



# **COMPARATIVE STUDY OF OPEN SOURCE AND DOT NET ENVIRONMENTS FOR ONTOLOGY DEVELOPMENT**

Dissertation Submitted in Fulfillment of the Requirements for the Degree of  
**MAGISTER TECHNOLOGIAE:**

In the

Department of Information and Communication Technology, Faculty of  
Applied and Computer Sciences, Vaal University of Technology

By

Leki Jovial Mahoro

216165393

**Supervisor:** Dr J.V. Fonou-Dombeu

**Co-Supervisor:** Mrs. Sihle Moyo

**May, 2020**

# DECLARATION

I hereby declare that this dissertation, which I submit for the qualification of **Magister Technologiae in Information Technology** to the Vaal University of Technology, Department of Information and Communications Technology, Faculty of Applied and Computer Sciences, apart from the recognized assistance of my supervisor and provided citations, is my own work and has not previously been submitted to any other institution for any degree.

Leki Jovial Mahoro on this \_\_\_\_\_ day of \_\_\_\_\_  
Candidate

Dr. J.V. Fonou-Dombeu on this \_\_\_\_\_ day of \_\_\_\_\_  
Supervisor

# **ACKNOWLEDGEMENTS**

I would like to express my gratitude to my supervisor Dr J.V. Fonou-Dombeu for the useful comments, remarks and full guidance through the entire process of this Master dissertation. I would like to thank my mother for encouragement on the way and my brother Steven Manzi, who have supported me throughout entire process of my studies, both by keeping me harmonious and helping me putting pieces together. I owe my loving thanks to Guillaîne and Ngango's family for their motivation and advice. I will forever be grateful for your support

# ABSTRACT

Many studies have evaluated and compared the existing open-sources Semantic Web platforms for ontologies development. However, none of these studies have included the dot NET-based semantic web platforms in the empirical investigations. This study conducted a comparative analysis of open-source and dot NET-based semantic web platforms for ontologies development. Two popular dot NET-based semantic web platforms, namely, SemWeb.NET and dotNetRDF were analyzed and compared against open-source environments including Jena Application Programming Interface (API), Protégé and RDF4J also known as Sesame Software Development Kit (SDK). Various metrics such as storage mode, query support, consistency checking, interoperability with other tools, and many more were used to compare two categories of platforms. Five ontologies of different sizes are used in the experiments.

The experimental results showed that the open-source platforms provide more facilities for creating, storing and processing ontologies compared to the dot NET-based tools. Furthermore, the experiments revealed that Protégé and RDF4J open-source and dotNetRDF platforms provide both graphical user interface (GUI) and command line interface for ontologies processing, whereas, Jena open-source and SemWeb.NET are command line platforms. Moreover, the results showed that the open-source platforms are capable of processing multiple ontologies' files formats including Resource Description Framework (RDF) and Ontology Web Language (OWL) formats, whereas, the dot NET-based tools only process RDF ontologies. Finally, the experiment results indicate that the dot NET-based platforms have limited memory size as they failed to load and query large ontologies compared to open-source environments.

# Table of Contents

DECLARATION .....	i
ACKNOWLEDGEMENTS .....	ii
ABSTRACT .....	iii
Table of Figures .....	vii
Table of Tables .....	viii
<b>CHAPTER 1. INTRODUCTION</b> .....	1
1.1 Background .....	1
1.2 Rationale and Motivation .....	1
1.3 Problem Statement .....	2
1.4 Research Aim and Objectives .....	2
1.5 Methodology .....	3
1.5.1 Data Collection .....	3
1.5.2 Research Methods .....	3
1.5.3 Implementation .....	3
1.6 Dissertation Outline .....	4
1.7 Original Contributions .....	4
1.8 Publications .....	5
<b>CHAPTER 2. LITERATURE REVIEW</b> .....	6
2.1 Introduction .....	6
2.2 Semantic Web .....	6
2.3 Ontology .....	8
2.4 Languages for Representing Ontologies .....	10
2.4.1 Resource Description Framework (RDF) .....	10
2.4.2 Resource Description Framework Schema (RDFS) .....	11
2.4.3 Web Ontology Language (OWL) .....	12
2.5 dot NET-Based Semantic Web Libraries .....	12
2.5.1 SemWeb.NET .....	13
2.5.2 LinqToRdf .....	13
2.5.3 dotNetRDF .....	13
2.5.4 RDFSharp .....	13
2.5.5 OwlDotNetApi .....	14

2.5.6 dotSesame .....	14
2.5.7 BrightstarDB .....	14
2.5.8 TODE.....	15
2.6 Open Source Platforms for Ontologies Development .....	15
2.6.1 Protégé .....	15
2.6.2 Jena API.....	16
2.6.3 RDF4J .....	16
2.6.4 Ontostudio.....	16
2.6.5 Swoop .....	17
2.6.6 Neon Toolkit .....	17
2.6.7 Apollo .....	17
2.7 Related Work .....	17
2.9 Conclusion .....	20
<b>CHAPTER 3. RESEARCH METHODOLOGY .....</b>	<b>22</b>
3.1 Introduction.....	22
3.2 Data Collection .....	24
3.3 Types of Research Methodologies.....	22
3.4 Design Research process.....	24
3.4.1 Awareness .....	26
3.4.2 Suggestion.....	27
3.4.3 Development .....	27
3.4.4 Evaluation .....	27
3.4.5 Conclusion .....	28
3.5 Application of Design Research on this study .....	28
3.5.1 Awareness .....	29
3.5.2 Suggestion.....	29
3.5.3 Development stage.....	29
3.4.4. Evaluation .....	33
3.4.6 Conclusion .....	34
<b>CHAPTER 4. COMPARATIVE FRAMEWORK OF DOT NET-BASED AND OPEN SOURCE TOOLS FOR ONTOLOGY DEVELOPMENT.....</b>	<b>35</b>
4.1 Introduction.....	35
4.2 Framework Overview .....	35

4.2.1 Development layer .....	36
4.2.2 Storage Media Layer .....	37
4.2.3 Retrieval Layer.....	38
4.2.4 Comparison Layer.....	38
4.3 Related Work .....	40
4.4 Conclusion .....	42
<b>CHAPTER 5. EXPERIMENTS .....</b>	<b>43</b>
5.1 Introduction.....	43
5.2 Experiment Requirements.....	43
5.2.1 Dataset.....	43
5.3 Experimental Results .....	48
5.3.1 Comparison of dotNetRDF and SemWeb.NET Libraries .....	48
5.4 Comparison Analysis of Dot NET and Open Source Environments.....	53
5.4.1 Analysis Results on Loading, Executing and Querying Ontologies.....	53
5.5 Conclusion .....	59
<b>CHAPTER 6. CONCLUSION AND FUTURE WORK .....</b>	<b>61</b>
6.1 Summary of the Study .....	61
6.2 Limitations and Recommendations for Future Work .....	61
6.3 Conclusion .....	62
<b>BIBLIOGRAPHY .....</b>	<b>Error! Bookmark not defined.</b>
<b>APPENDIX A: FULL CODE OF ONTOLOGY DEVELOPMENT IN DOT NET ENVIRONMENT.....</b>	<b>71</b>
<b>APPENDIX B: FULL CODE OF ONTOLOGY DEVELOPMENT IN OPEN SOURCE ENVIRONMENT.....</b>	<b>79</b>
<b>APPENDIX C: FULL TABLE ON COMPARISON OF DOT NET AND OPEN SOURCE PLATFORMS FOR ONTOLOGY DEVELOPMENT .....</b>	<b>86</b>

# List of Figures

<b>FIGURE 2.1: THE SEMANTIC WEB ARCHITECTURE (TIM BERNERS LEE, 2001).</b>	7
<b>FIGURE 2.2:EXAMPLE OF RDFS ARCHITECTURE</b>	11
<b>FIGURE 3.1: DESIGN RESEARCH PROCESS (VAISHNAVI &amp; KUECHLER, 2004)</b>	26
<b>FIGURE 3.2: THE CORE DOTNETRDF LIBRARY</b>	30
<b>FIGURE 3.3: DOTNETRDF LIBRARY OVERVIEW IN VISUAL STUDIO 2010</b>	31
<b>FIGURE 3.4: SEMWEB LIBRARY OVERVIEW IN VISUAL STUDIO 2010</b>	31
<b>FIGURE 3.5: PROTÉGÉ LIBRARY OVERVIEW</b>	32
<b>FIGURE 3.6: JENA LIBRARY OVERVIEW IN ECLIPSE.</b>	32
<b>FIGURE 3.7: RDF4J SET UP OVERVIEW IN IN ECLIPSE</b>	33
<b>FIGURE 4.1: FRAMEWORK OF COMPARISON OF DOT NET-BASED AND OPEN SOURCE SEMANTIC WEB PLATFORMS.</b>	36
<b>FIGURE 5.1: VIEW OF ONTODPM ONTOLOGY IN PROTÉGÉ</b>	45
<b>FIGURE 5.2: VIEW OF WIKIMOVIE ONTOLOGY IN PROTÉGÉ</b>	45
<b>FIGURE 5.3: VIEW OF TERO ONTOLOGY IN PROTÉGÉ</b>	46
<b>FIGURE 5.4: VIEW OF GENE ONTOLOGY IN PROTÉGÉ</b>	46
<b>FIGURE 5.5: VIEW OF AGRICULTURE &amp; FORENSIC ONTOLOGY (AFO) IN PROTÉGÉ</b>	47
<b>FIGURE 5.6: VIEW OF DRUG ONTOLOGY IN PROTÉGÉ</b>	47
<b>FIGURE 5.7: LOADING TIMES OF ONTOLOGIES IN DOTNETRDF, SEMWEB, PROTÉGÉ, JENA, AND RDF4J</b>	54
<b>FIGURE 5.8: QUERIES RESPONSE TIMES OF ONTOLOGIES IN DOTNETRDF, SEMWEB, PROTÉGÉ, JENA AND RDF4J</b>	56
<b>FIGURE 5.9: QUERIES EXECUTION TIMES IN DOTNETRDF, SEMWEB, PROTÉGÉ, JENA AND RDF4J</b>	57



# List of Tables

<b>TABLE 5.1:</b> CHARACTERISTICS OF ONTOLOGIES IN DATASET.....	44
<b>TABLE 5.2:</b> SAMPLE CODES USED TO IMPLEMENT RDF TRIPLES IN DOTNetRDF .....	49
<b>TABLE 5.3:</b> SAMPLE CODES USED TO IMPORT RDF TRIPLES IN DOTNetRDF.....	49
<b>TABLE 5.4:</b> SAMPLE CODES USED TO CONSTRUCT SUBJECT, PREDICATE AND OBJECT IN SEMWEB.NET .....	50
<b>TABLE 5.5:</b> SAMPLE CODES USED TO IMPORT DIFFERENT RDF FORMATS IN SEMWEB.NET .....	50
<b>TABLE 5.6:</b> COMPARISON RESULTS OF SEMWEB.NET AND DOTNetRDF .....	52
<b>TABLE 5.7:</b> LOADING TIME IN DOTNetRDF, SEMWEB, PROTÉGÉ, JENA AND RDF4J.....	54
<b>TABLE 5.8:</b> MEANS OF QUERY RESPONSE TIME IN DOTNetRDF, SEMWEB, PROTEGE, JENA AND RDF4J .....	55
<b>TABLE 5.9:</b> QUERY EXECUTION TIME IN DOTNetRDF, SEMWEB, PROTÉGÉ, JENA AND RDF4J...	57
<b>TABLE 5.10:</b> SUMMARY OF COMPARISON OF DOT NET AND OPEN SOURCE SEMANTIC WEB PLATFORMS .....	58

# List of Acronyms

AI :	Artificial Intelligence
AJAX :	Asynchronous JavaScript and XML
API :	Application Programming Interface
ASP .NET :	Active Server Page (Microsoft script engine)
C# :	C Sharp
CERN :	European Organization for Nuclear Research
DAML :	DARPA Agent Markup Language
DL :	Description Logic
DOI :	Digital Object Identifier
GUI :	Graphical User Interface
HTML:	HyperText Markup Language
ICT:	Information and Communication Technology
icABCD:	International Conference on Advances in Big Data, Computing and Data Communication Systems
IDE:	Integrated Development Environment
LINQ:	Language Integrated Query (Microsoft)
MVC:	Model View Controller
MS SQL:	Microsoft Structured Query Language
NS:	Name Spaces
N3:	Notation 3
OIL:	Ontology Interchange Language (XML)
OKBC:	Open Knowledge Base Connectivity
OntoDPM:	Ontology of Development Project Monitoring

OWL:	Web Ontology Language
RDBMS:	Relational Database Management System
RDF:	Resource Description Framework
RDFS:	Resource Description Framework Schema
RDQL:	RDF Data Query Language
SPARQL:	SPARQL Protocol and RDF Query Language
SQL:	Structured Query Language
SW:	Semantic Web
TODE:	Tool for Ontology Development and Editing
URI:	Uniform Resource Indicator
URL:	Uniform Resource Locator
URN:	Universal Resource Name
W3C:	World Wide Web Consortium
WWW:	World Wide Web
XML:	Extensible Markup Language

# CHAPTER 1. INTRODUCTION

## 1.1 Background

The amount of data created each day on the web has led to the creation of a new technology that enables integrated information to be understood and processed by machines. This technology is called semantic web and it is an extension of the current World Wide Web (Fluit et al., 2003). Semantic web depends strongly on the development of ontologies which play a significant role in the engineering and transport of machine-readable information. The main advantage of the semantic web is the presentation of information to humans and machines in the same way (Taye, 2010). Ontology is based on information-sharing concepts that allow users and software programs to reuse the content of information from other ontologies (d'Aquin et al., 2002; Ochs et al., 2017). Also, it enables users to extract and easily choose information based on the domain of interest (Sun et al., 2011). Hence, Ontology-based systems have been adopted in various domains including e-commerce, e-learning and e-government (Lu et al., 2007).

To develop ontologies, several platforms including open source and dot NET environments have been created. Some commonly used platforms include Protégé, Jena and RDF4J known as Sesame for open source environment (Broekstra et al., 2002; Buranarachi et al., 2016) and dotNetRDF, SemWeb.NET and RDFSharp for dot NET users (Mazilu et al., 2009). These platforms enable Semantic web developers to build, use and maintain ontologies (Lambrix et al., 2003). In general, dot NET environments are rarely used in Ontology creation and management (Islam et al., 2010). The purpose of this study was to empirically analyze and compare the use of open source and dot NET environments for Ontology development.

## 1.2 Rationale and Motivation

Ontology development for the Semantic web has made a substantial contribution in handling and processing data of the current World Wide Web (Horrocks, 2008; Ruta et al., 2017). Semantic web enables the sharing and reuse of information among people, organizations and computer-based software (Santos et al., 2016). The increase of web users has led to the development of customized platforms with enhanced querying capacity and therefore allowing the extraction of information according to the users' needs (Konstantinidis et al., 2017). Hence, semantic web has been adopted

in data management for public databases such as government, business and health organizations (Zenuni et al., 2015; Amato et al., 2016). These institutions generate a massive amount of data which need to be integrated and automatically processed by computers and displayed in simple formats for consumers (Aoki-Kinoshita et al., 2017; Kaur et al., 2017). Currently, there is a constant development of new tools for Ontology development as well as the improvement of existing ones to expand the application of semantic web (Rio, et al., 2017; Temourika et al., 2017). Due to the increasing popularity of semantic web, there is a need to evaluate different environments and their respective platforms for efficient application in Ontology development.

### **1.3 Problem Statement**

Ontology-based systems are currently used in different fields such as medicine, education and finance (Vegetti et al., 2016). Ontologies play a significant role in accessing and processing huge data generated by different web activities and provide quick responses to users' (Adrian et al., 2014; Slater et al., 2016). Although there are several tools for Ontology development, there is a lack of guidelines that allow developers to choose a suitable tool for any given application (Noy et al., 2002). Previous studies conducted on ontologies have focused on the application of semantic web using open source platforms (Slimani, 2015) but little has been done under licensed environments such as dot NET. Furthermore, there is no empirical research on the comparative use of open source and dot NET environments in constructing ontologies. Therefore, there is a need to compare the effectiveness of Ontology development in both open source and dot NET environments.

### **1.4 Research Aim and Objectives**

The aim of this research is to compare the use of open source and dot NET environments for developing ontologies. The objectives are:

1. To investigate existing open source and dot NET platforms for Ontology development.
2. To investigate ontologies on the Semantic Web.
3. To implement selected ontologies in both open source and dot NET environments
4. To compare and discuss the strengths and weaknesses of open source and dot NET environments for Ontology development.

