prediabetes (impaired fasting tolerance). This is because Kamaliza has sign/symptom of Diabetes and FPG test exam with value between 109 and 126. The system recommends that Kamaliza should take a healthy diet, improve his/her physical activity and take another test at least each year. The selected test, its value, its ontological representation and the obtained result is shown in **Figure 4.8**.



Select Test: FPG

Input Result: 115

Property assertions: KamalizaFPGTest

Data property assertions

test.hasName "FPG"

hasFastGlucoseLevel "115"^^xsd:int

patient name: Kamaliza

FinalDiagnosis: IFTDiagnosis

Description: A patient has sign/symptom of Diabetes and FPG test exam with value between 109 and 126 so the patient has Impaired fasting glucose

Recommendation: physical activity and take another test at least each year

Figure 4.8: Patient4 test result extraction, formalization and final decision retrieval

5 Discussion

In this thesis, we have developed an ontology based clinical decision support system for assisting clinician when making diagnosis of diabetes mellitus. For CDSS to be successful, it should have access to an accurate clinical data, relevant medical knowledge and problem solving procedures. In order to represent clinical data, medical knowledge and answer RQ1, the ontology was used in which diabetes mellitus related knowledge was encoded based on diagnostic criteria. Additionally, patient data was collected and formally represented in the ontology. This helps us to maintain the semantic relationships between collected patient information in the ontology (e.g vital signs, tests and test results) which in turn makes them remain interpretable regardless of surrounding technology and software implementation **[87]**. Our ontology extends diabetes mellitus diagnosis ontology (DDO) and contains key concepts and relationship required for the diabetes mellitus diagnosis, including patient and social role, symptoms, diagnosis, disease, recommendation, PatientDecisonExplanation, and so on.

After ontology construction, we have come up with ontology capable of collecting patient related information including personal details (personal identification, address, health status and insurance); patient signs and symptoms; examination tests (test name, value, unit); This information was collected through the user interfaces. A patient was created as an instance of patient class using his/her first name. The remaining personal details were added either as individual and associated to the patient through object property (e.g Religion, insurance, etc) or as a data value of data type property (first name, sex, age, etc).

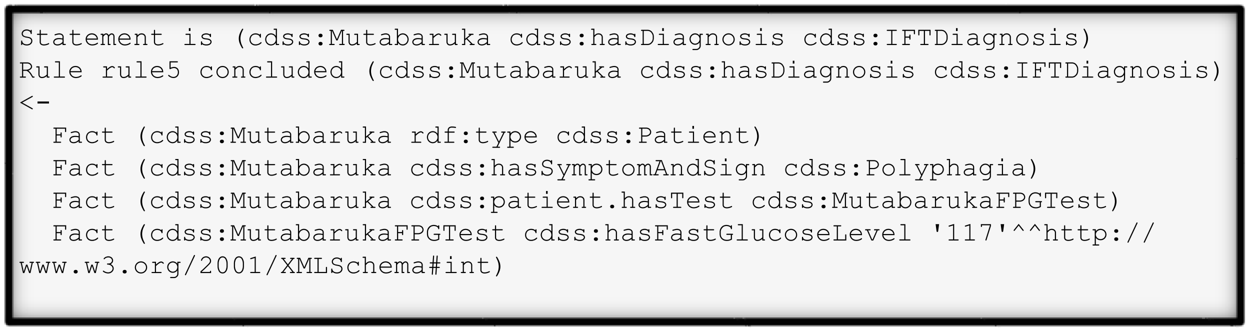
This ontology can be easily reused within other ontology based applications; patient information recorded in heterogeneous systems can be reliably linked to this ontology and generate diagnosis, recommendation and decision explanation; it can be used for statistic purpose including counting the number of patients that use insurance, number of patient that are affected by diabetes and son on. The ontology was built using protégé desktop 5.0.0, because it is easy to learn and use and fully support OWL language, there are other ontology development tools such as ontoEdit, webOnto and Swoop, etc that can be used to build ontology **[88]**. OWL-DL language was also selected because it is based on description logic and offers maximum expressiveness while maintaining computational completeness and decidability **[89]**, other ontology languages can be used based on application needs and uses. We used SPARQL to save to and retrieve information from our ontology, there other query languages that can work with RDF documents such RQL, seRQL, TRIPLE, RDQL, among others **[90]**.

For the system to behave like an expert physician, it should mimic the reasoning process used by physician when making the diagnosis. This means that it should have a reasoning component and uses the problem-solving procedures to arrive to that conclusion. To achieve this and answer RQ2, we have created decision rules that our CDSS should use to derive diagnosis, recommendation and decision explanation about individual patient's conditions and the reasoning was performed by jena rule engine. The decision rules were created based on diabetes diagnostic criteria by using jena rule syntax and were written in the text file. The advantage of writing rule in text file is that they can be reused and shared across different applications and can be updated without having to change the application code.

To produce the decision, the CDSS uses patient data contained in ontology file (this information was collected through user interface and entered by physician) and problem solving algorithms from the rules and draws conclusion based on individual patient's conditions. Our CDSS draws conclusion based on a patient vital signs and test result for blood glucose test. The conclusion is diagnosis (diabetes, prediabetes, no diabetes), recommendation suitable for the obtained diagnosis and decision explanation (why the conclusion is like that).

Our CDSS's output was similar to the result expected from the physician (i.e., in 100%, the result inferred was correct), it is produced at the point of care, presented to the intended person (physician), easy to understand and is made based on an individual patient data **[42] [41]**. Furthermore, our system's reasoning follows the reasoning process that is used by physician in diagnosis and has the whole components of an expert system **[23]**. The physician receives the system's output through user interface. Our interface is simple, effective and clearly presents the patient to whom the diagnosis is made, his/her diagnosis, the diagnosis description and recommendation.