## **1.5 Methodology**

### **1.5.1 Data Collection**

A literature search provided the data for this study. Journal articles, conference papers and books that focus on the use of open source and dot NET environments for Ontology development were used as the primary sources of information.

### **1.5.2 Research Methods**

This study applied a design research method to meet the research objectives. Design research consists of five processes (steps) including awareness of the problem, suggestions, development, evaluation and conclusion (Kuechler et al., 2011). In this study, the awareness stage identifies the gap between dot NET and open source environments in terms of creating, storing and querying ontologies and many more. Therefore, this stage clarifies the need of comparison framework of dot net and open source environments in developing ontologies. In the second stage of design research i.e. the suggestion stage, we proposed a framework that evaluates and compares two dot net-based tools, namely, SemWeb.NET and dotNetRDF, and three open source platforms including Protégé, Jena API and RDF4J. In the development stage, all five platforms were used to create an existing Ontology, namely, OntoDPM (Fonou-Dombeu et al., 2010) as well as processing other ontologies imported from the internet. The evaluation stage uses a set of metrics to evaluate the proposed framework. These metrics include Ontology loading time, query execution time, query response time, tool's architecture, query support, storage mode, import/export capabilities, built in or external reasoners and many more (García-Castro et al., 2005; Mazilu et al., 2009). Finally, the conclusion stage presents the results of the performance evaluation of the framework and also discusses its limitations.

### **1.5.3 Implementation**

This research developed ontologies in the development stage of the framework. The implementation was done in two environments including dot NET framework and open source. Java IDE (Integrated Development Environment) such as Eclipse was used as an open source while Visual Studio was used as an IDE that support dot NET framework. Application Programming Interfaces (API) including Jena and Protégé, RDF4J as well as SemWeb.NET and dotNetRDF dot NET libraries are configured in open source and dot NET platforms, respectively. The

abovementioned platforms were used to create ontologies and store them in file system in various formats such as N3, Turtle, RDF/XML etc. Also, these platforms were used to load ontologies of different formats and sizes in their internal memories. The stored RDF(S) and OWL Ontology files are queried using SPARQL a Semantic web query language.

## 1.6 Dissertation Outline

The rest of chapters are structured as follows:

**Chapter 2** presents the literature review that has been conducted on the use of semantic web technology, ontologies and Ontology languages in different domains. Also, this chapter discusses various studies on comparison and evaluation of Ontology development tools. **Chapter 3** outlines the materials and methods applied in this study by investigating existing methodologies in the information technology domain and explains in detail the design research method adopted for this study.

**Chapter 4** presents the proposed framework of comparing open source and dot NET platforms for Ontology development.

**Chapter 5** implements the proposed framework and analyses the results of the study. **Chapter 6** concludes the study, provides recommendations and outlines the limitations and future research.

## 1.7 Original Contributions

The main contributions of this research are as follow:

- A comparison framework of dot NET and open source platforms for Ontology development is presented in Chapter 4. The proposed framework aims to serve as a guideline to Semantic Web developers who wish to use either open sources or dot NET environments.
- In Chapter 5 Subsection 5.4 implemented the proposed framework and analyzed the results of the performance in processing ontologies in both dot NET and open sources environments
- In Chapter 5, an empirical evaluation of dot NET tools in terms of storing and querying RDF/OWL ontologies in presented. This work was published in Mahoro & Fonou-Dombeu (2019).

- Chapter 5 reports a comparative study of two dot NET Semantic Web libraries, namely, dotNetRDF and SemWeb.NET. This work was published in Mahoro & Fonou-Dombeu (2020).
- A comparative analysis of dot Net-Based and Open Source Platforms for Ontologies Development is described in Chapter 5. This work was published in Mahoro & Fonou-Dombeu (2020).

## 1.8 Publications

The following publications were extracted from this study:

- L.J. Mahoro, and J.V. Fonou-Dombeu, (2019) An Empirical Evaluation of dot NET-Based Tools for OWL/RDF Ontologies Processing, *In Proceedings of the 2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, 5-6 August 2019, Winterton, South Africa, ISBN: 978-1-5386-9236-3, pp. 280-284.
- L. J. Mahoro and J. V. Fonou-Dombeu, "A Comparative Analysis of dot NET-Based and Open Source Platforms for Ontologies Development," *In Proceedings of 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, Durban, KwaZulu-Natal South Africa, 6-7 August 2020, pp.1-7, DOI: 10.1109/icABCD49160.2020.9183887.
- L. J. Mahoro and J. V. Fonou-Dombeu, "A Comparative Study of dotNetRDF And Semweb.NET Semantic Web Libraries," *In Proceedings of 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, Durban, KwaZulu-Natal, South Africa, 6-7 August 2020, pp.1-6, DOI: 10.1109/icABCD49160.2020.9183808.



# CHAPTER 2. LITERATURE REVIEW

## 2.1 Introduction

The use of semantic web improves reusability and integrity of data on the web. For this reason, many platforms from different environments including open sources and licensed ones such as dot NET have emerged to facilitate programmers to build, maintain and reuse ontologies (Konstantinidis et al., 2017). This chapter presents the introduction and the structure of semantic web, the use of ontologies and languages recommended by W3C to represent ontologies in development of Semantic web applications. Furthermore, in this chapter we provide existing open source and dot NET platforms for developing ontologies. Also, works related to this study are presented and discussed in this chapter.

## 2.2 Semantic Web

The term “Semantic Web” was coined by Tim Berners-Lee the inventor of the first generation of web referred to World Wide Web (WWW) also known as Web1.0 created at CERN Laboratories in 1989 (Hiremath & Kenchakkanavar, 2016). This web was created for nothing other than displaying information from the web creator to the end users.

As technology improved, the second generation of World Wide Web also referred as web2.0 has emerged as an extension of web1.0. This Web enables people to share information among them in many ways such as social media networks (Facebook, twitter, Instagram, LinkedIn....) or other web activities where users can create their own contents (Darwish, 2011; Choudhury, 2014).

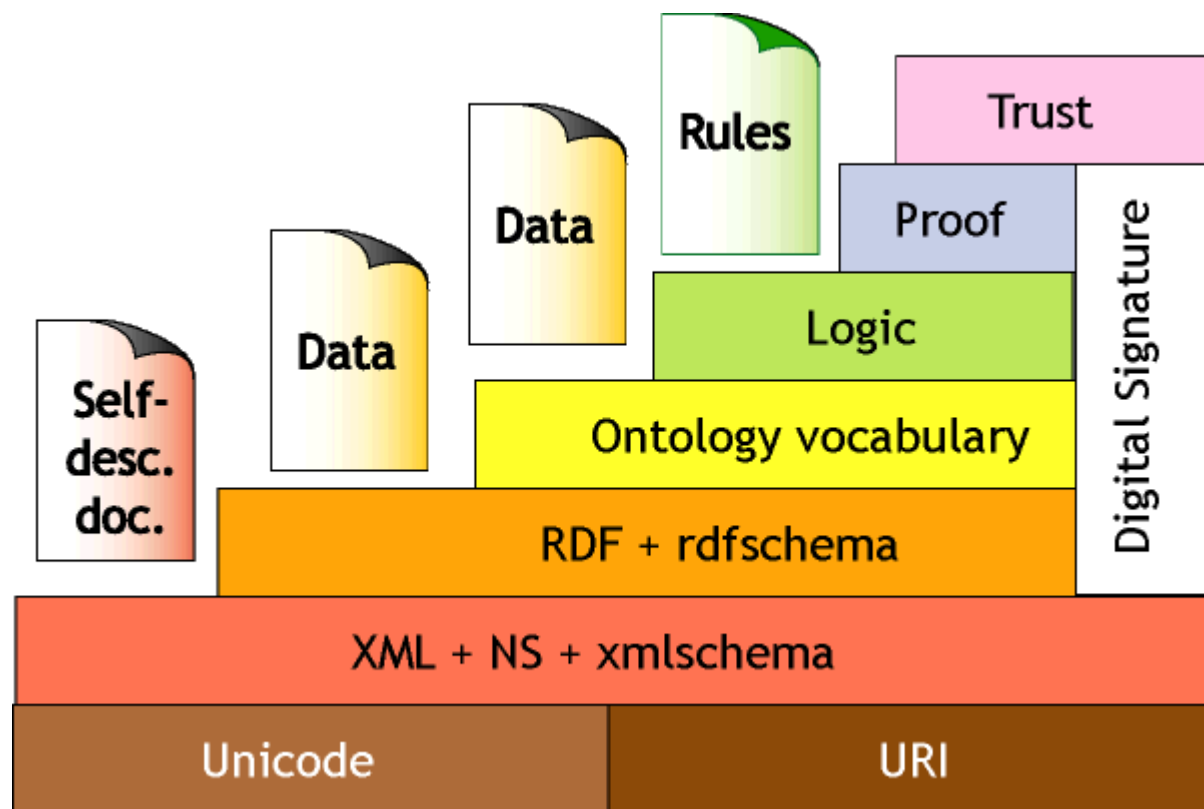
The huge amount of data on this web include text documents, games and multimedia files are written in HTML and other markup languages that display information in the way that can be manipulated only by humans via web browsers such as Firefox, Chrome etc. (Taye, 2010).

The current web is significantly important in our daily lives by providing the useful links of document pages for easier communication between people (Berners-Lee, Handler & Lassila, 2002). However, this web does not answer the question of knowledge sharing and data integration on the web without human intervention. Hence, Tim Berners-Lee had a vision to move from web of information to the web of knowledge so-called semantic web, with the purpose of not replace but by extending the current World Wide Web (Zhang, 2007).

In Semantic web technology, data are represented based on their meaning rather than its content documents and their links, to be understood by both humans and computers (Berners-Lee et al., 2001).

Tim Berners-Lee and his colleague (2001) defined Semantic web as an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. This succeeded by adding meta-data i.e. data to describe the meaning of web pages in machine-readable formats.

The implementation of semantic web application was based strongly on various technologies that automate data processing on the web. Refer to the Figure 2.1 shows these technologies and explains the building block of semantic web layers.



**Figure 2.1:** The Semantic Web Architecture (Tim Berners Lee, 2001).

Unicode and URI layers, Universal Resource Identifiers (URIs) including Universal Resource Locator (URL), Universal Resource Name (URN) are used in semantic web technology to give

names and locate resources to be identified over the web. Unicode is a standard level of machine language that gives a unique computer number to every character of all languages of the world to be identified (Berners-Lee et al., 2001).

The extensible Markup Language (XML) refers to data representation model which is in machine-readable format that enable describing exchanging data on the web; it also facilitates the creation of interoperability of metadata. NS are name spaces which can be identified via URIs to enable semantic interoperability among metadata (Berners-Lee et al., 2001).

Resource Description Framework (RDF) is the beginning of semantic web technology. RDF provides a framework to represent and describe data with semantics in the way that machine can access those data; RDF Schema defined as an extension of RDF where data are defined with rich semantics by adding more vocabularies so that agents can logically infer metadata to perform their tasks (Berners-Lee et al., 2001).

Ontology vocabulary layer is a knowledge representation that defines concepts and relation between them. In this layer, simple descriptions and complex classifications are created to enable a software agent to interpret data intelligently (Berners-Lee et al., 2001).

The logic layer rules are expressed with logic based on First Order Logic where agents can draw logical conclusion from semantic encoded data (Berners-Lee et al., 2001). The proof layer validates the evidence from inference logic activities, and finally the last semantic web layer Trust will execute the rules generated in the logic layer.

## **2.3 Ontology**

The term “Ontology” borrowed from the realm of philosophy is the study of nature of being (Gruber, 1993). Ontology is the transportation channel of meaning data to be used by semantic web applications (Fernandez-Lopez & Gomez-Perez, 2003; Uthayan & Anandha Mala, 2015). It allows knowledge to be represented, shared and reused across different people, organizations and application systems (Slimani, 2015; Zenuni et al., 2015).

In computer sciences, many artificial intelligence researchers define ontology in different ways. Gruber (1993) defined Ontology as a formal, explicit specification of a shared conceptualization for a given domain.

- Formal: means that Ontology should be in machine-readable format so that machines can process the provided semantic information.
- Explicit: means that concepts and constraints within Ontology must be clearly defined.
- Specification: Ontology domain must be determined.
- Conceptualization: concepts with the same semantics must be in the same class.
- Shared: reflects the notion that an Ontology captures consensual knowledge, that is, it is not private for some individuals, but a large community can make a common agreement.

Chandrasekaran (1999) described Ontology as a content theory about the sorts of objects, properties of objects and the relations between objects within a specific domain of knowledge which lead Ontology technology to be adopted in different domains to help people to share their knowledge. It also plays a great role in solving issues of data integration and provides common vocabularies for different applications. Hence, Ontology is considered as the backbone of semantic web (Splendianiet al., 2011).

The main Ontology components are Concepts, Individuals, Relationships, Attributes and Axioms. These components provide knowledge models that can be used by software agents to browse the web contents on the humans' behalf (Heidari, 2009). The use of ontologies has become successful in several activities such as searching and retrieving information from files and databases built for semantic web applications (Uthayan et al., 2015). This has found applications in different fields such as health, education and e-government as well as e-commerce (Khozoie, 2012; Almeida, Santos & Monteiro, 2013).

For example, the clinic website with contents published in the form of Ontology where a software agent retrieves information from the doctor's page and schedules the appointments for patients according to the Doctor's working days and his availability (Berners-Lee et al., 2001).

In e-government, ontologies are applied to help users to find appropriate information and help them to solve problems related to interoperability and heterogeneity of data. In addition, concepts, rules and regulations used within government domains are presented with semantic annotation with a meaningful representation of civic events such as marriage, birth and death records (Klischewski & Jeenicke, 2004).

## **2.4 Languages for Representing Ontologies**

To achieve the semantic web vision various researcher's suggested different languages, standards, Application Programming Interfaces, platforms and other W3C recommendations for easy development of ontologies. In this section we provide a brief overview of common and popular languages used in Ontology development processes.

Ontologies can be expressed using knowledge representation languages that provide semantics in Ontology concepts. These languages include but are not limited to Resource Description Framework (RDF), Resource Description Framework Schema (RDFS), DARPA Agent Markup Language (DAML), Ontology Interchange Language (OIL), and Ontology Web Language (OWL) (Maniraj & Sivakumar, 2010)

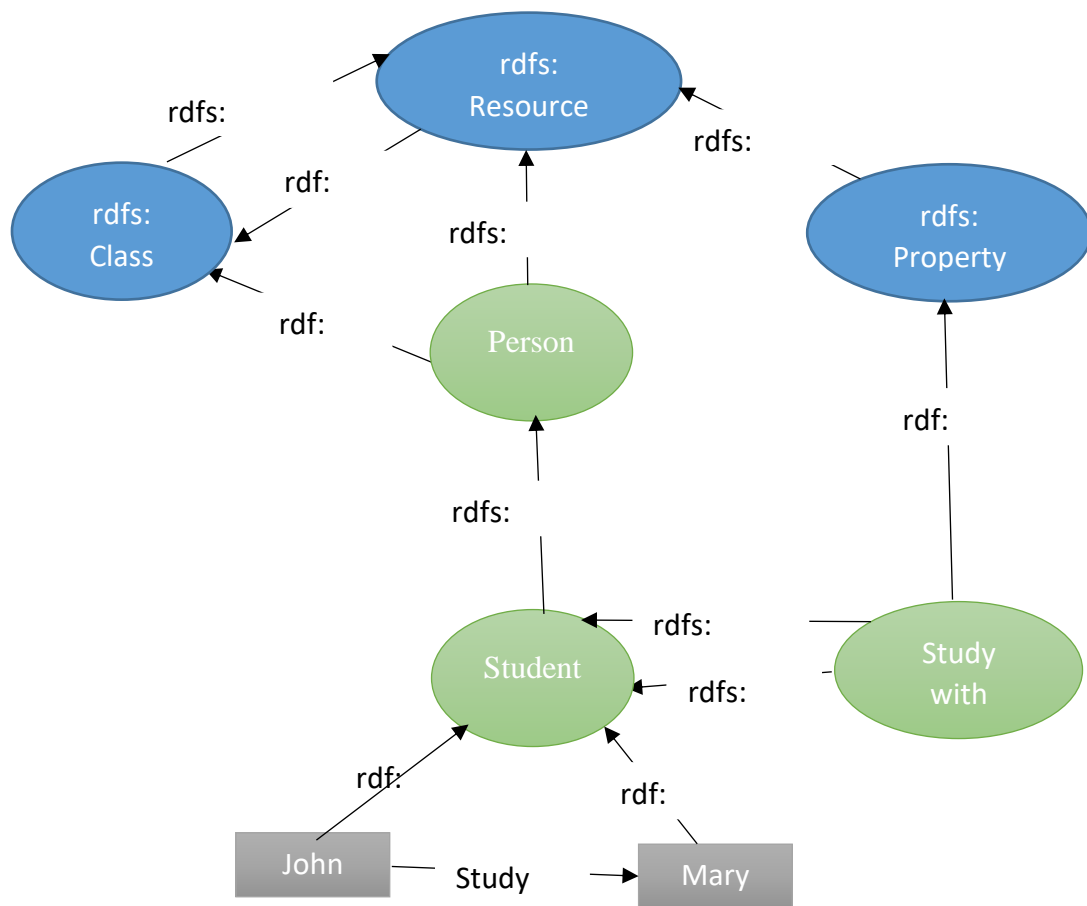
### **2.4.1 Resource Description Framework (RDF)**

RDF is the first layer of Semantic web technology recommended by W3C as the building block of the semantic web. It is defined as a graph- oriented data model designed to represent and exchange semantic data of any resources using meta-data i.e. data that describe other data; a resource can be anything such as web page identified by Uniform Resource Identifiers (URIs) on the web (Sanjay et al, 2010). RDF is used to process metadata which enables interoperability between application systems and transform information in machine-readable form on the web; furthermore, it is used to improve searching and navigation on the semantic web search engines (Taye, 2010).

RDF defines resources and their properties using simple statements, subject, predicate and object also known as triples, <S, P, O >. The Subject defines the thing (the resource) of what a statement is all about, the predicate identifies information or property that express about the subject and the object defines the value of the predicate (Kalyanpur, 2006). Subject, Predicate are Universal Resource Identifiers (URIs) and Object should be a URI or a literal value (Chakraborty et al., 2013). RDF statements can also be represented as a graph with two nodes and one arc where the arc represents resources property that link two nodes and nodes represent the resources Subject and Object, respectively (Fagbola et al., 2012).

### 2.4.2 Resource Description Framework Schema (RDFS)

RDF(S) Resource Description Framework Schema is an Ontology language that enables users to add basic vocabulary while describing RDF resources of interest domains and define relationship between classes and properties.



**Figure 2.1:** Example of RDFS Architecture

The top blue ellipses represent RDFS primitive modeling classes built in RDFS which are `rdfs:Class`, `rdfs:Resource` and `rdfs:Property`, the rest is simple student Ontology. This Ontology has two classes; `Student` class is an `rdfs:SubPropertyOf` class `Person`. `Study with` is the `Property` whose `rdfs:domain` and `rdfs:range` are both class `Student`. `John` and `Mary` objects are instance of class `Student`. Also, `rdf:type` link the objects and Ontology classes and between Ontology classes with

RDFS classes, `rdfs: SubClassOf` is used between Ontology classes (Student and Person) and among RDFS classes (`rdfs: Class`, `rdfs: Resource` and `rdfs: Property`).

### **2.4.3 Web Ontology Language (OWL)**

In addition to RDF and RDFS languages, Web Ontology Language (OWL) emerged to add vocabulary for high description of properties and classes (Sabou, 2006). OWL is an Ontology language based on description logic used for describing classes, properties and individuals. OWL is based on RDF/RDFS and provides knowledge representation, vocabulary sharing, advanced search and knowledge management (Kyeungshun et al., 2006).

OWL is a combination result of DAML+OIL, it has been standardized by W3C as a tool that build infrastructures to realize the semantic web vision. It comprises three sub languages which are OWL DL, OWL Lite and OWL Full (Kyeungshun et al., 2006).

1. OWL-Lite. This is easier to implement and expand the functionality of RDFS by supporting a classification hierarchy and simple features like cardinality constraints which is restricted to 0, 1. It attempts to provide more functionality than RDFS, which is important in order to support web applications.
2. OWL-DL. Includes all OWL language construct with restrictions. It contains the whole OWL vocabulary that is interpreted under a number of simple constraints. Primary among these can be found by the type separation. Class identifiers cannot simultaneously be properties or individuals. Similarly, properties cannot be individuals.
3. OWL-Full. Supports all expressiveness and the syntactic freedom of RDF. This is composed of the complete vocabulary but interpreted more broadly than in OWL DL. A class can be treated simultaneously as a collection of individuals and can have properties of their own.

## **2.5 dot NET-Based Semantic Web Libraries**

Although dot NET environment is slow to react on developing tools for editing, storing and querying ontologies (Islam, Siddiqui & Shaikh, 2010), many efforts have been made to develop Semantic Web tools compatible with the Microsoft dot NET platform such as TODE, Knowledge.NET, SemWeb.NET, dotNetRDF, LinqToRdf, RDFSharp, OwlDotNetApi,

dotSesame, BrightstarDB. These platforms facilitate developers to easily conduct the entire development process of ontologies in Microsoft .NET environment.

### **2.5.1 SemWeb.NET**

Tauberer (2005) released SemWeb.NET a dot NET library for developing ontologies for RDF level. It provides features like reading and writing RDF data encoded with different notation such as RDF/ XML, N3, Turtle etc. SemWeb.NET provides two persistent RDF storage mode, namely, in-memory store for storing small amounts of data and SQL store for storing large amounts of data. However, this library does not provide any tool for reading and writing OWL ontologies (Tauberer 2010). The core classes in the SemWeb.NET library is found in the SemWeb namespace. In SemWeb namespace, four classes provide the fundamentals for all aspects of the library: Resource, Statement, StatementSource and StatementSink.

### **2.5.2 LinqToRdf**

SemWeb.NET has been extended by Mathews (2007) to make a new tool called LinqToRdf. The author presents LinqToRdf as a framework that represents and query the RDF data model in dot net environment. It uses the RdfDataContext which is the source of all entities from triple store. LinqToRdf provides Tools such as LinqToRdf Designer, a visual tool to edit ontologies, and LinqToRdf Metal a command line for querying semantic web applications. LinqToRdf comes with two parts namely, LinqToRdf MSI which is the installer that installs the core assemblies needed to conduct Semantic queries and LinqToRdf Designer which integrates with visual studio.

### **2.5.3 dotNetRDF**

Robe (2009) developed a library written in C# compatible with dot NET framework called dotNetRDF. It provides an easy but strong API for reading and writing RDF data. dotNetRDF is an extensible library which means that any user can contribute by adding features. Also, dotNetRDF provides a built in rdfEditor tool for editing RDF data and rdfConveter tool which provides a Graphic User Interface (GUI) to manipulate the contents of triples stores. It supports triple store platforms such as Allegro graph, 4stores, Sesame and Virtuoso.

### **2.5.4 RDFSharp**

Another dot net platform called RDF Sharp was developed in C# designed to create applications based on RDF model. RDF Sharp is composed by three layers, namely, RDF Sharp Model, RDF



Sharp Store and RDF Sharp Query. The first layer RDF Model provides core classes which create and manage RDF model such as resources, literals, triples, graphs and namespaces. The second layer i.e. RDF Sharp Store While RDF Sharp query enables modeling, storing and querying RDF data of applications developed in dot NET environment (Mdesalvo, 2010).

### **2.5.5 OwlDotNetApi**

The OwlDotNetApi is an OWL (Ontology Language) API written in C# for the .NET environment based on RDF model. It is fully compliant with the W3C OWL syntax specifications and can be used within any dot NET language. The API uses the underlying data model from Drive to build a directed linked graph from the OWL Ontology. The RDF parser which is based on the dot NET platform has been modified to parse OWL ontologies instead (Owldotnet, 2009).

### **2.5.6 dotSesame**

The dotSesame project is a cross platform library developed in C# a dot NET language. Initially dotSesame was derived from the Sesame project, which is originally written in Java. It is an open source RDF database with supports for RDF Schema inferencing and querying Ontology files (Openrdf, 2010).

### **2.5.7 BrightstarDB**

BrightstarDB is a dot net library which provides three levels of API that store processes and maintain ontologies (Brightstardb, 2006). It provides several layers of API that are involved in the development of ontologies; the three main API's level in BrightstarDB are described as:

1. Entity Framework API is a powerful and simple technology to construct dot NET domains model by simply creating a set of interfaces compatible with dot NET framework to define the data model.
2. Data Object API which provide a generic object layer on top of RDF data. This layer provides abstract classes that allow the developer to manipulate collections of triples into data objects.

3. RDF Client API provides a simple set of methods for creating and deleting execution queries. It enables Visual Studio integration that takes Interface definitions and generated a strongly typed dot NET domain model that stores its data as RDF in a BrightstarDB instance.

### **2.5.8 TODE**

Islam et al. (2010) presented a Tool for Ontology Development and Editing based on dot NET framework called TODE. Its interface is easy to use and provides a good environment to create and edit ontologies and supports an Ontology development methodology called methOntology.

The architecture of TODE is composed of three main parts, namely, Model, View and Controller. These parts facilitate any modification from users without changing its business logic. TODE has been implemented in C# a dot NET programming language and MS SQL Server have been used to create database for storing ontologies. ASP.NET and AJAX also used to create its interface (Islam et al, 2010). However, this tool is still in the development phase.

## **2.6 Open Source Platforms for Ontologies Development**

Several Ontology development platforms have been developed in the past (Rastogi et al., 2017). Most of these platforms are open source which gives the right to Semantic web users to customize such tools based on the requirements to accomplish their tasks. This section discusses, in detail, some of these tools including Protégé, Jena API, RDF4J, Ontostudio, Swoop, Neon Toolkit and Apollo.

### **2.6.1 Protégé**

Protégé is an open source and free Ontology development tool, developed at Stanford Medical Informatics (Alatrish, 2013). Its architecture is separated by two parts, namely, model and view. The Model part is used for ontologies and knowledge representation. Based on the structure of the metamodel the constituent ontologies can be represented as classes, properties (slots), property characteristics and instances. On the other hand, the view part displays interface and manipulate the underlying model. Protégé can edit or build ontologies of different formats such as XML, RDF, RDF (S) and OWL (Fonou –Dombeu & Magda Huisman, 2011). As protégé provides a Graphic User Interface, using this view part Designers can create classes, instances (individuals) and give properties to those classes and restriction on the properties of facet. Also, it provides API for

querying and manipulating models and has a library withhold tabs to edit, visualize and manage ontologies (Jambhulkar and Karale, 2016).

### **2.6.2 Jena API**

Jena is another popular open source framework which has been developed in HP labs. It provides an Application Programming Interface (API) which enables the creation and manipulation of RDF graphs. Jena provides RDF API which enables reading and writing RDF data in the form of XML, Turtle, N3 etc. The query languages supported by Jena API are SPARQL and RDQL. Jena uses RDQL and SPARQL as query languages to query out RDF data in Jena persistence store or in-memory storage for storing temporary or permanently RDF data (Wang, Ding, Xiang & Xun, 2005). The Jena API supports three storage modes, namely, in-memory RDF storage, native storage and database storage of RDF graphs. There are several databases currently supported by Jena such as MySQL, Oracle, PostgreSQL, and many others (McBride, 2010). Also, Jena can be used as application by using the Jena Fuseki or as an API configured in Eclipse Integrated Development Environment (IDE) (Stegmaier et al., 2009)

### **2.6.3 RDF4J**

RDF4J, formerly known as Sesame SDK, is an open source platform for persistence storage of RDF data and provides mechanisms of querying those data. It provides Storage and Inference Layer (SAIL) which is an application programming interface (API) that gives special methods to RDF and use these methods to store RDF data in DBMS (Broekstra, Kampman and Harmelen, 2002). The SAIL API has three major parts including RQL which supports two query languages namely, SPARQL and SeRQL. RQL query engine is used to query the repository through the SAIL API. The second part is RDF admin module which allows data from the file to be loaded into the RDF4J repository and the final part the RDF export module allows data to be exported from the repository into a file (Ye, Ouyang & Dong, 2010). RDF4J can visually process RDF data using a Graphical User Interface (GUI) or using a Command Line Interface (CLI).

### **2.6.4 Ontostudio**

Ontostudio is a Semantic Web tool developed by OntoPrise which provides full support of all W3C standards like RDF(S), OWL, and RIF etc. It also offers extra features such as collaborative

Ontology development using collaborative server, drag and drop conversion between different languages, and easy integration of database information to the Ontology (Kapoor & Sharma, 2010).

### **2.6.5 Swoop**

Swoop is also an open source tool for Ontology editing/constructing. Swoop tool allows the comparison between entities and the relationship of different ontologies. Swoop, enables the import of ontologies from text formats, XML, OWL and RDF. Swoop supports reasoned like RDFS, PELLET etc. (Kapoor & Sharma, 2010).

### **2.6.6 Neon Toolkit**

Neon is an open source and multi-platform environment tool for Ontology editing construction. It is mainly based on the eclipse platform. It can be extended by using plugins (Erdmann & Waterfeld, 2012). These plugins have the capacity to cover the whole life cycle of the Ontology engineering. The main features are visualization, XML editing, import to F-Logic, export to F-Logic etc.

### **2.6.7 Apollo**

Apollo is a user-friendly knowledge modeling application. It allows a user to model Ontology with basic primitives, such as classes, instances, functions, relations and so on. The internal model is a frame system based on the OKBC protocol. The knowledge base of Apollo consists of a hierarchical organization of ontologies. Ontologies can be inherited from other ontologies and can be used as if they were their own ontologies (Kapoor & Sharma, 2010). Each Ontology is the default Ontology, which includes all primitive classes. Each class can create a number of instances, and an instance inherits all slots. The summary of dot NET and open sources platforms discussed above is presented in Appendix C.

## **2.7 Related Work**

Choosing the right tool is the first step to begin Ontology development. Developers are required to choose appropriate metrics to evaluate the performance of various tools. For example, Iacob (2009) compared the performance of SemWeb and dotNetRDF APIs for dot Net environment with different criteria including IDE integration, triple storage, SPARQL integration support, performance level of documentation and licensing.

Over the past years, many ontologies have been developed by researchers for different purposes. In the e-government domain Fonou-Dombeu (2010) presented a conceptual Ontology for e-government monitoring of development projects in Sub Saharan Africa (OntoDPM). Protégé, an open source knowledge base editor tool was used to create OntoDPM Ontology.

Another study on OntoDPM (Fonou-Dombeu & Huisman, 2011) was done in the e-government field by combining Ontology building methodology using a framework adopted from the Uschold and King and two semantic web platforms namely Protégé and Jena. Furthermore, the OntoDPM written in UML formalism and implemented in web Ontology language (OWL) with protégé using its semi-formal representation. Jena Ontology was employed to create the resource description framework (RDF) in formal representation of OntoDPM.

Kiong et al. (2009) presented a Health Ontology Generator (HOG) using a health database such as Microsoft Access and SQL Server. The development of the Ontology generator involves building methods for creating and reading the Ontology. The Health Ontology Generator performs both these tasks. In generating the Ontology, database tables are treated as classes, fields as functional properties, and records as instances. The Ontology generated can be read using third-party software such as Microsoft Word, Excel and Internet Explorer. HOG is implemented on the Windows platform using C#.NET.

A popular and mature Ontology in biology domain, Gene Ontology, has been created in GO project developed using OBO-Edit tool. This project combines three subprojects, namely, FlyBase, Mouse Genome informatics and *Saccharomyces* Genome database. The GO Ontology covers three main biology domains including molecular function, cellular components and biological process. The GO Ontology contains the vocabulary used in the biology field and relationship between those terms (Taha, 2013).

Also, in the biology domain, Raffat et al. (2012) proposed a human biological viruses (HBVO) Ontology for classification of viruses that belongs to the human community. Human biological viruses' Ontology intends to support an integrated conceptual framework of open biology Ontology with a structure and controlled vocabulary to describe and categorized biological viruses. This Ontology was created using Open Biological Ontology (OBO) principles in OBO- Edit tool.

Yeong et al. (2013) proposed a system based on Ontology called Learner-centered Smart E-learning System (OLSES). The system creates profiles for learners after registration to OLSES and store the track of study program of learners. The Ontology learner centered smart E-learning has been developed using protégé editor. Also, many studies have been carried out on the comparison and performance evaluation of Ontology development tools. Therefore, this section discusses completed studies on the comparison and evaluation of Ontology development tools as reported in the literature.

Alatrish (2013) conducted a survey on five Ontology editors including Protégé, Apollo, Ontostudio, Swoop and free edition of TopBraid Composer. The survey classifies evaluated tools according to the general description of the tool, how it interoperates with other tools, the architecture and usability of the tool.