For the system's output to be accepted by the physician, it should explain how the conclusion was reached. Jena provides the reasoning explanation using InfModel.getDerivation(Statement) method. The explanation that is generated includes the inferred statements, the facts that was asserted in the model and the rule that was used to draw the conclusion as shown in **Figure 5.1**, however, it is expensive to compute and store derived information **[73]**. Additionally, it is not easy for a physician to understand and interpret this information without having knowledge representation skills. To address this issue, diagnosis explanation was created in ontology as individuals and used in the jena rule conclusion part based on the diagnosis to be inferred by the rule engine.



```
Statement is (cdss:Mutabaruka cdss:hasDiagnosis cdss:IFTDiagnosis)
Rule rule5 concluded (cdss:Mutabaruka cdss:hasDiagnosis cdss:IFTDiagnosis)
<-
  Fact (cdss:Mutabaruka rdf:type cdss:Patient)
  Fact (cdss:Mutabaruka cdss:hasSymptomAndSign cdss:Polyphagia)
  Fact (cdss:Mutabaruka cdss:patient.hasTest cdss:MutabarukaFPGTest)
  Fact (cdss:MutabarukaFPGTest cdss:hasFastGlucoseLevel '117'^^http://
www.w3.org/2001/XMLSchema#int)
```

Figure 5.1: Example of jena inference explanation

Some CDSS confirm the existence of diabetes based on only symptoms of diabetes **[91]**, others consider one positive finding from any laboratory tests in addition to the symptoms of diabetes **[27]**, however, it is recommended to confirm diabetes after two positive findings from any two tests **[53]**. More specifically, physician should repeat the same test or select any other test. For instance, if fasting plasma glucose (FPG) is performed at the first time and the result indicates a value greater or equal to 126 mg/dl, it should be repeated or Oral glucose tolerance test (OGT) can be performed instead. If OGT result indicates a value greater or equal to 200mg/dl, diabetes is confirmed. In this thesis, we followed the same approach. A physician can select one test from four tests (FPG, OGT, RPG, A1c) at the first time and submit the result. If the result is in the range of diabetes diagnosis threshold, the system notifies the physician that another test is needed. At this time, the physician is allowed to select another test which is different from the first one in order to avoid test duplication. If the result of the second test is higher, the system confirms diabetes. Our system does not provide a decision if the result of the second test is normal. However, in normal diagnosis process, it is recommended to repeat the test that has a higher blood glucose level.

People with diabetes require a regular monitoring of their blood glucose level. They are recommended to frequently visit their physicians for checkup. In this case, a physician should perform test and see if the glucose level is normal or not. Our system does not allow an introduction of an existing test to an existing patient. That’s why it should be extended by adding time stamp to each test whenever it is performed and the diagnosis decision could be done based on the recent introduced test. Additionally, the tests were programmed within application code. Since medical knowledge keeps changing, these tests need to be encoded in ontology so that if a new test is introduced, it could be incorporated into ontology without having to change application code.

The differential diagnosis must contain a list of possible diseases as opposed to our differential diagnosis. This is because, we only have one disease (i.e diabetes mellitus) in our ontology. There is a need to develop a knowledge base that contains many diseases. In this case, probabilistic reasoning approach should be used. This approach uses prior probability formulas to calculate the probability of having diseases given their symptoms.

Now coming towards the difficulties that we have faced throughout this project. Firstly, we have spent most of our time searching over the internet so that we can understand some terms used in the medical field and the diagnostic criteria. Secondly, we have faced the problem of internet connection when conducting the interview with physicians and nurses from Rwanda. Consequently, the data collection process took longer. Finally, we have spent time creating SWRL rules and when it comes to the implementation part, we failed as we were not familiar with OWL API suitable for working with SWRL rules. Then we switched from SWRL rules to jena rules so that we can develop a java based application using Jena API, thus, pushing us to work under pressure so that the project objective can be achieved.

6 Conclusion and future work

6.1 Conclusion

Medical diagnosis is a multi-step process which is complex as it requires the consideration of many factors. Additionally, the accuracy of diagnosis varies depending on the skill and knowledge a physician has in medical field. Using ICT solution, the physicians can be assisted so that they can make an accurate decision. Today, many ontologies based applications are deployed to enhance physician performance and improve patient outcome. But some of these applications do not explain their reasoning process; their knowledge base does not build using standard medical ontologies and they build ontology but failed to develop an application that a physician can use.

In this thesis, we showed that semantic web technology can play a crucial role in patient diagnosis when the diagnostic criteria is given and well-defined. It provides semantic to biological terms which make it suitable for solving many medical informatics problems. To prove this, firstly, we have extended DDO ontology that is built using existing medical standards. Secondly, based on diabetes mellitus diagnostic criteria, we created 19 production rules by using jena rule syntax and forward chaining inference. Thirdly, we have implemented the user interfaces for extracting patient data and our system formalized and saved them into our ontology. Fourthly, our application uses jena rule engine to reason with patient data. Jena rule engine does so by matching the facts stored in the ontology file against conditions of rules in the rule file and draws correct conclusion (diagnosis, recommendation for action, and decision explanation). Finally, 35 SPARQL queries were created to retrieve these inferred conclusions and give them back to the physician through user interface. Our ontology can support the interoperability between CDSS and healthcare system. Additionally, the result of our system is useful as it accurately matches the one that were expected by physician. This system can be used as diagnostic tool for diabetes mellitus in order to reduce diagnostic errors and improve the quality of care.

6.2 Future work

The ontology based application were developed as proof of concept. The following work can be done to improve the system:

- The application is connected to an ontology that contains only one disease (diabetes mellitus) which makes it to draw one disease when making differential diagnosis; the ontology can be extended by adding so many diseases as possible and uses Bayesian reasoning approach to predict the probability of the presence possible diseases given their symptoms.

- The application suggests the patient diagnosis based on his/her vital signs and blood glucose test result (OGT, FPG, RPG, glycated hemoglobin); It can be extended by considering other diabetes aspect such as risk factors, other examination tests, etc).

- The application uses user interface to extract patient data and are filled by physicians, the tools can be proposed to import patient details from the existing health information systems or database. Additionally, web based or mobile application can be proposed so that this detail can be entered by patient before coming to the hospital.

- This application assists the diagnosis of diabetes mellitus; the same approach can be applied to other diseases.

References

[1] R. Kiran , "Developing Reliable Clinical Diagnosis Support System," 2014. [Online]. Available: http://www.kiranreddys.com/articles/clinicaldiagnosissupportsystems.pdf. [Accessed 03 01 2017].

[2] "Medical diagnosis," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Medical_diagnosis. [Accessed 04 January 2017].

[3] A. M.M and K. S, "Clinical Decision Support Systems: A discussion on different methodologies used in Health Care," sweeden, 2006.

[4] S. P. C and K. Reshmy, "OntoDiabetic: An Ontology-Based Clinical Decision Support System for Diabetic Patients," *Arabian Journal for Science and Engineering,* vol. XLI, no. 3, p. 1145–1160, 2016.

[5] . A. A.-M. AL-Gamdi, K. S. Albeladi and R. F. AlCattan, "Clinical Decision Support System in HealthCare Industry Success and Risk Factors," *International Journal of Computer Trends and Technolog,* vol. 11, no. 4, 2014.

[6] S. Weber, "Impacts of Clinical Decision Support Technology on Nursing and Medical Practice in U.S. Critical Care," *Canadian Journal of Nursing Informatics,* vol. 5, no. 4, 2011.

[7] D. Dinevski, U. Bele, T. Šarenac, U. Rajkovič and Šušteršič, "Clinical Decision Support Systems," in *TELEMEDICINE TECHNIQUES AND APPLICATIONS*, Croatia, InTech, 2011, pp. 201-226.

[8] S. Dag Vosgraff, "GenSupport. A generic guideline-based clinical decision support system," The University of Bergen, 2008.

[9] S. El-Masri and S. H. El-Sappagh, "A distributed clinical decision support system architecture," *Journal of King Saud University – Computer and Information Sciences,* vol. XXVIII, no. 1, p. 69–78, 2014.

[10] S. S Andrew, "Mathematical Foundations of Decision Support Systems," in *In Clinical Decision Support Systems*, Springer International Publishing, 2016, pp. 19-43.

[11] V. Filip , "Clinical Decision Support for screening, diagnosis and assessment of respiratory diseases: Chronic Obstructive Pulmonary Disease as a use case," 2016.

[12] Chang, Ying-Jui, Min-Li Yeh, Chyou-Shen Lee, Chien-Yeh Hsu, Yu-Chuan Jack Li, and Wen-Ta Chiu, "Cross-domain probabilistic inference in a clinical decision support system: Examples for dermatology and rheumatology," *Computer methods and programs in biomedicine,* vol. CIV, no. 2, p. 286–291, 2011.

[13] M. Malhotra and T. G. Nair, "Evolution of Knowledge Representation and Retrieval Techniques," *International Journal of Intelligent Systems and Applications,* vol. 7, no. 7, pp. 18-28, 2015.

[14] M. L. J, S. A, K. J, M. D.M and M. T.G, "User centered clinical decision support tools: adoption across clinician training level.," *Appl Clin Inform,* vol. V, no. 4, pp. 1015-25, 2014.

[15] H. Aworinde, F. O Alamu and O. Obinna, "DEVELOPMENT OF A SEMANTIC ONTOLOGY FOR MALARIA DISEASE USING PROTÉGÉ-OWL SOFTWARE," *Journal of Multidisciplinary Engineering Science and Technology (JMEST),* vol. I, no. 3, 2014.

[16] S. R and A. J, "A survey on ontology construction methodologies," *International Journal of Enterprise Computing and Business Systems,* vol. 1, no. 1, pp. 60-72, 2011.

[17] F. Blanca, "Design and development of a decision support system for screening of Lynch syndrome using openEHR," Karolinska Institutet, 2015.

[18] J. Kannry, L. McCullagh, A. Kushniruk and D. Mann, "A Framework for Usable and Effective Clinical Decision Support: Experience from the iCPR Randomized Clinical Trial," *eGEMs,* vol. 3, no. 2, 2015.

[19] B. Eta S., "Clinical Decision Support Systems: State of the Art," *Agency for healthcare research and quality Advancing excellence on health care*, 2009.

[20] F. Velickovsk, L. Ceccaroni, J. Roca, F. Burgos, J. B Galdiz, N. Marina and M. Lluch-Ariet, "Clinical Decision Support Systems (CDSS) for preventive management of COPD patients," *Journal of translational medicine,* vol. XII, no. 2, p. S9, 2014.

[21] K. Kawamoto, J. Jacobs, B. M. Welch, V. Huser, M. D. Paterno, G. D. Fiol, D. Shields, H. R. Strasberg, P. J. Haug, Z. Liu, R. A. Jenders, D. W. Rowed, D. Chertcoff, K. Fehre, K.-P. Adlassnig and C. A. Clayton , "Clinical information system services and capabilities desired for scalable, standards-based, service-oriented decision support: Consensus assessment of the health level 7 clinical decision support work group.," *In AMIA Annual Symposium Proceedings,* vol. 2012, pp. 446-455, 2012.

[22] A. Sen, A. Banerjee, A. P. Sinha and M. Bansal, "Clinical decision support: Converging toward an integrated architecture," *Journal of Biomedical Informatics,* vol. XLV, no. 5, p. 1009–1017, 2012.

[23] R.-C. Chen, Y.-H. Huang, C.-T. Bau and S.-M. Chen, "A recommendation system based on domain ontology and SWRL for anti-diabetic drugs selection," *Expert Systems with Applications,* vol. 39, no. 4, pp. 3995-4006, 2012.

[24] S. SM , G. A , N. G , Z. XJ , . K. J and J. A , "Decision support systems for clinical radiological practice towards the next generation.," *The British journal of radiology.,* 2014.

[25] O. Bunyamin , H. J Michael and C. David C, "Data Mining and Clinical Decision Support," in *In Clinical Decision Support Systems* , Springer International Publishing, 2016, pp. 45-68.

[26] C. B. , J. R. Josephson and V. R. Benjamins, "What are ontologies, and why do we need them?," *IEEE Intelligent Systems and their applications,* vol. 14, no. 1, pp. 20-26, 1999.