The exploration and analysis on Ontology development tools was presented in Singh & Anand (2013). The authors categorized Ontology construction tools in two categories, namely, Ontology development tools and tools for mapping, alignment and merging tools. The evaluation and comparison were conducted using eight Ontology development tools which were selected based on various criteria such as general issues, software architecture, interoperability, inference services and usability.

One of the recent studies on the survey and comparison of Ontology development tools including Ontostudio 3.1, Protégé 5.0, Swoop, TODE, OWLGrEd as well as Odese have been published in Rastogi et al. (2017). This survey focused on a small number of features provided by tools such as architecture, interoperability, storage, library and GUI design. However, the authors did not clarify the availability of these tools, i.e., whether being open source or commercial. A similar survey was conducted by Kapoor and Sharma (2010). Their survey consists of four open source tools (Protégé 3.4, IsaViz, SWOOP and Apollo). They also described the tool by presenting the developer, and their features and functionalities. These tools were categorized based on their architecture, interoperability, inference services, usability, and overview of their versioning and collaborative work support. Another study by Duineveld et al. (2000) presented a comparison framework on different development tools including Ontolingua, Webonto, Protégé win, Ontosaurus, ODE and KADS22. The authors concluded the paper by arguing that Webonto, Protégé win and ODE were the best suited for conceptualization and formalization phase in Ontology development.

Rahamatullah et al. (2010), conducted an online survey in which users were asked some questions on the usability of the tools. They evaluated Ontology tools using Questionnaire methodology. The questions were formulated in four categories including tools, task, environment, and user friendliness.

Islam and Abbasi (2010), also provided a comparison of different Ontology development tools based on some criteria like availability, architecture, imports/exports and supporting tools etc. In Denny (2002), Ontology development tools were compared based on different features like modeling limitations, base language, web support and use, import and export format, graph user view, consistency checking, multi-user support, merging, lexical support, and information extraction.

The authors in Norta et al. (2010) did a comparison of different Ontology development tools based on functional and non-functional requirements for developing a good Ontology editor. In the functional requirements they focused on features like collaboration, multilingual and natural language support and verification, whereas in the non-functional requirements, they addressed features such as modifiability, inerrability and portability.

A similar study in Rastogi et al. (2017), a survey was conducted based on a number of features of tools including architecture, interoperability, storage, library and graphic user interface design.

In García & García-Peñalvo (2011), the authors evaluated a visual modelling tool for OWL ontologies. The evaluation was mainly focused on user-centered approach and questionnaires. Noy & Musen (2002) did another comparison study, where they evaluated Ontology mapping tools. The evaluation criteria included the interoperability of a tool with other tools, the ability to import and export ontologies, the scalability, extensibility and usability of the tool. However, none of above-mentioned studies evaluated and compared Semantic Web libraries for building and processing ontologies in the dot NET environment.

## **2.9 Conclusion**

In this chapter, the overview of Semantic Web Technology, the use of Ontologies and languages used to represent ontologies were discussed. We also presented existing comparison metrics in object-oriented programming reported in literature, Ontologies developed within open source such as Protégé, Jena API, RDF4J, Ontostudio, Swoop, Neon Toolkit and Apollo and dot net platforms

including TODE, Knowledge.NET, SemWeb.NET, dotNetRDF, LinqToRdf, RDFSharp, OwlDotNetApi, dotSesame, BrightstarDB were presented.



# CHAPTER 3. RESEARCH METHODOLOGY

## 3.1 Introduction

This chapter aims to outline and explain in detail the research methods used in this study. The presentation of the chapter starts with data collection techniques used to gather relevant information and the domain Ontologies used for this study. Also, this chapter presents existing research methodologies in scientific research as well as in Information Technology domain such as experimental research, creative research, design research, descriptive research, explanatory research etc. The description of each research methodology is provided and the methodology used for this study namely, design research is presented. Furthermore, all stages of design science research including Awareness, Suggestion, Development, Evaluation and Conclusion are explained in more details. Finally, the application of design research in this study is provided and discussed.

## 3.2 Types of Research Methodologies

The characteristic of good research strongly depends on appropriate and efficient methodology that meets the research objectives (James et al., 2012). Using the correct methodology increases the accuracy of results, and makes these results more credible (Oates, 2005). Several research methodologies exist and are identified by Goddard and Melville (2004), and Oates (2005):

- **Experimental research:** here, the value of an independent variable is varied whilst another variable known as the dependent variable is analysed every time there is a change in the independent variable. Other variables that are neither independent nor dependent may still be used in the experiment.
- **Creative research:** this type of research deals with the discovery of new models, theorems, algorithms etc. It is less structured and may not always be planned. Trials and errors are the main methods used in this type of research.
- **Descriptive research:** this type of research is also called the “case-study” research, and involves the studying of a precise situation to ascertain whether it is a blueprint of any existing general theories.

- **Ex post facto research:** this type of research works by analysing the effects and tries to deduce the relevant and related causes. It is different from experimental research since the directional flow of research is from effects to causes.
- **Action research:** this is a stepwise research which entails the following steps; identifying a problem, gathering comprehensive data, agreements made between the researcher and the stakeholders; implementation of the remedial action, and after a period of time, an evaluation is performed to see if the problem has been resolved.
- **Historical research:** this method involves studying past events to identify effects-causes patterns. Current situations are examined using past effect-cause patterns in order to predict future events.
- **Expository Research:** this research methodology is based entirely on existing information. Researchers widely read, compare, contrast, analyse and synthesize all points of view on a specific subject; new important insights can be developed as a result of that deep analysis and comparison (Goddard et al. 2004).
- **Qualitative Research:** is a type of scientific research which consists of an investigation process that seeks answers to a question, systematically uses a predefined set of procedures to answer the question, collects evidence, produces findings that were not determined in advance and produces findings that are applicable beyond the immediate boundaries of the study (Marshall, 2003).
- **Quantitative Research:** is a study involving the use and analyses of numerical data using statistical techniques. They pose questions of who, what, when, where, how much, how many, and how is an inquiry into an identified problem, based on testing a theory, measured with numbers, and analyzed using statistical techniques. The goal of quantitative methods is to determine whether the predictive generalizations of a theory hold true hypothesis (Apuke, 2017)
- **Design Science Research:** it consists of following a set of predefined steps in order to solve a problem or create new knowledge. The steps consist of seven activities, namely: awareness, suggestion, development, evaluation, and conclusion.

### **3.3 Data Collection**

Data collection for this study was conducted out through the use of a qualitative and quantitative research approaches. Qualitative in the sense that all comparison criteria used were learned from the literature and quantitative in that the experiments were carried out to analyse and compare two environments including dot NET-based and open source semantic web platforms for developing ontologies. The dataset used consist of six ontologies, five of them were downloaded from the internet while OntoDPM Ontology which is available in description logic was developed in Protégé platform.

The next section describes the methodology used in this research which is design science research.

### **3.4 Design Science Research (DSR) Process**

According to Friedman (2009), design is a set out procedure that is followed when solving a problem, improving on something that already is in existence or when creating something new altogether. The concept of design is applicable in several domains including engineering and architecture (Hevner & Chatterjee, 2009). Oates (2005), defines research as a practice that is done in order to create new knowledge or make some contribution to the field of knowledge. Oates (2005) goes on to describe six core elements of research known as (6P's) that must be considered in any research area of interest. The elements are: purpose, product, process, paradigm, participants and presentation.

- Purpose specifies the main reason of undertaking the research study.
- The product represents the outcome of the research; it represents the contribution to the body of knowledge.
- The process represents the sequence of tasks or activities undertaken for the research.
- Participants include people directly or indirectly involved in the research. They could be people that you interviewed, people editing the research report and more. Involving people must be done ethically and legally.
- Paradigm represents a pattern or shared model of thinking. The paradigms which can be used by researcher are positivism, interpretivism, and critical research (Friedman, 2003). The positivism paradigm defines the reality as anything that can be perceived by human

senses. With interpretivism, the researchers interpret elements of the study or the society; the reality exists only in their mind. It is experienced through social interaction with several actors. In critical research, the reality is created by people who tend to manipulate others and drive them to perceive things according to their belief.

- Presentation is the mean by which the research is disseminated to the public or stakeholder. It may be presented as a written paper, thesis and computer-based product.

Design science research can be summarized as a process followed to develop or create new knowledge. Knowledge consists of artefacts that solves a problem or improves a situation. Simon (1996), defined artefacts as things that do not occur naturally but instead, are produced by humans. Hevner et al. (2010) provides some guidelines to follow when performing design science research, which are:

- The output of the research must be an artefact used in the field of computer science. It does not have to be a completed product or artefact; enough knowledge can be gained in the analysis and the design; thus, the artefact does need to be completed for knowledge to be gained.
- The goal of the research must solve an identified and relevant problem.
- The resulting product or artefact must be evaluated using appropriate methods.
- The research process and creation of the artefact must add to the body of knowledge.
- A proper research process must be followed.
- A proper search and investigation into appropriate solutions must be done, the environment in which the artefact is applied must be considered during the search.
- A report of the research outcome must be provided to all involved parties.

The guidelines mentioned above aims at increasing the value of design science research (Hevner et al., 2010). The design science research process consists of five stages namely: Awareness, Suggestion, Development, Evaluation and Conclusion (Vaishnavi & Kuechler, 2004). Figure 3.1 shows the five stages involved in design science research. The output of one process is used as the input in the next process as shown



**Figure 3.1:** Design Science Research Process (Vaishnavi & Kuechler, 2004)

### 3.4.1 Awareness

The Awareness stage identifies and recognizes the existence of a problem then categorically states the problem. Some of the methods used in identifying the existence of a problem are:

- Problems are identified in literature and future work by authors.
- Users have experienced a problem which need to be solved.
- New findings are made and,
- Technological developments are made.

This stage produces outputs to be used in the second stage, the suggestion stage. Outputs are statements describing problems that need to be addressed. Nevertheless, no solutions need to be found. But those statements will be of great help in proposing possible solutions and benefits. The goal of the research will be identified by those statements. The statements will be of great help during the development and the evaluation stages. In the development phase, they will be used as guidelines to follow while in the evaluation phase they will be used as criteria to evaluate the output.

### **3.4.2 Suggestion**

This stage is also known as the proposition stage and here, all the available solutions to the statements identified in the awareness stage are formulated. The solutions are obtained by checking on the currently available knowledge base, and in related works. Some suggestions may be found in the literature done by other researchers and in a scenario where no solutions have been identified, new ideas as possible solutions are explored. The output of this stage is a proposal of the possible solutions that can be applicable.

### **3.4.3 Development**

Out of all the possible solutions obtained in the suggestion stage, one solution that is superior to the rest of the solutions is implemented in the development stage. The implementation of the solution is however either partially or fully implemented. Partial implementation of a solution is considered when there are limitations such as time or resources. The output of the development stage, as identified by Oates et al. (2005) must be an artefact; thus, traditional software life cycle methodologies should be followed. Software life cycle typically consists of five phases namely: requirements, design, implementation, testing and maintenance (Van Vliet, 1993; Kumar & Bhatia, 2014).

The main purpose of the requirement phase is to get a complete description of the problem that needs to be solved; also, it identifies all the conditions that should be met by the solution. The design phase proposes a model which represents the solution to be implemented. The implementation phase implements the model using a specific technology or programming language; the solution is physically implemented. In the testing phase, the solution is tested. The goal of the testing is to ensure that problems identified in the requirements phase are addressed. The maintenance phase helps fix errors which have gone undetected in the testing phase and other phases.

### **3.4.4 Evaluation**

This stage evaluates the proposed solution. The evaluation stage finds out if the questions identified in the awareness stages have been answered. Hevner et al. (2010) have identified three reasons why the evaluation phase should take place. The first reason is that the solution is

applicable to the problem and that it adds value; the second reason is that the evaluation gives credibility to the research and, thirdly, it helps determine the practical value of the development or the experiment. Methods chosen to evaluate should be aligned with the research objectives (Oates 2005). Evaluating the proposed solution can result in suggestions to modify the solution. A conclusion must be drawn.

### **3.4.5 Conclusion**

This last stage presents the output of the research. This stage presents the research and its contribution to the body of knowledge. The contribution must be clearly listed in such a way that all questions identified in the awareness stage are identified. The result of the research must be published, and knowledge gained must be identified. The result could lead to further investigations.

The output of design research is to provide an artefact. Oates (2005) identified and listed several artefact types which could be identified in computer sciences fields and they are:

- Constructs: these represent conceptual vocabulary of a specific domain which are constructed in the initial stage of design.
- Models: these are representation of relationship between constructs
- Methods: a set of steps followed in order to perform some activities
- Instantiations: which is the practical implementation of construct, models and methods in a predefined environment.
- Theory developments: this is the development of new theories and improvement of existing theories in a knowledge domain.

### **3.5 Application of Design Science Research on this study**

As Marco (2010) mentioned, a design science research is an approach and a set of useful methods and guidelines used as a framework of conducting design research. The previous section presented all phases of design science research and the different processes involved in each phase. Therefore, the following section present the application of design research on this study.

### **3.5.1 Awareness**

The advent of the internet has made technology a significant component of the modern world lifestyle. Technology is used daily in different activities such as communication, banking, healthcare and education systems. Many tools such as Protégé, WebODE, OntoEdit, Sesame, Jena, Ontolingua, and so forth have been developed (Islam et al., 2010) to enable Ontology developers to create and edit, store, query, maintain and integrate ontologies. Most of these tools are open-source and can support several external plugins that perform the reasoning, visualization and management of ontologies. Furthermore, many studies have been conducted to evaluate such tools in the last few years. Although efforts have been made to develop Semantic Web tools for the Microsoft .NET platform such as TODE (Islam et al., 2010), Knowledge.NET (Safonov, 2006) SemWeb.NET (Tauberer, 2010) and dotNetRDF (Rob, 2009), for Ontology development. However, not enough work has focused on the analysis and comparison of dot NET-based tools against open source platforms to guide Semantic web developers on the choice of the appropriate tool to use.

### **3.5.2 Suggestion**

Ontology development activities have increased in numbers in recent years. Several tools have been developed to support the Ontology development process. Many studies have been conducted on the evaluation and comparison of such tools. This study conducts a comparative analysis of open-source and dot NET-based semantic web platforms for ontologies development. Two popular dot NET-based semantic web platforms, namely, SemWeb.NET and dotNetRDF are analyzed and compared against open-source environments including Jena API, Protégé and RDF4J also known as Sesame SDK.

### **3.5.3 Development stage**

The implementation of the proposed framework was done by using two dot net libraries namely, dotNetRDF and SemWeb.NET and three open source tools including Protégé, Jena and RDF4J. The design of the framework is fully covered in Chapter 4. The next section presents the development activities within the framework.



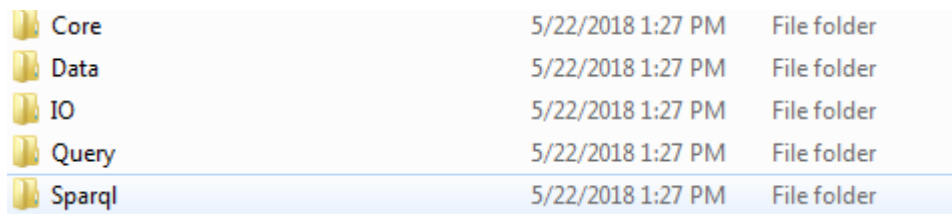
### a) Hardware and Software environment

The experiments were carried out on a computer with the following specifications: Windows 7 operating system, 4.00 GB RAM, and Intel (R) Core (TM) i3-2350M CPU @2.30GHz processor.

The software utilized in the experiments include Microsoft Visual Studio 2010 and Eclipse IDEs, two dot NET-based tools, namely, SemWeb.NET, dotNetRDF and three open source tools including Protégé, Jena API and RDF4J.