[27] R. F. Alharbi, J. Berri and S. El-Masri, "Ontology based clinical decision support system for diabetes diagnostic," in *In Science and Information Conference*, London, 2015.

[28] C.-T. Bau, R.-C. Chen and C.-Y. Huang, "Construction of a Clinical Decision Support System for Undergoing Surgery Based on Domain Ontology and Rules Reasoning," *TELEMEDICINE and e-HEALTH,* vol. 20, no. 5, pp. 460-472, 2014.

[29] S. El-Sappagha and M. Elmogy, "A fuzzy ontology modeling for case base knowledge in diabetes mellitus domain," *Engineering Science and Technology, an International Journal,* 2017.

[30] . S. El-Sappagh and A. Farman , "DDO: a diabetes mellitus diagnosis ontology," *In Applied Informatics,* vol. 3, no. 1, p. 5, 2016.

[31] M. Eneida A. , "Clinical decision support systems: perspectives in dentistry," *Journal of dental education,* vol. 68, no. 6, pp. 589-597, 2004.

[32] T. Anagnostou, M. Remzi and B. Djavan, "Artificial neural networks for decision-making in urologic oncology," *European urology,* vol. 43, no. 6, pp. 596-603, 2003.

[33] M. Thirugnanam, P. Kumar, V. S. Srivatsan and N. C.R, "Improving the Prediction Rate of Diabetes Diagnosis Using Fuzzy, Neural Network, Case Based (FNC) Approach," *Procedia Engineering,* vol. 38, pp. 1709-1718, 2012.

[34] K. Manoj , S. Anubha and A. Sonali , "Clinical decision support system for diabetes disease diagnosis using optimized neural network," *In Engineering and Systems (SCES),* pp. 1-6, 2014.

[35] R. K. Saripalle, "Current status of ontologies in Biomedical and Clinical informatics.," *International Journal of Science and Information.,* 2010.

[36] O. Bodenreider, "Biomedical ontologies in action: role in knowledge management, data integration and decision support," *IMIA Yearbook,* p. 67–79., 2008.

[37] S. Richard H and W. C. Barry S, "Toward an ontological treatment of disease and diagnosis.," *Proceedings of the 2009 AMIA summit on translational bioinformatics 2009,* pp. 116-120, 2009.

[38] B.-G. Valérie, B. Anita and D. Régis, "Ontology and medical diagnosis," *Informatics for Health and Social Care,* vol. 37, no. 2, pp. 51-61, 2012.

[39] T. Jodie A , M. Susana B , M. Martha C , W. Dan, T. Samson W , C. David J , E. Jan , V. Brigit Vucic, B. Steve , C. Michael E , S. Charles D , R. Jack , D. Denise and G. Mary K , "Designing an automated clinical decision support system to match clinical practice guidelines for opioid therapy for chronic pain," *Implementation Science,* vol. 5, no. 1, p. 26, 2010.

[40] M. Pusic and M. Ansermino, "Clinical decision support systems," *British Columbia Medical Journal,* vol. 46, no. 5, pp. 236-239, 2004.

[41] P. Meenal B, K. Kensaku , L. David , P. Uptal D and M. David B, "Recommendations for a Clinical Decision Support for the Management of Individuals with Chronic Kidney Disease," *Clinical Journal of the American Society of Nephrology,* vol. 4, no. 2, pp. 273-283, 2009.

[42] H. Jeff , "Integrating Clinical Decision Support Tools into Ambulatory Care Workflows for Improved Outcomes and Patient Safety," *Qualis Health,* pp. 15-16, 2013.

[43] K. Kensaku, C. A Houlihan, B. E Andrew and L. D.F, "Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success," *Bmj,* vol. 330, no. 7494, p. 765, 2005.

[44] I. SIM, P. GORMAN, G. ROBERT A. , H. R. BRIAN , K. BONNIE , L. HAROLD and T. PAUL C, "Clinical decision support systems for the practice of evidence-based medicine," *Journal of the American Medical Informatics Association,* vol. 8, no. 6, pp. 527-534, 2001.

[45] B. Patrick Emanuel , B. David Westfall and H. Balthasar Luzius, "Clinical decision support systems," *Swiss Med Wkly*, vol. 144, p. w14073, 2014.

[46] W. Adam and S. Dean F, "A framework and model for evaluating clinical decision support architectures," *Journal of Biomedical Informatics,* vol. 41, no. 6, p. 982–990, 2008.

[47] O. World Health, "Global report on diabete," World Heath Organization, france, 2016.

[48] A. American Diabetes , "Screening for Diabetes," *American Diabetes Association Diabetes Care,* vol. 25, no. 1, pp. S21-S24, 2002.

[49] F. Cynthia M , K. Emma W and S. Jean R , "Clinicians' attitudes to clinical practice guidelines: a systematic review," *Medical Journal of Australia,* vol. 177, no. 9, pp. 502-506, 2002.

[50] T. Astrid van and W. Ulrich , "Diagnosis and classification in spondyloarthritis: identifying a chameleon," *Nature Reviews Rheumatology,* vol. 8, no. 5, pp. 253-261, 2012.

[51] A. American Diabetes , "2. Classification and diagnosis of diabetes.," vol. 38, no. 1, pp. S8-S16., 2015.

[52] A. American Diabetes , "Diagnosing Diabetes and Learning About Prediabetes," American Diabetes Associations, 21 November 2016. [Online]. Available: http://www.diabetes.org/diabetes-basics/diagnosis/?referrer=https://www.google.no/. [Accessed 06 February 2017].

[53] M. JENNIFER , "Diagnosis and Classification of Diabetes Mellitus: New Criteria," *America Familly physician*, vol. 58, no. 6, pp. 1355-1362, 1998.

[54] B. Femke De , M. Hendrik , S. Kristof , C. Kirsten , D. Johan and T. Filip De , "Towards automated generation and execution of clinical guidelines: Engine design and implementation through the ICU Modified Schofield use case," *Computers in biology and medicine*, vol. 42, no. 8, pp. 793-805, 2012.

[55] S. Rachna and K. Gurmeet , "Basic Approaches to Semantic Web Services : A Comparative Study," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 3, 2012.

[56] D. Nassim , V. Jean Marc and R. Jos De , "New Semantic Web rules and new medical reasoning framework," in *In Instrumentation and Measurement Technology Conference (I2MTC)*, 2013.

[57] "Semantic Web Stack," Wikipedia, the free encyclopedia, 17 September 2015. [Online]. Available: https://en.wikipedia.org/wiki/Semantic_Web_Stack. [Accessed 04 April 2017].

[58] A. Dean and H. James , "RDF-the basis of sementic web," in *Semantic Web for the Working Ontologist : Effective Modeling in RDFS and OWL*, USA, Morgan Kaufmann, 2011, pp. 27-50.

[59] A. Dean and H. James , "RDF schema," in *Semantic Web for the Working Ontologist : Effective Modeling in RDFS and OWL*, Burlington, MA: Morgan Kaufmann, 2012, pp. 125-152.

[60] B. Tiffani J , F. E Yoko , K. Gilad J and B. Suzanne , "Development and evaluation of an ontology for guiding appropriate antibiotic prescribing," *Journal of Biomedical Informatics*, vol. 45, no. 1, p. 120–128, 2012.

[61] H. Ian and P.-S. Peter F, "KR and Reasoning on the the semantic web: OWL," in *Handbook of Semantic Web Technologies*, New Jersey, Springer, 2011, pp. 365-398.

[62] S. Andy and P. Eric , "SPARQL Query Language for RDF," W3C, 26 March 2013. [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/. [Accessed 15 March 2017].

[63] H. Jiewen , A. Daniel J and R. Kun , "Scalable SPARQL querying of large RDF graphs," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 1123-1134., 2011.

[64] "Tutorial 5: Querying Semantic Data," LinkedDataTools.com , [Online]. Available: http://www.linkeddatatools.com/querying-semantic-data. [Accessed 15 March 2017].

[65] H. Ian , P.-S. Peter F, B. Harold , T. Said , G. Benjamin and D. Mike , "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," 19 November 2003. [Online]. Available: http://www.daml.org/2003/11/swrl/rules-all.html. [Accessed 08 February 2017].

[66] M. O'Connor, N. Csongor , S. Ravi , D. Amar and M. Mark , "The SWRLAPI: A Development Environment for Working with SWRL Rules.," *OWLED,* 2008.

[67] Zippyrate, "SWRLDroolsTab," 3 April 2015. [Online]. Available: https://github.com/protegeproject/swrlapi-drools-engine/wiki/SWRLDroolsTab. [Accessed 15 March 2017].

[68] M. O'Connor, K. Holger , T. Samson , G. Benjamin , D. Mike, G. William and M. Mark , "Supporting rule system interoperability on the semantic web with SWRL," *International Semantic Web Conference,* pp. 974-986, 205.

[69] M. Alves , D. Carlos Viegas and C. Nuno , "SPARQL commands in Jena rules," in *International Conference on Knowledge Engineering and the Semantic Web*, 2015.

[70] "Apache Jena," The Apache Software Foundation, [Online]. Available: http://jena.apache.org/. [Accessed 24 02 2017].

[71] D. Ion-Mircea Diaconescu, L. Sergey and G. Adrian , "Semantic web and rule reasoning inside of e-learning systems," *Advances in intelligent and distributed computing,* pp. 251-256, 2008.

[72] S. Tilotma , T. Navneet and K. Deepali , "STUDY OF DIFFERENCE BETWEEN FORWARD AND BACKWARD REASONING," *International Journal of Emerging Technology and Advanced Engineering,* vol. 10, no. 2, pp. 271-273, 2012.

[73] "Reasoners and rule engines: Jena inference support," The Apache Software Foundation, [Online]. Available: https://jena.apache.org/documentation/inference/. [Accessed 12 February 2017].

[74] S. Holger , "Basic Formal Ontology," Institute for Formal Ontology and Medical Information Science (IFOMIS), 31 March 2017. [Online]. Available: http://ifomis.uni-saarland.de/bfo/. [Accessed 25 April 2017].

[75] S. Barry, "The Ontology for General Medical Science," SlideShare, 02 May 2013. [Online]. Available: https://www.slideshare.net/BarrySmith3/the-ontology-for-general-medical-science. [Accessed 13 April 2017].

[76] D. Kh Rekha , S. A.M and H. K, "A working Framework for the User-Centered Design Approach and a Survey of the available Methods," *International Journal of Scientific and Research Publications,* vol. 2, no. 4, pp. 1-8, 2012.

[77] N. Natalya F and M. Deborah L, "Ontology development 101: A guide to creating your first ontology," *Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880,* vol. 15, no. 2, 2001.

[78] M. MARTIN , "Methods to support human-centred design," *International journal of human-computer studies,* vol. 55, no. 4, pp. 587-634, 2001.

[79] "The mobile phone – Rwanda’s key weapon in making maternal deaths history," unicefstories , 13 November 2013. [Online]. Available: http://unicefstories.org/2013/11/13/the-mobile-phone-rwandas-key-weapon-in-making-maternal-deaths-history/. [Accessed 13 February 2017].

[80] R. Suzanne and R. James , mastering the requirement process, Addison-Wesley, 2012.

[81] C. Alessandra , D. Jim , J. Thierry , M. Laurent , H. Alan and O. Sergey , "Using UML for Automatic Test Generation," in *In Proceedings of ISSTA*, 2002.

[82] "Visio Professional 2016," Microsoft, [Online]. Available: https://products.office.com/en/visio/visio-professional-business-and-diagram-software. [Accessed 07 April 2017].

[83] K. Aditya , H. Bina Bestina and S. Surya , "UML tools collaboration for use case, class and activity diagram implemented with html 5 and java script framework," *Journal of Computer Science,* vol. 10, no. 9, pp. 1440-1446, 2014.

[84] B. Sabin Corneliu , C. Liliana and N. Ovidiu Cătălin , "Survey on Web Ontology Editing Tools," *Transactions on Automatic Control and Computer Science,* pp. 1-6, 2006.