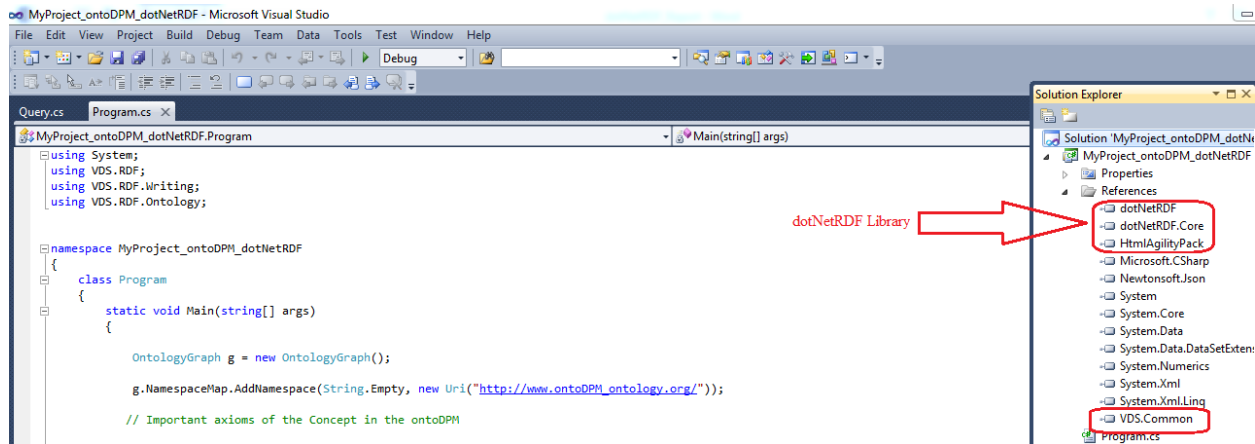
### b) Setup and Implementation

In the First stage, SemWeb.NET and dotNetRDF libraries were installed to run on Visual studio 2010 IDE and no specific configuration was required rather than importing these libraries in the IDE. The set-up overview is depicted in Figures 3.3 and 3.4. The dotNetRDF is simple but powerful semantic web library developed for dot NET developers who intend to create, process, query and write RDF data. This library provides a command line interface (CLI) and a graphical user interface which enables users to process and develop Ontology applications visually. The library is available to be downloaded freely from <https://github.com/dotnetrdf/dotnetrdf>, followed by unzipping the dotNetRDF folder which contain 5 subfolders as shown in Figure 3.2.



Core	5/22/2018 1:27 PM	File folder
Data	5/22/2018 1:27 PM	File folder
IO	5/22/2018 1:27 PM	File folder
Query	5/22/2018 1:27 PM	File folder
Sparql	5/22/2018 1:27 PM	File folder

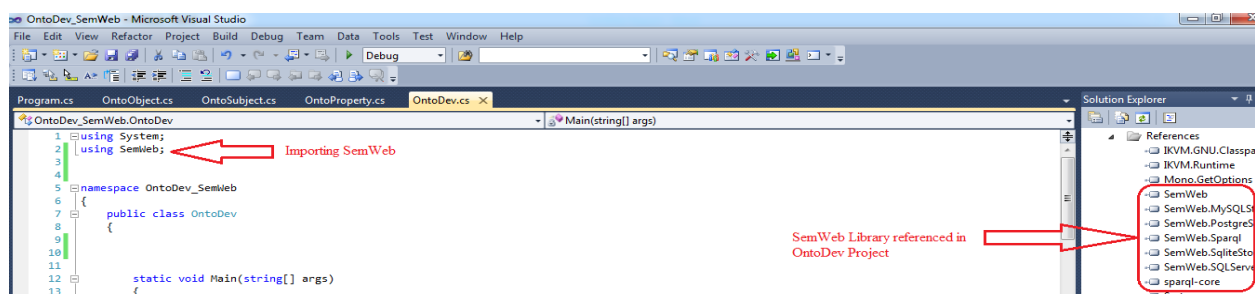
**Figure 3.2:** The Core dotNetRDF Library



**Figure 3.3:** dotNetRDF Library Overview in Visual Studio 2010

Before starting the Ontology development, after the addition of the dotNetRDF Library to the project, the dotNetRDF Namespaces must be imported globally to the class program as shown in Figure 3.3. The full code on Ontology processing in dotNetRDF library is provided in appendix A.

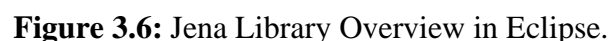
SemWeb.NET library is freely available from <https://github.com/JoshData/semweb-dotnet>. The library is configured within Microsoft.NET framework and employed to develop an Ontology in RDF/XML format. The SemWeb.NET library's namespaces including: SemWeb, SemWeb.MySQLStore, SemWeb.Sparql are added under references section in the OntoDev project within Microsoft dot NET using C# language as shown in Figure 3.4.



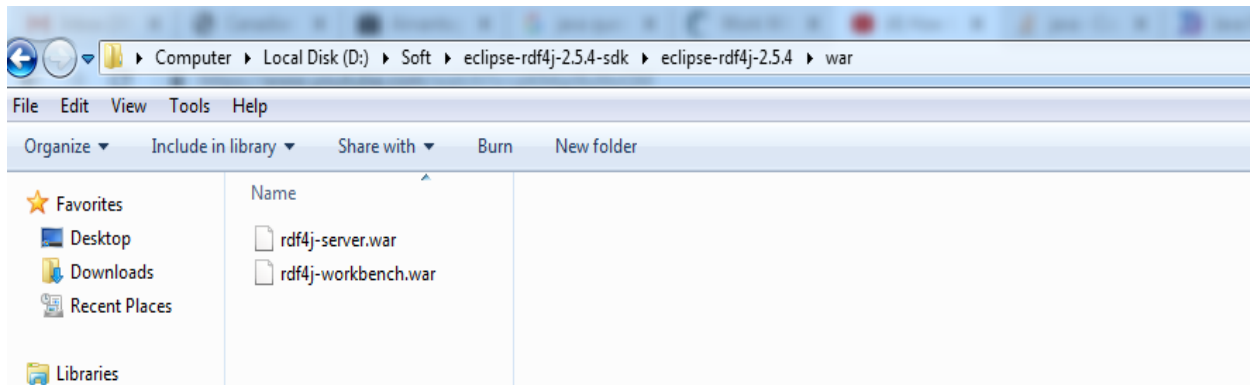
**Figure 3.4:** SemWeb Library Overview in Visual Studio 2010

After adding the SemWeb.NET library to build RDF triples of OntoDPM Ontology, three main classes were created including OntoSubject.cs, OntoProperty.cs and OntoObject.cs under OntoDev\_SemWeb solution in OntoDev project.

Like Dot Net Environmental Setup, Protégé, Jena, and RDF4J platforms were installed to run on Java environment. Note that Only Jena library was imported in Eclipse while the remaining platforms were installed as standalone applications. The set-up overview is depicted in Figures below.



Firstly, an integrated development environment called Eclipse is downloaded from <https://org/downloads/www.eclipse>. Eclipse IDE provides easy development of any application. Thereafter, Jar files of Jena API is downloaded and configured in Eclipse IDE. Full codes on implementation of Ontology using Jena API is provided in Appendix B.



**Figure 3.7:** RDF4J Set up Overview in in Eclipse

### 3.4.4. Evaluation

In the evaluation stage we checked the consistency and efficiency of developed and imported ontologies. Also, the evaluation of proposed framework is undertaken at this stage. The evaluation consists of three phases including: 1) Loading ontologies in Storage Media Layer (SML). In this phase Loading Time (LT) metric is used to identify the platform that took a lesser time to load ontologies in repositories. 2) Executing SPARQL Query against ontologies in dataset. This phase use Query Execution Time (QET) as a metric to measure which platform is faster to execute and process the query. 3) Query Response which use a metric called Query Response Time (QRT) to provide which platform responded to the query faster than others and for how long the query took to respond. More details on the evaluation of proposed framework is presented in chapter 5.

### 3.4.5 Research outcome and contribution

The main contributions of the research design are as follows:

- A comprehensive review and discussion of Semantic Web platforms for developing, storing and querying ontologies presented in (Chapter 2 Section 2.5 and 2.6). This work will serve as a guideline for Semantic Web developers who wish to use either dot net or open source environments.

- The proposed framework for comparison of dot net and open source environments for ontologies development (Chapter 4). The architecture of the framework is presented in Chapter 4 and implemented in Chapter 5.
- The evaluation and comparison of ontologies storage and query under two popular semantic libraries, namely, dotNetRDF and SemWeb.NET. This work was published in International Conference on Advances in Big Data, Computing and Data Communication Systems in Mahoro et al., (2019).
- The empirical analysis of the mechanisms used by Sesame and Jena to store ontologies in relational database presented in Chapter 5.

### **3.4.6 Conclusion**

This chapter introduces methodologies applied to research in computer sciences. Design research which is the method used in this study is described and all stages involved are presented. Design research is applied to this study and the results of each stage are discussed. The awareness stage identified the need to have an architecture to test several storage types and mechanisms. The suggestion stage proposed a framework for the comparison of dot net and open source environment for Ontology development. In the development stage, the tools needed to implement the framework are identified and presented. The evaluation stage presents the guidelines to test the framework. These guidelines include the usage of metrics such as loading of ontologies, the test of storage and query ontologies on dotNetRDF, Semweb.NET, Protégé, RDF4J known as Sesame and Jena. Finally, the conclusion stage identified the outcome of the performance of the framework.

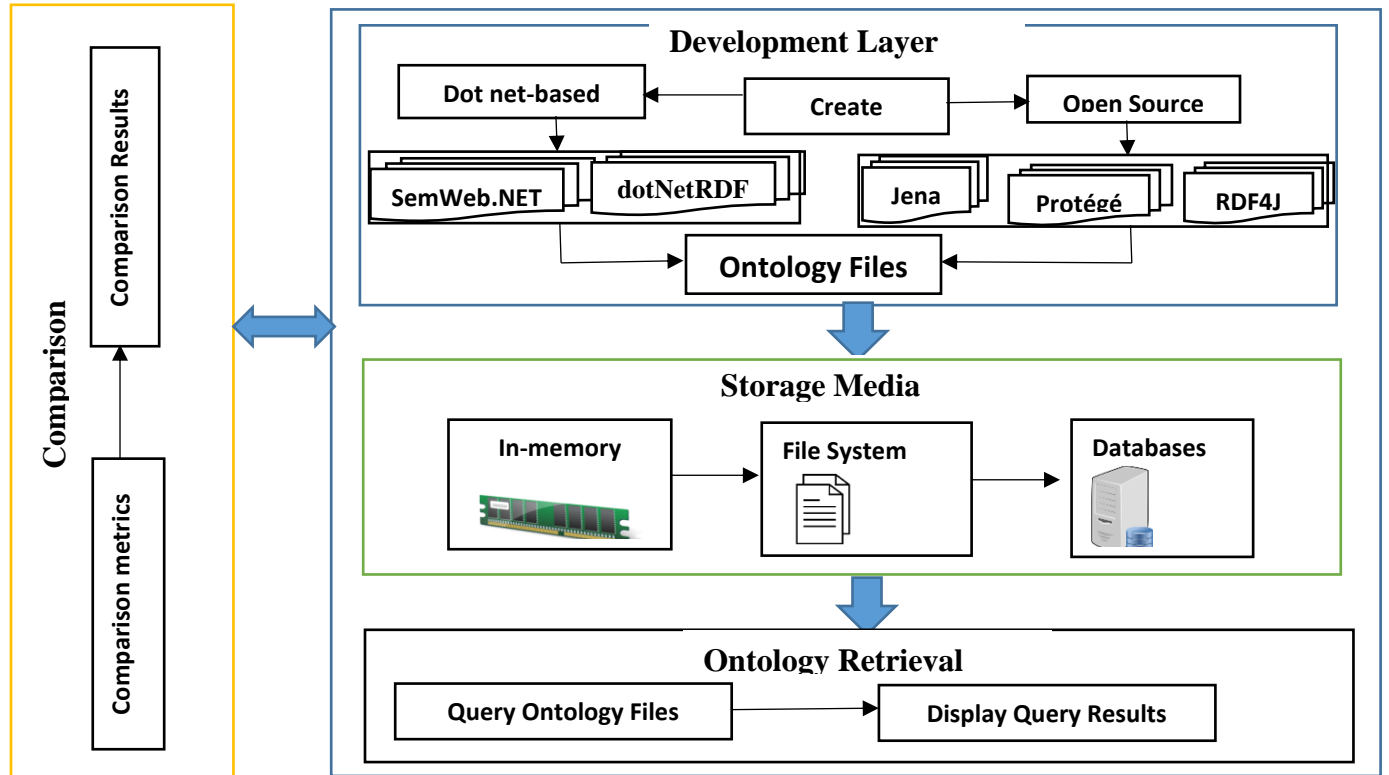
# **CHAPTER 4. COMPARATIVE FRAMEWORK OF DOT NET-BASED AND OPEN SOURCE TOOLS FOR ONTOLOGY DEVELOPMENT**

## **4.1 Introduction**

This chapter presents a proposed comparative framework of dot net based and open source environments for Ontology development. The framework compares the facilities in developing, storing and querying ontologies provided in both environments. The framework consists of four phases (layers) namely, development layer, storage media layer, Ontology retrieval layer and comparison layer. The first layer of the framework is the development layer. The purpose of this layer is to create new or import existing ontologies which will served as an input for the other layers in the framework. Furthermore, this layer is also used to check the consistency and efficiency of created or imported ontologies and load the useful ontologies into the next phase for further processing. The second layer is the storage media layer; the role of this layer is to create, edit, search and delete Ontology graphs or repositories where ontologies are stored. The third layer which is the Ontology retrieval layer is used as an interface for querying out the stored ontologies from graphs or repositories created in the memory of local or remote computers. Finally, the last layer of the framework is the comparison layer which uses several performance evaluation metrics such as the programming construct, Ontology storage and querying supports, documentation level, usability and scalability, loading time, query response time, disk space, and the storage efficiency and many more to test and compare the state of Ontology development in both dot net and open source environments.

## **4.2 Framework Overview**

The framework consists of four layers as shown in Figure 4.1. The layers are development layer, storage media layer, Ontology retrieval layer and comparison layer



**Figure 4.1:** Framework of Comparison of dot NET-based and Open Source Semantic Web Platforms.

#### 4.2.1 Development layer

This layer provides mechanisms for building a new Ontology from scratch or reusing existing ontologies. The popular Ontology development tools used in this phase are dotNetRDF, SemWeb.NET in dot NET environment, Jena API, Protégé and RDF4J open source platforms as presented in Figure 4.1. Firstly, an existing Ontology called OntoDPM developed in Fonou-Dombeu and Huisman (2011) is used as an input Ontology to test the framework. In fact, OntoDPM represents the domain of monitor's development projects monitoring in developing countries. OntoDPM is presented in Description Logic language in Fonou-Dombeu and Huisman (2011) and rewritten in machine readable format. This layer enables the user to create an Ontology file *de novo* or importing existing ontologies from different sources on internet.

- **dotNetRDF**

The dotNetRDF is a free Library written in C# language for dot NET users. The dotNetRDF is designed with a powerful API which provides methods to write and read the Resource Description Framework (RDF) data and support SPARQL a Semantic web query language (Rob, 2009).

- **SemWeb.NET**

SemWeb.NET is another Semantic Web library written in C# for Microsoft dot NET platform. This library also provides classes for reading, writing, manipulating and querying ontologies. However, SemWeb.NET only operates at the level of RDF (Tauberer, 2010).

- **Protégé**

Protégé is an open source Semantic Web editor developed at Stanford University. It provides different interfaces to design models and knowledge-based systems using ontologies. Protégé also provides a full support in creating and editing multiple ontologies (Rastogi et al., 2017). It can be extended with many plugs-ins which performs extra functionalities such as visualization and reasoning. An example of Visualization plug-in in Protégé is OntoViz, whereas, examples of reasoners plug-ins are hermit and pellet. Protégé allows the definition of classes, class hierarchy's, restrictions and the relationships between classes and properties (Ochs et al., 2017).

- **Jena**

Jena is another open source Semantic Web platform developed in the HP labs. It is a java Application Programming Interface (API) which enables the creation and manipulation of RDF graphs. Jena provides API which enables the reading and writing of RDF data in the form of XML, Turtle and N3. Jena uses RDQL as query language to query RDF data in Jena persistence store and in-memory storage (Carroll et al., 2004).

- **RDF4J**

RDF4J formerly known as Sesame Software Development Kit (SDK) is an open source framework written in java which enables Ontology creation, storage and management of repositories. Also, RDF4J provides an Inference Layer (SAIL) which is an application programming interface (API) that gives special methods to create RDF and use these methods to store RDF data in memory, native or DBMS databases (Singh et al., 2015).

#### **4.2.2 Storage Media Layer**

This layer allows the ontologies developed or imported in the development layer to be stored either in the in-memory, file system or database.



- **In-memory Storage:** This storage mode allows a small amount of OWL/RDF data to be stored in the main memory of the computer for a short time. The disadvantages of this kind of storage is that it takes long to process medium ontologies; furthermore, it failed to process big ontologies (Zhou & Xing, 2013).
- **File System Storage:** This storage method uses files to store OWL/RDF data permanently on the hard drive of the computer. Although this storage approach provides a high speed in loading and querying ontologies (Zhou & Xing, 2013), it has some limitations such as data redundancy, poor accessibility and limited data sharing.
- **Database Storage:** This approach is used to store ontologies permanently in relational databases such as MySQL, SQL Server and Oracle or in No-SQL databases such as Mongo DB, HBase and Cassandra (Zhou & Xing, 2013).

### 4.2.3 Retrieval Layer

This layer allows users to query and display useful information from the stored ontologies using Semantic Web query languages such as SPARQL, RDQL, SeRQL, etc. The code below illustrated the sample of SPARQL query that retrieve subject, property and object stored within an Ontology.

```
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
```

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
SELECT? Subject? Object
```

```
WHERE { ? subject rdfs: subclassOf ? object }
```

The first line of code provides the namespace of querying ontologies at RDF level.

The second line of code provides the namespace while querying ontologies at RDF Schema level and finally; the last part is the statement used to select RDF data in the form of subject, property and object.

### 4.2.4 Comparison Layer

This layer is organized as of a set of comparison criteria used to compare the performance of the dot NET and open source tools. These criteria are defined below:

- **Tool's Developers:** This provides information about the creators of the tool.
- **Last version:** Indicates the latest version of the tool available.
- **Availability:** States the way of accessing the tool, that is, open source or commercial.
- **Tool's Architecture:** this criterion indicates the type of the tool i.e., standalone, client/server or n-tier.
- **Interoperability with Other Tools:** this criterion indicates if the tool can offer any collaboration with other tools during the Ontology development process.
- **Query Support:** This tells whether the tool supports any of semantic web query languages such as SPARQL, RDQL, SeRQL, etc.
- **Ontology Storage Mode:** This specifies the back-end systems supported by the tool such as Relational databases, no Sql databases, etc.
- **Import/Export Format:** This provides information on which Ontology format can be imported or exported.
- **Build-in Inference Engine:** This shows the type of inference engine used by the tool.
- **Consistency Checking:** This tells whether the tool includes features that checks the validity of created Ontology.
- **Stability:** This criterion indicates if the tool is still active or is no longer in use.
- **Extendibility:** This refers to the possibility of customizing the tool by developing plug-ins or other class libraries to work with the tool.
- **Multiple Users Support:** This provides information on whether the tool can support more than one user at the same time.
- **Ontology Library:** This indicates if the tool has an ontologies repository for further reuse.
- **Graphical User Interface:** This determines if the tool has any support that help users to create or edit ontologies graphically.
- **Web Support:** This criterion refers to the availability of web-based component in the tool.

- **OWL Editor:** This specifies if the tool can process and develop ontologies in OWL format.
- **Reasoners:** This tells whether the tool includes software components that enable inferring new information from Ontology axioms.
- **Implemented in:** This indicates the programming language used to develop the tool.
- **Backup Management:** This shows the mechanism provided by the tool to support the backup of data.
- **Exception Handling:** This gives information about the tool's capabilities of catching logic and runtime errors and throws exceptions.
- **Operating System Support:** This indicates the operating systems that can support the tool such as, Linux, Macintosh or Windows.

### 4.3 Related Work

For Semantic web developers it is very important to have a good understanding on tools and languages to use in development process. Moreover, there have been several studies conducted on comparison and evaluation of Ontology editors based on various features provided by these tools.

Singh et al. (2013) provided an evaluation and comparison on Ontology development tools that were selected based on various criteria such as general issues, software architecture, interoperability, inference services and usability of the tool. Rahamatullah et al. (2010) conducted an online survey in which users were asked some questions on usability of the tools and evaluated Ontology tools using the Questionnaire methodology. The questions were formulated in four categories including tools, task, environment, and user friendliness. Kapoor & Sharma (2010) conducted a comparative study of Ontology editing tools based on four types of metrics including tool's architecture, interoperability, inferencing services and usability from users. Islam & Abbis (2010) also provides a comparison of different Ontology development tools based on some criteria like availability, architecture, imports/exports and supporting tools etc. Denny's (2002) Ontology development tools were compared based on different features like modeling limitations, base language, web support and use, import and export format, graph user view, consistency checking, multi-user support, merging, lexical support, and information extraction. The authors in Norta et al. (2010) did a comparison of different Ontology development tools based on functional and non-

functional requirements for a good Ontology editor. In the functional requirements they focused on features like collaboration, multilingual and natural language support and verification whereas, in the non-functional requirements they addressed features such as modifiability, inerrability and portability. Five Ontology editors Apollo, Onto Edit, Protégé, Swoop and TopBraid Composer were analysed and compared Alatrish (2010). Evaluation comprised qualitative evaluation of tools based on many features such as architecture, interoperability, knowledge representation, inference support and usability of tools. Garcia et al. (2011) evaluated a visual modelling tool for OWL ontologies. The evaluation was mainly focused on a user-centered approach and questionnaires. (Noy et al., 2002) did another comparative study, where they evaluated Ontology mapping tools. The evaluation criteria included the interoperability of a tool with other tools, the ability to import and export ontologies, the scalability, extensibility and usability of the tool. However, none of above-mentioned studies has evaluated and compared Semantic Web libraries for building and processing ontologies in the dot NET environment. Slimani (2015) conducted a comparative study on Ontology development tools, languages and formalisms. The main criteria in this comparison was the users' interest and their ability to solve real world problems. A similar study is presented in Kapoor et al. (2010). The authors performed a comparison of Ontology editing tools including Protégé 3.4, IsaViz, SWOOP and Apollo. This comparison was done based on four metrics including tool's architecture, interoperability, inferencing services and usability. Also, in Zhdanova et al. (2005) provided useful information to assist developers to choose the appropriate Ontology language while developing semantic web application.

A study by Mizoguchi (2003) on the evaluation of different Ontology development tools based on criteria like development process support, collaboration with other tools, tool's architecture, interoperability, Ontology model, instance definition and inference support. Islam et al. (2010) also provides a comparison of different Ontology development tools based on their availability, architecture, imports/exports and supporting tools. Rastogi et al. (2017) an analysis and comparison of Ontology editing tools including TODE, OWLGrEd, Odase and Ontostudio was undertaken. This comparison was done based on features like architecture, storage, interoperability and design of graphical user interface.

The authors in Duineveld et al. (2000) presented a comparison framework for the evaluation of Ontology editors based on three dimensions. The general dimension investigated the features of

the tools that are also found in other programs. The Ontology dimension analyzed the features that are specific for Ontology development. Finally, the third dimension dealt with the features required to support collaborative Ontology development, interoperability, storage, library and GUI design.

Alatrish (2013), developed a comparative framework on Ontology development tools. Five Ontology editors, namely, Apollo, Ontostudio, Protégé, Swoop and TopBraid Composer were analysed and compared. The evaluation was based on user friendliness and the applicability of the tool in different applications.

#### **4.4 Conclusion**

In this chapter, the proposed comparative framework of dot net and open source environments in developing, storing and querying ontologies was presented. The architecture of the framework which consist of four phases (layers) including development layer, storage media layer, Ontology retrieval layer and comparison layer were discussed. Five Ontology tools used to evaluate the framework such as Semweb.NET, dotNetRDF, Protégé, Jena and RDF4J were provided and discussed. All metrics used to compare the performance of tools from both dot net and open source environments were discussed in this chapter. Furthermore, the related studies on comparison of Ontology development tools were discussed. Unlike these studies which only compare Ontology development tools in open source environment; this study brings in some difference by investigating tools from dot net environment.

# CHAPTER 5. EXPERIMENTS

## 5.1 Introduction

This chapter focuses on implementation of the experiments conducted in this study. Firstly, the characteristics of the dataset are provided. Secondly, the experimental results on a comparative study of two dot net based semantic web libraries namely, SemWeb.NET and dotNetRDF are presented and discussed. Finally, the facilities in terms of creating, storing and processing ontologies in both open source and dot net environments are identified and discussed.

The experiments in this study consists of six ontologies with different sizes which were parsed and processed in dot net and open source environments using platforms such as dotNetRDF and SemWeb.NET for dot net users; Protégé, Jena and RDF4J known as sesame in open source environment. Thereafter, various metrics such as Loading Time, Query Execution Time, Query Response Time and Storage Capacity were empirically measured and used to determine the performance of each tool.

## 5.2 Experiment Requirements

This section presents the characteristics of the dataset, hardware and software environments used to perform the experiments in this study.

### 5.2.1 Dataset

The dataset used in this study constitutes of six ontologies from different domains which were selected based on their format i.e. RDF, RDFS, OWL and their sizes i.e. small, medium, large ontologies. These ontologies are OntoDPM developed for e-government domain; WikiMovie which was developed for cinema industry; Gene Ontology used in biotechnology domain; Agriculture and Forestry Ontology (AFO) vocabularies and terms for agriculture and forest domains; Tero, an Ontology of health and welfare for medicine and pharmacy domains and Drug Ontology (Dron) an Ontology for drugs in pharmacy field. The OntoDPM was created using Protégé 5.2.0 an open source tool that create and edit ontologies while the other five were searched using semantic web search engines such as Swoogle and downloaded from internet in different repositories. The OntoDPM Ontology consists of 214 statements, 30 classes, 19 properties and 18 individuals; The WikiMovie Ontology consists of 504 statements, 35 classes, 2 properties and 104 individuals; The Tero Ontology consists of 412742 statements, 7 classes, 4 data properties, 4 object

properties and 27625 individuals. The Gene Ontology consists of 40675 classes, 5 object properties and 17 individuals; The Agriculture and Forestry Ontology consists of 372054 statements, 12 classes, 3 object properties, 26 data properties and 31776 individuals while Drug Ontology consists of 344385 statements, 85552 classes, 1 data property, 19 object properties and 19 individuals. The OntoDPM Ontology is a knowledge-based model that monitors and evaluates the development government's projects management and NGOs in developing countries and Sub Sahara Africa (Fonou-Dombeu & Huisman, 2011). In fact, the axioms of concepts, class hierarchy and class instances use description logic to represent the Semi-formal of OntoDPM model. Description Logic is a formal language for knowledge representation that has a syntax that uses basic mathematical logic symbols to represent the relationships that exist between the vocabularies and concepts that constituents a domain (Horrocks, 2007).

**Table 5.1:** Characteristics of Ontologies in Dataset

<b>Ontology Name</b>	<b>No. of C</b>	<b>No. of S</b>	<b>No. of P</b>	<b>No. of I</b>	<b>Size in byte</b>	<b>Format</b>
OntoDPM	30	87	19	18	26000	RDF
WikiMovie	35	505	14	104	60000	RDF
Tero	7	412743	4	27625	168942000	RDF
Gene	49761	1405520	9	-	157692000	OWL
AFO	10	372078	3	31776	30800000	RDF
Dron	434,663	547360	20	19	473000000	RDF/XML

The characteristics of the ontologies used in the experiments in this study such as the number of classes, number of statements properties and individuals as well as their sizes in bytes and formats are presented in Table 5.1 where the letters C, S, P and I in the headings of Table 5.1, represent the words classes, statements, properties and instances, respectively. The figures below indicate the view of each Ontology used in the dataset in Protégé 5.2.0 an Ontology editing tool.

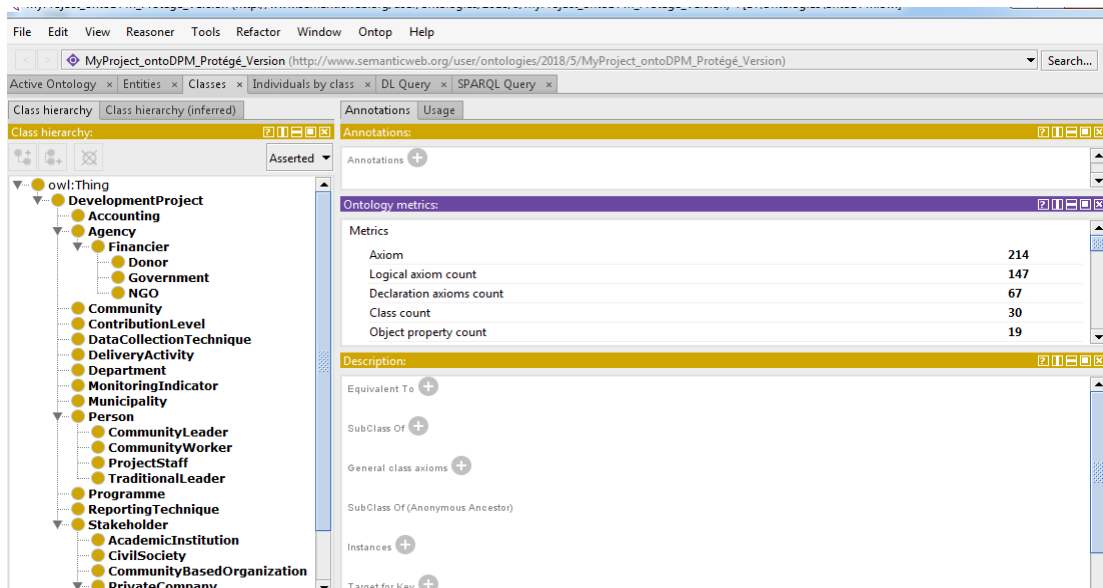


Figure 5.1: View of OntoDPM Ontology in Protégé

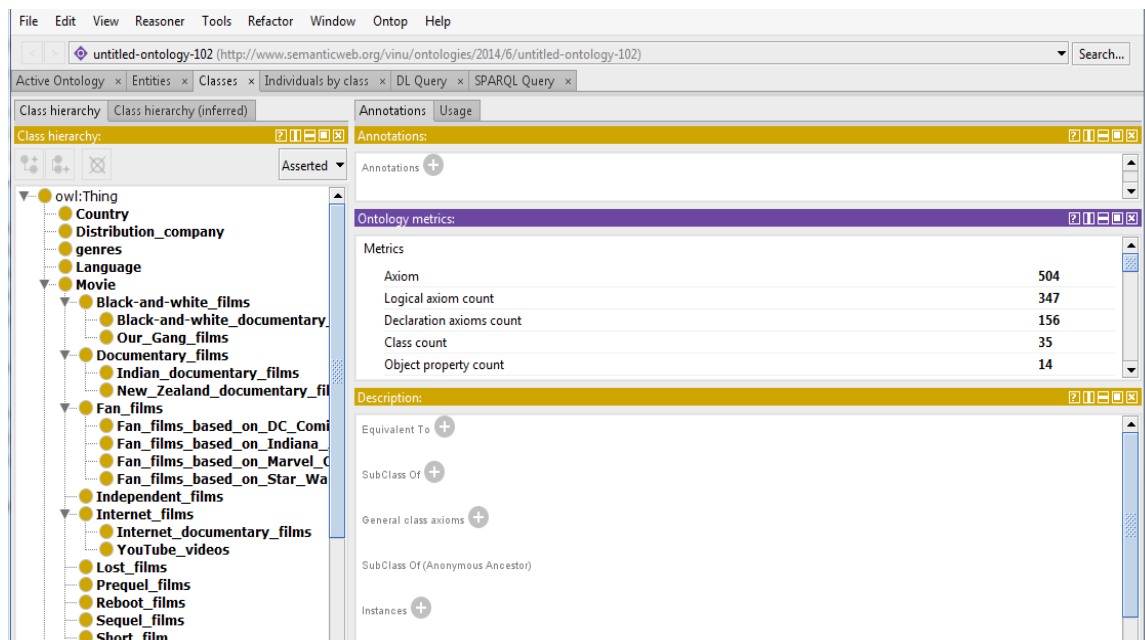


Figure 5.2: View of WikiMovie Ontology in Protégé



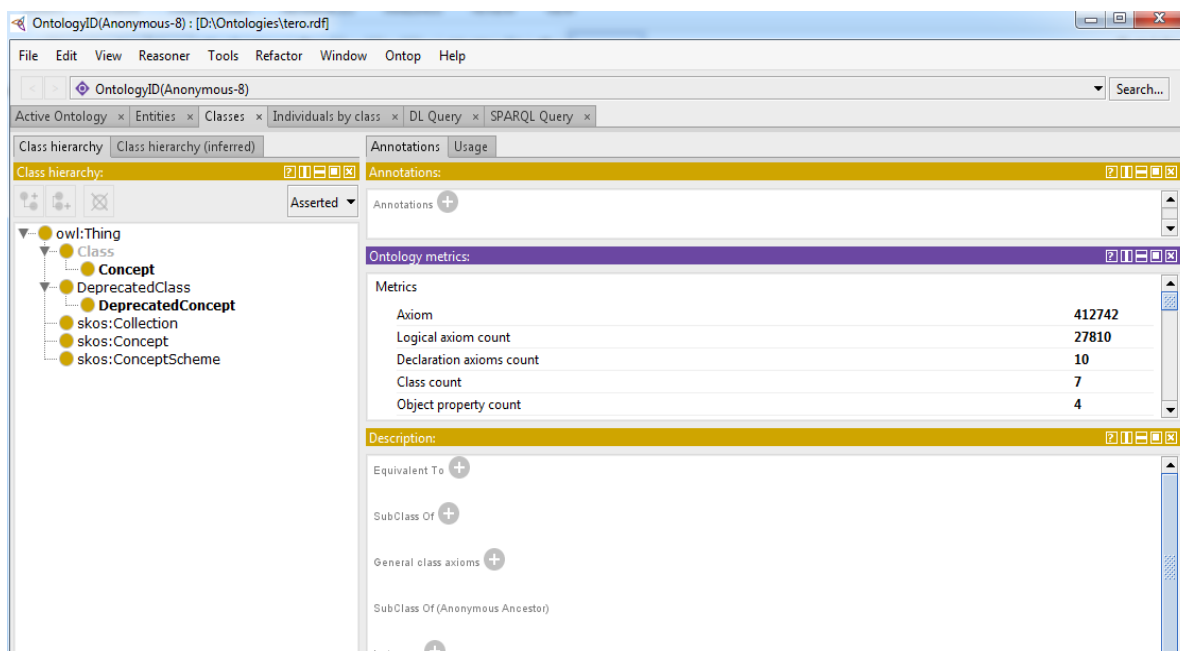


Figure 5.3: View of Tero Ontology in Protégé

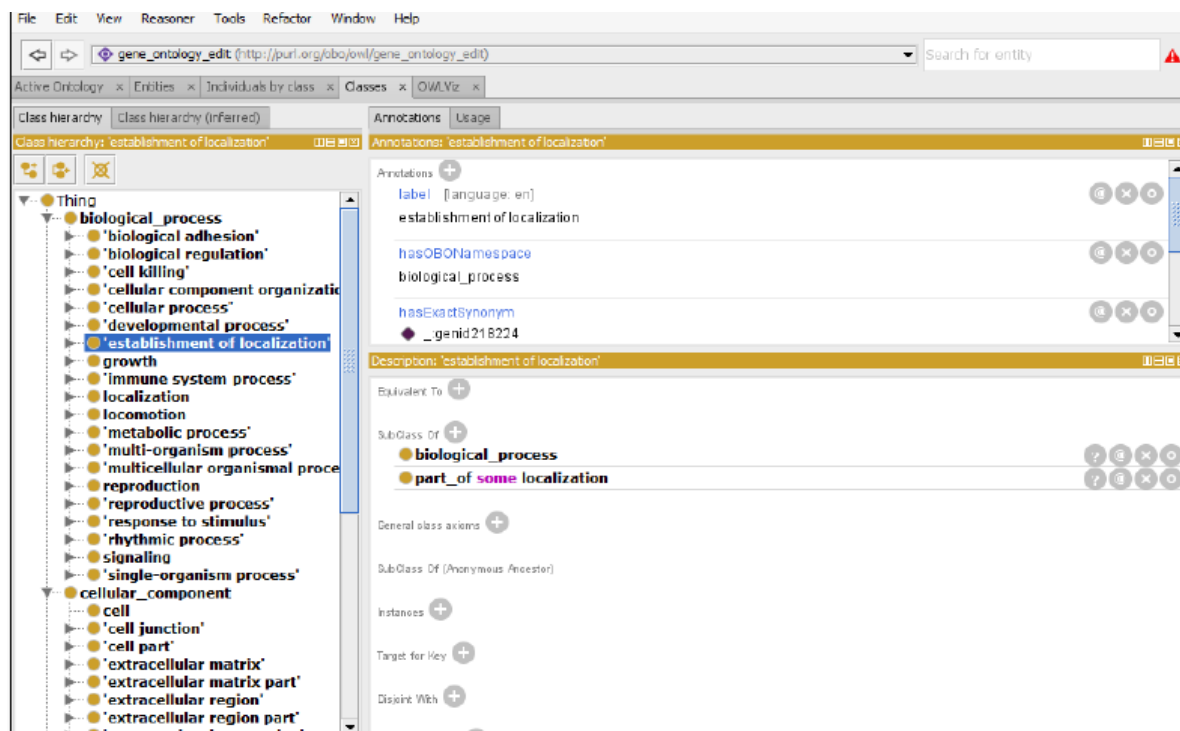
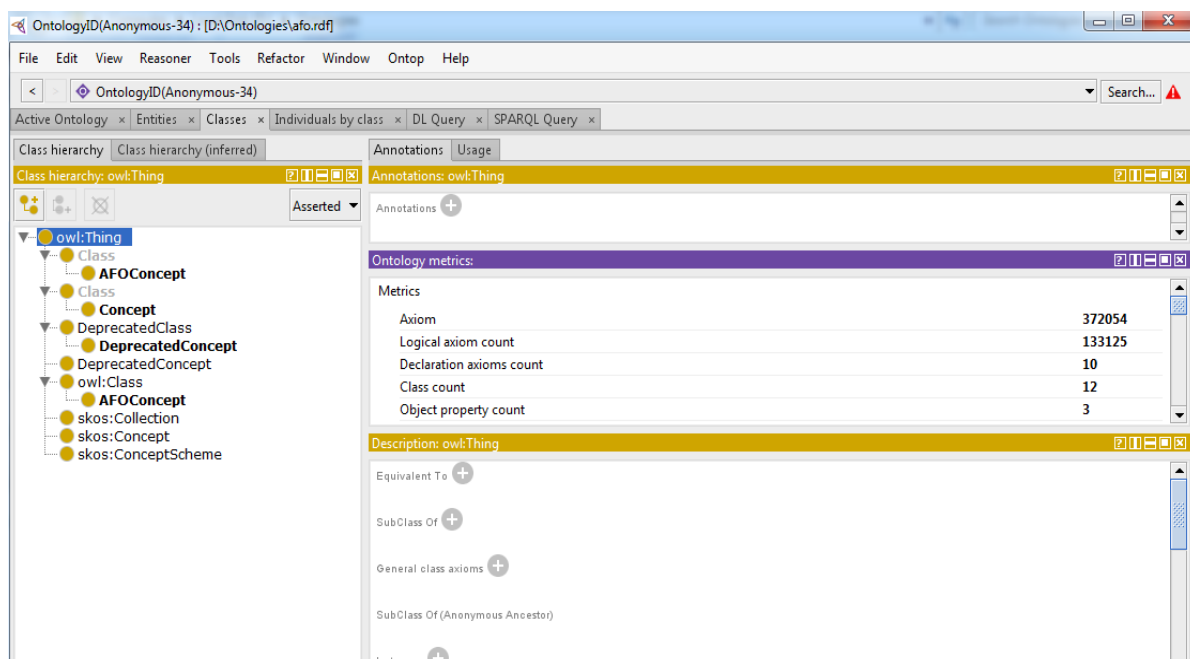
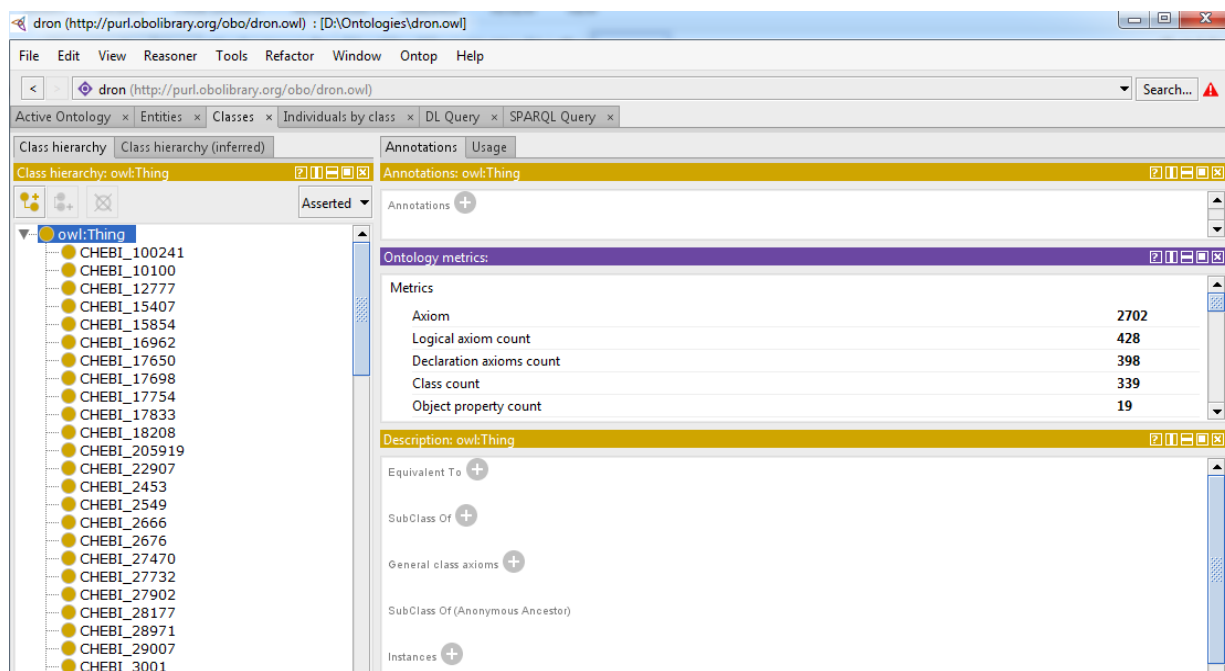


Figure 5.4: View of Gene Ontology in Protégé



**Figure 5.5:** View of Agriculture & Forensic Ontology (AFO) in Protégé



**Figure 5.6:** View of Drug Ontology in Protégé

### **5.2.2 Computer Hardware and Software Environments**

The experiments were carried out on a computer with the following specifications: Windows 7 operating system, 4.00 GB RAM, and Intel (R) Core (TM) i3-2350M CPU @2.30GHz processor.

The software utilized in the experiments include Microsoft Visual Studio 2010 Integrated Development Environment (IDE) and C Sharp (C#), an object oriented programming language compatible with dot net environment were used to write programs that process selected ontologies in two dot NET-based tools, namely, SemWeb.NET, dotNetRDF; Eclipse IDE and three open source tools for Java developers using Protégé, Jena API and RDF4J platforms.

### **5.3 Experimental Results**

This section presents and discusses the results from experiments conducted in this study. The experiments started by implementation of Ontology development in SemWeb.NET and dotNetRDF semantic web libraries. Furthermore, their capabilities in terms of programming construct, Ontology storage and querying supports, documentation level, usability and scalability are evaluated and compared. Thereafter all six ontologies from the dataset were loaded to be processed into five semantic web platforms including dotNetRDF, SemWeb.NET, Protégé, Jena and RDF4J. Also, the comparative results of these tools based on different criteria such as loading time; query execution time and query response time are experimented and presented in this section.

#### **5.3.1 Comparison of dotNetRDF and SemWeb.NET Libraries**

##### **a) Creation of RDF Ontology Graphs in dotNetRDF**

To develop any Ontology-based application with the dotNetRDF library, a dot NET Framework version 3.5 or higher is required; it provides a Common Language Runtime (CLR) to support application development in the dot NET environment as well as the Visual Studio Integrated Development Environment (IDE). The dotNetRDF is a Semantic Web library compatible with Microsoft Visual C# .NET which is freely available to download. The core classes for Ontology development in dotNetRDF are included in the VDS.RDF namespace. These core classes are based either on interfaces or abstract classes. These interfaces are INode, IGraph and ITripleStore. The codes used to implement RDF triples are presented in Table 5.2:

**Table 5.2:** Sample Codes Used to implement RDF triples in dotNetRDF