[85] J. Kruti and C. V.M , "A Study on Semantic Web Framework: JENA and Protégé," *Indian Journal of Applied research,* vol. 4, no. 1, 2014.

[86] M. o. health, "Outpatient file," [Online]. Available: http://www.moh.gov.rw/fileadmin/templates/Clinical/OUTPATIENT-FILE.pdf. [Accessed 24 January 2017].

[87] B. Matt-Mouley , R. Alan and H. Martin , "Using ontologies for an intelligent patient modelling, adaptation and management system," *On the Move to Meaningful Internet Systems: OTM 2008,* pp. 1458-1470, 2008.

[88] A. Sunitha and B. Suresh , "Survey on Ontology Construction Tools," *International Journal of Scientific & Engineering Research,* vol. 4, pp. 1748-1752, 2013.

[89] R. David , R. Francis , . L.-V. Joan Albert, C. Fabio , E. Sara , M. Patrizia , A. Roberta and C. Carlo , "An ontology-based personalization of health-care knowledge to support clinical decisions for chronically ill patients," *Journal of biomedical informatics,* vol. 45, no. 3, pp. 429-446, 2012.

[90] H. Peter , B. Jeen , E. Andreas and V. Raphael , "A comparison of RDF query languages," in *International Semantic Web Conference.,* Springer Berlin Heidelberg, 2004.

[91] M. Piyush , S. D.B.V, R. Nagendra Singh and S. Shailendra, "Clinical Decision Support System for Diabetes Disease Diagnosis," *International Journal of Engineering Research and Applications (IJERA).*

Appendix

Appendix1

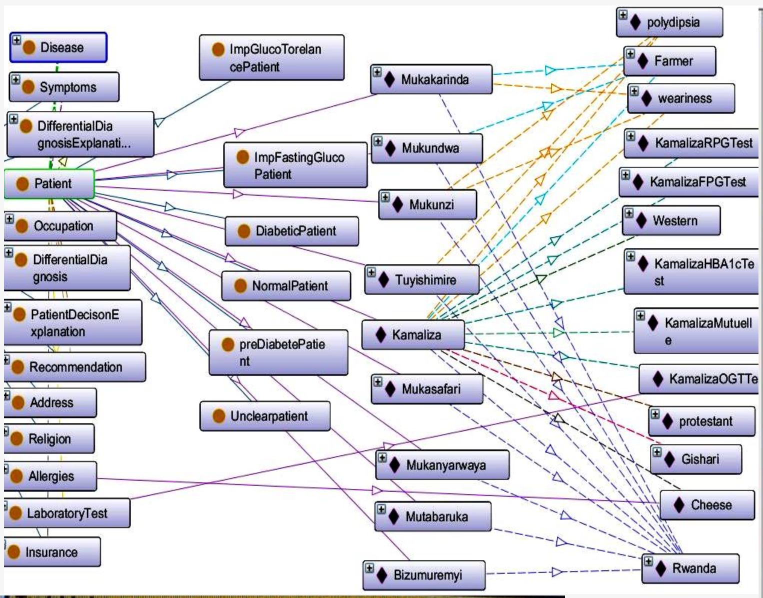
Table 0-1: Example of jena rules

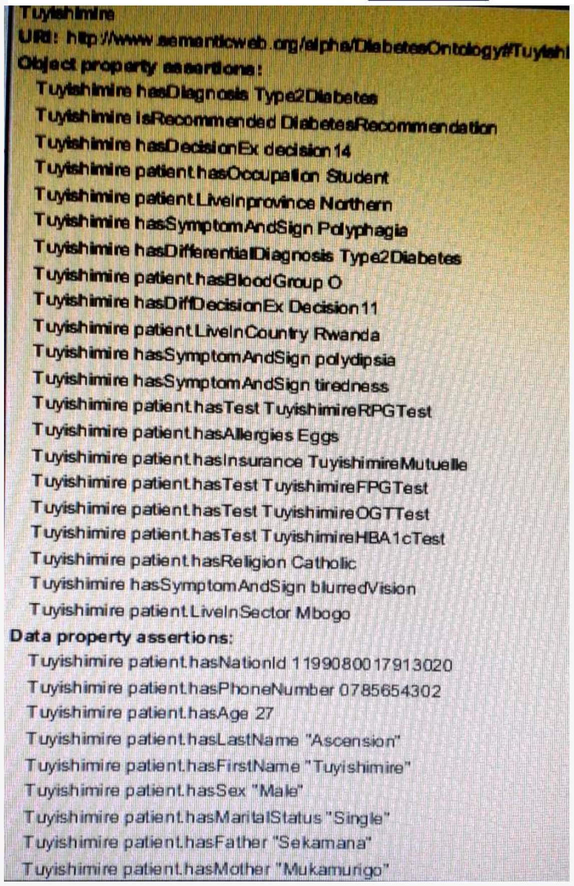
No	Rule
1	[rule1:(?Patient rdf:type :Patient)(?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test)(?test :hasAc1percentage ?value)ge(?value,'6.6'^^xsd:float)(?patient :patient.hasTest ?test1)(?test1:hasAc1percentage?value1)ge(?value1,'6.6'^^xsd:float) ->(?patient :hasDiagnosis:Type2Diabetes)(?Patient rdf:type :DiabeticPatient)(?patient :hasDecisionEx :decision1)(?patient :isRecommended :DiabetesRecommendation)]
2	[rule2:(?patient rdf:type :Patient)(?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test)(?test :hasAc1percentage ?value)le(?value,'5.7'^^xsd:float)->(?patient :hasDiagnosis :NoDiabeteDiagnosis)(?patient rdf:type :NormalPatient)(?patient :hasDecisionEx :Decision2)(?patient :isRecommended :NormalRecommendation)]
3	[rule3:(?patient rdf:type :Patient)(?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test)(?test:hasAc1percentage ?value)ge(?value,'5.6'^^xsd:float)le(?value,'6.7'^^xsd:float)-> (?patient :hasDiagnosis :preDiabetesDiagnosis) (?patient rdf:type :preDiabetePatient)(?patient :hasDecisionEx :Decision3) (?patient :isRecommended :PrediabetesRecommendation)]
4	[rule4:(?patient rdf:type :Patient) (?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test) (?test :hasFastGlucoseLevel ?value)le(?value,'110'^^xsd:int)->(?patient :hasDiagnosis :NoDiabeteDiagnosis)(?patient rdf:type :NormalPatient)(?patient :hasDecisionEx :Decision4)(?patient :isRecommended :NormalRecommendation)]
5	[rule5:(?patient rdf:type :Patient)(?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test)(?test :hasFastGlucoseLevel ?value)ge(?value,'109'^^xsd:int)le(?value, '126'^^xsd:int)-> (?patient :hasDiagnosis :IFTDiagnosis) (?patient rdf:type :ImpFastingGlucoPatient)(?patient :hasDecisionEx :Decision5) (?patient :isRecommended :PrediabetesRecommendation)]
6	[rule6:(?patient rdf:type :Patient)(?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test)(?test :has2HourGlucoseLevel ?value)ge(?value,'139'^^xsd:int)le(?value,'200'^^xsd:int)-> (?patient :hasDiagnosis :IGTDiagnosis)(?patient rdf:type :ImpGlucoTorelancePatient)(?patient :hasDecisionEx :Decision7) (?patient :isRecommended :PrediabetesRecommendation)]
7	[rule7:(?patient rdf:type :Patient)(?patient :hasSymptomAndSign ?sign) (?patient :patient.hasTest ?test) (?test :has2HourGlucoseLevel ?value)le(?value,'140'^^xsd:int)->(?patient

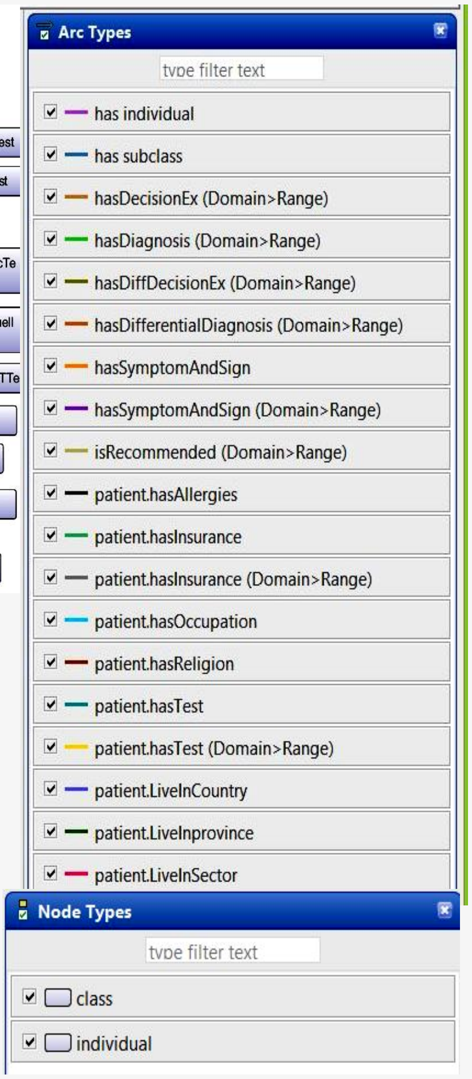
	:hasDiagnosis :NoDiabeteDiagnosis)(?patient rdf:type :NormalPatient)(?patient :hasDecisionEx :Decision6)(?patient :isRecommended :NormalRecommendation)]
8	[rule8:(?patient rdf:type :Patient)(?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test)(?test :hasFastGlucoseLevel ?value)ge(?value,'125'^^xsd:int)(?patient :patient.hasTest ?test1)(?test1 :has2HourGlucoseLevel ?value1)ge(?value1, '199'^^xsd:int)->(?patient :hasDiagnosis :Type2Diabetes)(?patient rdf:type :DiabeticPatient)(?patient :hasDecisionEx :Decision8) (?patient :isRecommended :DiabetesRecommendation)]
9	[rule9:(?patient rdf:type :Patient)(?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test)(?test :hasFastGlucoseLevel ?value)ge(?value,'125'^^xsd:int)(?patient :patient.hasTest ?test1)(?test1 :hasRandomGlucoseLevel ?value1)ge(?value1, '199'^^xsd:int)->(?patient :hasDiagnosis :Type2Diabetes) (?patient rdf:type :DiabeticPatient)(?patient :hasDecisionEx :Decision9) (?patient :isRecommended :DiabetesRecommendation)]
10	[rule10:(?patient rdf:type :Patient)(?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test)(?test :hasRandomGlucoseLevel ?value)ge(?value, '199'^^xsd:int)(?patient: patient.hasTest ?test1)(?test1 :has2HourGlucoseLevel ?value1)ge(?value1,'199'^^xsd:int)->(?patient :hasDiagnosis :Type2Diabetes) (?patient rdf:type :DiabeticPatient)(?patient :hasDecisionEx :Decision10) (?patient :isRecommended :DiabetesRecommendation)]
11	[rule11:(?patient rdf:type :Patient)(?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test)(?test :hasAc1percentage ?value)ge(?value, '6.6'^^xsd:float)(?patient :patient.hasTest ?test1)(?test1 :has2HourGlucoseLevel ?value1)ge(?value1, '199'^^xsd:int)->(?patient :hasDiagnosis :Type2Diabetes)(?patient rdf:type :DiabeticPatient)(?patient :hasDecisionEx :Decision12) (?patient :isRecommended :DiabetesRecommendation)]
12	[rule12:(?patient rdf:type :Patient)(?patient: hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test) (?test :hasAc1percentage ?value)ge(?value, '6.6'^^xsd:float)(?patient :patient.hasTest ?test1)(?test1 :hasRandomGlucoseLevel ?value1)ge(?value1, '199'^^xsd:int)->(?patient :hasDiagnosis :Type2Diabetes)(?patient rdf:type :DiabeticPatient)(?patient :hasDecisionEx :Decision13) (?patient :isRecommended :DiabetesRecommendation)]
13	[rule13:(?patient rdf:type :Patient)(?patient:hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test) (?test :hasAc1percentage ?value)ge(?value, '6.6'^^xsd:float)(?patient :patient.hasTest ?test1) (?test1 :hasFastGlucoseLevel ?value1)ge(?value1, '125'^^xsd:int)->(?patient :hasDiagnosis :Type2Diabetes)(?patient rdf:type :DiabeticPatient)(?patient :hasDecisionEx :decision14) (?patient :isRecommended :DiabetesRecommendation)]