```
1. Graph g = new Graph ();
2. foreach (Triple t in g.Triples)
3. {
4.     Console.WriteLine(t.ToString());
5. }
6. TurtleWriter turtle = new TurtleWriter ();
7.     turtle.Save (g, "OntoDPM.ttl");
8.     NTriplesWriter ntriple = new NTriplesWriter ();
9.     ntriple.Save (g, "OntoDPM.nt");
10. RdfXmlWriter rdfxml = new RdfXmlWriter();
11.     rdfxml.Save (g, "OntoDPM. rdf");
12. Console.ReadLine ();
```

Although, dotNetRDF can create RDF triples, it has also capabilities of importing other Ontology files from different sources as illustrated in Table 5.3.

**Table 5.3:** Sample Codes Used to Import RDF Triples in dotNetRDF

```
1. Graph g = new Graph ();
2. FileLoader.Load (g, "OntoDPM.rdf");
3. foreach (Triple t in g.Triples) {
4.     Console.WriteLine (t.ToString ());
5. }
```

The above lines of codes illustrate the importation of multiple formats of RDF files and their storage into the dotNetRDF library. In these codes, g is an instance of the Graph class, it stores all Ontology triples. The FileLoader.Load() method is used to load Ontology in g and a loop is used to read the triples of the g object line by line and display them of the console with the Console.WriteLine() method.

#### **b) Creation of RDF Ontology Triples in SemWeb.NET**

SemWeb.NET is a Semantic Web library written in C# for processing and manipulating RDF data in Microsoft .NET platform. The SemWeb.Net library provides useful classes for reading, writing and querying ontologies. The latest version used in this study is 1.0.7 released in 2010; since then, no updated version has been released. The core classes for handling ontologies in the

SemWeb.NET library include SemWeb, SemWeb.MySQLStore, and SemWeb.Sparql. The SemWeb namespace consist of four subclasses which provide the functionalities for all aspects of ontologies development in SemWeb.NET including Resource, Statement, StatementSource and StatementSink. To construct a triple in SemWeb.NET, the constructor of the Statement class is used to define the Subject, Predicate and Object as shown in sample code presented in table 5.4.

**Table 5.4:** Sample Codes Used to Construct Subject, Predicate and Object in SemWeb.NET

```
1. ...Add new Statement (
2. new Entity ("http://OntoDPM/Person"),
3. new Entity ("http://OntoDPM/isA"),
4. new Literal ("Project Manager")
5. );
```

SemWeb.NET can import other Ontology files of different format (Ntriples, Turtle, Notation 3, RDF/XML etc.) as shown in Table 5.5.

**Table 5.5:** Sample Codes Used to Import different RDF Formats in SemWeb.NET