14	[rule14:(?patient rdf:type :Patient)(?patient :hasSymptomAndSign ?sign)->(?patient :hasDifferentialDiagnosis :Type2DiabeteDF) (?patient :hasDiffDecisionEx :Decision11)]
15	[rule15: (?patient rdf:type :Patient) (?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test) (?test :hasFastGlucoseLevel ?value)ge(?value, '125'^^xsd:int)(?patient :patient.hasTest ?test1) (?test1 :hasFastGlucoseLevel ?value1)ge(?value1, '125'^^xsd:int)-> (?patient :hasDiagnosis :Type2Diabetes) (?patient rdf:type :DiabeticPatient)(?patient :hasDecisionEx :decision15)(?patient :isRecommended :DiabetesRecommendation)]
16	[rule16: (?patient rdf:type :Patient) (?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test) (?test :hasRandomGlucoseLevel ?value)ge(?value, '199'^^xsd:int)(?patient :patient.hasTest ?test1) (?test1 :hasRandomGlucoseLevel ?value1)ge(?value1, '199'^^xsd:int)-> (?patient :hasDiagnosis :Type2Diabetes) (?patient rdf:type :DiabeticPatient)(?patient :hasDecisionEx :Decision16)(?patient :isRecommended :DiabetesRecommendation)]
17	[rule17: (?patient rdf:type :Patient) (?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test) (?test :has2HourGlucoseLevel ?value)ge(?value, '199'^^xsd:int)(?patient :patient.hasTest ?test1) (?test1 :has2HourGlucoseLevel ?value1)ge(?value1, '199'^^xsd:int)-> (?patient :hasDiagnosis :Type2Diabetes) (?patient rdf:type :DiabeticPatient)(?patient :hasDecisionEx :Decision17)(?patient :isRecommended :DiabetesRecommendation)]
18	[rule18:(?patient rdf:type :Patient)(?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test)(?test :hasAc1percentage ?value)ge(?value,'6.6'^^xsd:float)(?patient :patient.hasTest ?test1)(?test1 :hasAc1percentage ?value1)ge(?value1, '6.6'^^xsd:float)->(?patient :hasDiagnosis :Type2Diabetes) (?patient rdf:type :DiabeticPatient)(?patient :hasDecisionEx :decision18)(?patient :isRecommended :DiabetesRecommendation)]
19	[rule19:(?patient rdf:type :Patient)(?patient :hasSymptomAndSign ?sign)(?patient :patient.hasTest ?test)(?test :hasRandomGlucoseLevel ?value1)le(?value1,'200'^^xsd:int)-> (?patient :hasDiagnosis :NoDiabeteDiagnosis)(?patient rdf:type :NormalPatient)(?patient :hasDecisionEx :Decision19)(?patient :isRecommended :NormalRecommendation)]

Appendix2




Tuyishimire
URI: http://www.semanticweb.org/alpha/DiabetesOntology#Tuyish
Object property assertions:
Tuyishimire hasDiagnosis Type2Diabetes
Tuyishimire isRecommended DiabetesRecommendation
Tuyishimire hasDecisionEx decision14
Tuyishimire patient.hasOccupation Student
Tuyishimire patient.LiveInprovince Northern
Tuyishimire hasSymptomAndSign Polyphagia
Tuyishimire hasDifferentialDiagnosis Type2Diabetes
Tuyishimire patient.hasBloodGroup O
Tuyishimire hasDiffDecisionEx Decision11
Tuyishimire patient.LiveInCountry Rwanda
Tuyishimire hasSymptomAndSign polydipsia
Tuyishimire hasSymptomAndSign tiredness
Tuyishimire patient.hasTest TuyishimireRPGTest
Tuyishimire patient.hasAllergies Eggs
Tuyishimire patient.hasInsurance TuyishimireMutuelle
Tuyishimire patient.hasTest TuyishimireFPGTest
Tuyishimire patient.hasTest TuyishimireOGTTest
Tuyishimire patient.hasTest TuyishimireHBA1cTest
Tuyishimire patient.hasReligion Catholic
Tuyishimire hasSymptomAndSign blurredVision
Tuyishimire patient.LiveInSector Mbogo
Data property assertions:
Tuyishimire patient.hasNationId 1199080017913020
Tuyishimire patient.hasPhoneNumber 0785654302
Tuyishimire patient.hasAge 27
Tuyishimire patient.hasLastName "Ascension"
Tuyishimire patient.hasFirstName "Tuyishimire"
Tuyishimire patient.hasSex "Male"
Tuyishimire patient.hasMaritalStatus "Single"
Tuyishimire patient.hasFather "Sekamana"
Tuyishimire patient.hasMother "Mukamurigo"



- ✓ has individual
- ✓ has subclass
- ✓ hasDecisionEx (Domain>Range)
- ✓ hasDiagnosis (Domain>Range)
- ✓ hasDiffDecisionEx (Domain>Range)
- ✓ hasDifferentialDiagnosis (Domain>Range)
- ✓ hasSymptomAndSign
- ✓ hasSymptomAndSign (Domain>Range)
- ✓ isRecommended (Domain>Range)
- ✓ patient.hasAllergies
- ✓ patient.hasInsurance
- ✓ patient.hasInsurance (Domain>Range)
- ✓ patient.hasOccupation
- ✓ patient.hasReligion
- ✓ patient.hasTest
- ✓ patient.hasTest (Domain>Range)
- ✓ patient.LiveInCountry
- ✓ patient.LiveInprovince
- ✓ patient.LiveInSector

- ✓ class
- ✓ individual

Figure 0.1: Part of the ontology in ontograph visualization