```
1. MemoryStore mem = new MemoryStore ();
2. mem.Import (new RDFReader ("OntoDPM.rdf"));
3. foreach (Statement stmt in mem) {
4. Console.WriteLine (stmt.ToString ());
5. }
6. MemoryStore data = new MemoryStore ();
7. data.Import (new N3Reader (OntoDPM.rdf));
```

In the above lines of codes an object of MemoryStore class called mem is created to store all RDF statements using RDF Reader parser to read RDF/XML files. Thereafter a loop is used to read all the statements in the files and display them on the console with the Console.WriteLine function.

The above analysis of the implementation and import of ontologies shows that both SemWeb.NET and dotNetRDF uses the same approach to store RDF triples. In SemWeb.NET, RDF triple is defined as a statement while in dotNetRDF RDF triple is defined as a graph. Furthermore, the two libraries provide many methods to read RDF data either from the file system or from repositories. However, with SemWeb.NET the user must determine which method can be used to parse RDF

files in different formats such as Ntriple, Turtle, Notation 3, RDF/XML etc., unlike dotNetRDF which provide an automatic parser based on the type of the file to be processed.

The availability criterion shows that the two platforms are open-sources and can be downloaded free of charge from the internet. The comparative results reveal that the two platforms are standalone and can only be used on local computers. However, both libraries also have some dissimilarities in their functionalities such as the reasoning system where dotNetRDF use the inference engine while SemWeb use Euler. SemWeb runs on Windows, Macintosh and Linux while dotNetRDF runs only on Windows system. Also, dotNetRDF is an ongoing project while SemWeb seems to have been discontinued. The full comparative results of analysis of SemWeb and dotNetRDF libraries is provided in Table 5.6.

**Table 5.6:** Comparison Results of SemWeb.NET and dotNetRDF

<b>Ontology Metrics Names</b>	<b>dotNetRDF</b>	<b>SemWeb</b>
Tool Architecture	Standalone	Standalone
Query Supports	SPARQL	SPARQL
Ontology Storage	Files, DBMS	Files, DBMS
Interoperability with Another Tool	Yes	Yes
Import/Export to/From	RDF, Turtle, N-triples, N3	RDF, Turtle, N-triples, N3
Supporting Platform	Windows (32 and 64 bit)	Windows (32 and 64 bit) Linux & Macintosh via Mono
Supporting File Formats	RDF, Turtle, N-Triple, Xml	RDF, Turtle, N-Triple, xml
Graphical User Interface	Available	Not available
Reasoner	Inference Engine	Euler
Programming Languages interface	Interface with C# program	Interface with C# program
Availability	Free	Free
Multi user support	Not Supported	Not Supported
Consistency check	Supported	Supported
Language supported to define	English	English
Synonyms		
OWL format	Not Supported	Not Supported
RIF (Rule Interchange Format)	Not Supported	Not Supported
Report generation support	Not Supported	Not Supported
Chart Generation support	Not Supported	Not Supported
Graphical editor support	Supported	Not Supported
Scalability	Small	Small
Extensibility	Supported	Supported

Ontology version comparison facility	Not Supported	Not Supported
Database support	Virtuoso, MySQL, Sesame Fuseki, 4Store, AllegroGraph	Sql-backed Persistent Stores, MySQL, PostgreSQL, SQL Server, SQLite
Back-up Management	Not Supported	Not Supported
Last stable version	2.1.0	1.07
Documentation Level	Very Good	Good
Dynamic help facility	Not available	Not available
Syntactic validation	Not Supported	Not Supported
Cloning of concepts	Not Supported	Not Supported

---

## 5.4 Comparison Analysis of Dot NET and Open Source Environments

### 5.4.1 Analysis Results on Loading, Executing and Querying Ontologies

Performance evaluation is the last layer of the framework. It provides metrics used to evaluate the Ontology storage platform.

#### a) Loading Time:

Loading Time indicates the time taken by the tools to load an Ontology in their memory. This metric was computed by loading each Ontology ten times in the memory of the tool and calculating the average of all attempted loading times. The average was calculated using the formula below,

$$\text{Average Loading Time} = \frac{1}{n} \sum_{i=1}^n xi \quad (5.1)$$

Where n = 10 which is the total number of times we loaded an Ontology in the tool's memory and xi is the loading time per Ontology.

Table 5.7 presents the captured loading time of all ontologies; the first column, presents the name of Ontology; the second up to fifth column shows the average of loading time in milliseconds (ms) used by dotNetRDF, SemWeb, Protégé, Jena and RDF4J, respectively.



**Table 5.7:** Loading Time in dotNetRDF, SemWeb, Protégé, Jena and RDF4J

Ontology Name	Loading Time in dotNetRDF (hh:mm:ss.000)	Loading Time in SemWeb (hh:mm:ss.000)	Loading Time in Protégé (hh:mm:ss.000)	Loading Time in Jena (hh:mm:ss.000)	Loading Time in RDF4J (hh:mm:ss.000)
OntoDPM	00:00:54.957	00:00:23.679	00:00:02.394	00:00:02.291	00:00:02.170
WikiMovie	00:01:14.662	00:01:05.944	00:00:04.113	00:00:03.756	00:00:02.972
Gene	00:04:29.450	Unsupported format	00:01:12.945	00:00:37.362	00:01:57.619
Tero	00:05:09.948	00:03:23.359	00:00:40.323	00:00:27.265	00:00:57.816
AFO	00:04:34.999	00:03:51.176	00:00:30.673	00:00:22.380	00:00:51.410
Dron	OutOfMemoryException	01:02:49.764	00:01:48.892	00:00:56.911	00:01:47.210

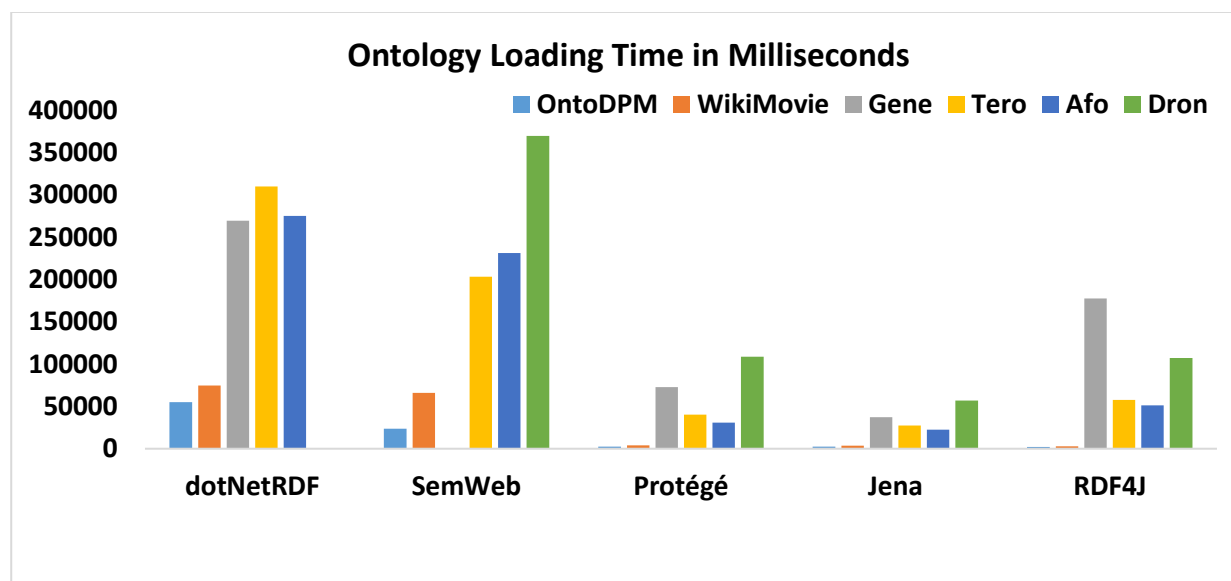
**Figure 5.7:** Loading Times of Ontologies in dotNetRDF, SemWeb, Protégé, Jena, and RDF4J

Figure 5.7 depicts the chart of the data in Table 5.6. The loading times are converted from hh:mm:ss.000 format to the standard unite of time, milliseconds. The chart shows the times taken by the ontologies to be loaded into the tools.

Even though dotNetRDF took longer, it managed to load the RDF statements stored in gene Ontology. Unlike SemWeb.NET that took less time to load small and medium ontologies. It failed to load gene Ontology as the fatal error of unsupported format occurred. Protégé performs better in terms of loading ontologies compared to other platforms even if it took more time to load gene

Ontology as well because of big size of Ontology. Also Jena and RDF4J performed well as their loading times are not much different to the loading times used by Protégé. However, Jena took less time to load Gene Ontology compared to Protégé and RDF4J.

#### b) Means of Query Response Time

Means of query response time refers to the time taken by the tools to return results of query. A sample SPARQL query on the dataset is given below.

```
SELECT? Ancestor WHERE { ? s rdfs: label "nucleus"@en;
    rdfs: subClassOf+? ancestor. }
```

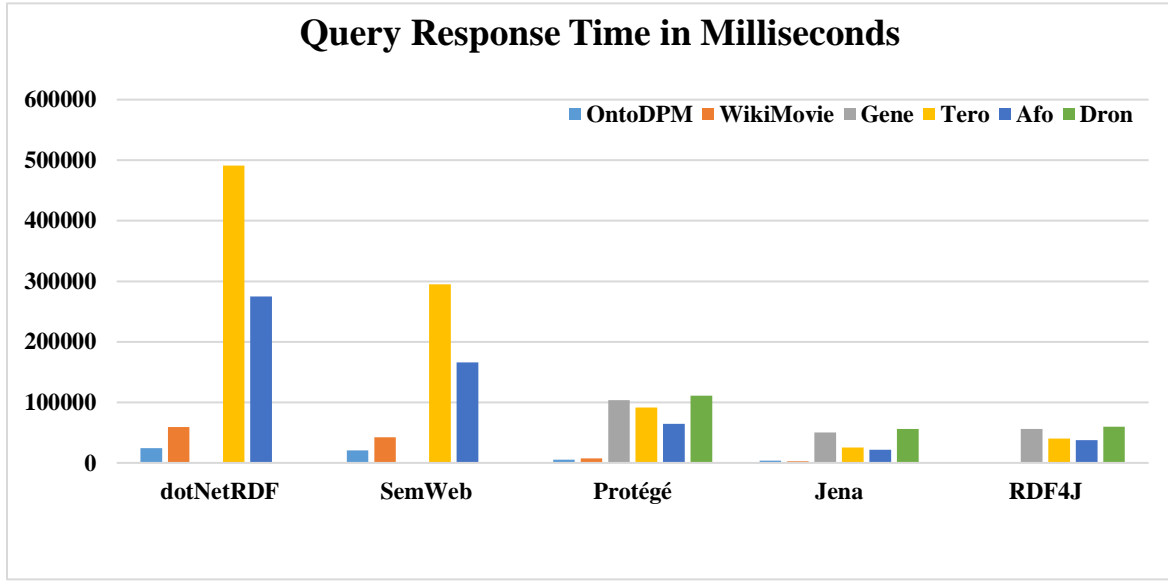
This sample query retrieves all super classes of the nucleus class in the gene Ontology. The results of the query response time on all the platforms are provided in Table 5.7. This was done by using the calculations from the formula below.

$$\text{Mean of Query Response Time} = \frac{1}{n} \sum_{i=1}^n xi \quad (5.2)$$

Where n = 10 which is the total number of times an Ontology responded to the query and xi is query response time per Ontology. Table 5.8 presents the captured query response time of all ontologies; the first column, presents the name of Ontology; the second up to fifth column shows the average of loading time in milliseconds (ms) used by dotNetRDF, SemWeb, Protégé, Jena and RDF4J, respectively.

**Table 5.8:** Means of Query Response Time in dotNetRDF, SemWeb, Protégé, Jena and RDF4J

Ontology Name	Query Response Time in dotNetRDF (hh:mm:ss.000)	Query Response Time in SemWeb (hh:mm:ss.000)	Query Response Time in Protégé (hh:mm:ss.000)	Query Response Time in Jena (hh:mm:ss.000)	Query Response Time in RDF4J (hh:mm:ss.000)
OntoDPM	00:00:24.551	00:00:20.889	00:00:05.471	00:00:03.510	00:00:00.900
WikiMovie	00:00:59.517	00:00:42.394	00:00:07.321	00:00:02.747	00:00:01.150
Gene	OutOfMemoryException	Unsupported format	00:01:43.547	00:00:50.363	00:0:56.011
Tero	00:08:11.322	00:04:54.783	00:01:31.425	00:00:25.227	00:00:40.049
AFO	00:04:34.657	00:02:46.192	00:01:04.502	00:00:21.502	00:00:37.836
Dron	OutOfMemoryException	OutOfMemoryException	00:01:51.126	00:00:56.143	00:00:59.956



**Figure 5.8:** Queries Response Times of Ontologies in dotNetRDF, SemWeb, Protégé, Jena and RDF4J

Fig. 5.8 Depicts the chart of query response time data recorded in Table 5.7. The results show that Jena, RDF4J and Protégé took less time to respond to the query compared to SemWeb and dotNetRDF. Furthermore, for dotNetRDF and SemWeb.NET no query respond time could be reported for the gene Ontology. In fact, both platforms failed to execute queries against the gene Ontology due to their limited memory sizes.

### c) Means of Query Execution Time

Means of Query Execution Time: This refers as the average of the time taken by the tools to execute a SPARQL query against ontologies in the dataset. A sample of such query is provided below.

prefix xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?s

WHERE { ?s oboInOwl:hasAlternativeId "GO:0050875"^^xsd:string }

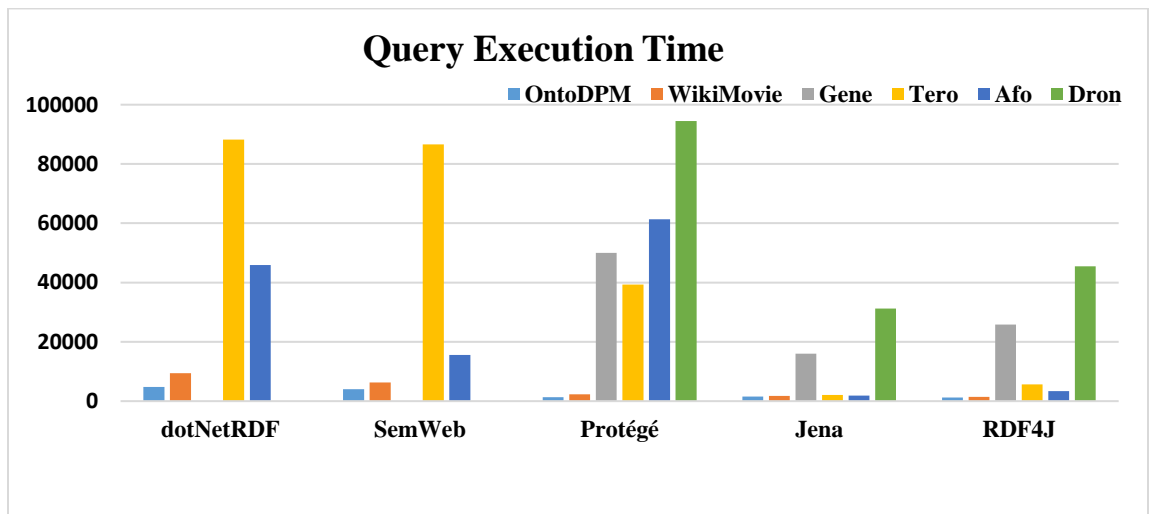
The first line of query above declares the namespace and the remaining lines that constitute the body of the query. The query selects the subject *s*, which has a property *hasAlternativeId* in gene Ontology. The calculations were made by using the following formula.

$$\text{Mean of Query Execution Time} = \frac{1}{n} \sum_{i=1}^n x_i \quad (5.3)$$

Where  $n = 10$  which is the total number of times we executed an Ontology in the tool's memory and  $x_i$  is execution time per Ontology. Table 5.9 presents the captured execution time of all ontologies, the first column, presents the name of Ontology; the second up to fifth column shows the average of loading time in milliseconds (ms) used by dotNetRDF, SemWeb, Protégé, Jena and RDF4J, respectively.

**Table 5.9:** Query Execution Time in dotNetRDF, SemWeb, Protégé, Jena and RDF4J

Ontology Name	Query Execution Time in dotNetRDF (hh:mm:ss.000)	Query Execution Time in SemWeb.NET (hh:mm:ss.000)	Query Execution Time in Protégé (hh:mm:ss.000)	Query Execution Time in Jena (hh:mm:ss.000)	Query Execution Time in RDF4J (hh:mm:ss.000)
OntoDPM	00:00:04.762	00:00:03.986	00:00:01.284	00:00:00.509	00:00:00.736
WikiMovie	00:00:09.457	00:00:06.270	00:00:02.336	00:00:01.781	00:00:01.400
Gene	OutOfMemoryException	Unsupported format	00:00:49.964	00:00:15.999	00:00:25.786
Tero	00:01:28.233	00:01:26.642	00:00:39.346	00:00:02.038	00:00:05.601
AFO	00:00:45.935	00:00:15.613	00:01:01.304	00:00:01.878	00:00:03.431
Dron	OutOfMemoryException	OutOfMemoryException	00:01:34.452	00:00:31.283	00:00:45.527



**Figure 5.9:** Queries Execution Times in dotNetRDF, SemWeb, Protégé, Jena and RDF4J

Fig. 5.9 represents the chart of query execution time in Jena, Protégé, RDF4J, dotNetRDF and SemWeb as presented in Table 5.10. SemWeb.NET and dotNetRDF libraries took longer to execute queries against ontologies in the datasets. Also, these libraries failed to execute queries on

gene Ontology because of the insufficient memory of dotNetRDF and inability to support owl format in SemWeb.NET library. Furthermore, RDF4J and Jena performed well in execution of queries as they took three time fast than Protégé, dotNetRDF and SemWeb.NET platforms.

**Table 5.10:** Summary of Comparison of Dot Net and Open Source Semantic Web Platforms

Features	Protégé	Jena API	RDF4J	SemWeb.NET	dotNetRDF
Developers	Stanford University	HP Labs	Aduna	Joshua Tauberer	Rob Vess
Last version	Protégé 5.5.0	Jena 3.12.0	rdf4j 2.5.4	SemWeb 1.07	dotNetRDF 2.2.0
Availability	Free, open source	Free, open source	Free, open source	Free to download	Free to download
Semantic web Architecture	Web based, Standalone & Client Server	Client/Server	Client/Server	Standalone	Standalone
Interoperability with other tools	Jena, Prompt, OKBC, aCT	Yes	Yes	No	Sesame, Allegro Graph
Query Support	SPARQL	SPARQL, RDQL	SPARQL, SeRQL	SPARQL	SPARQL
Ontology Storage mode	Memory, Files, DBMS	Memory, Files, DBMS	Memory, Files, DBMS	Memory, Files, DBMS	Memory, Files, DBMS
Import/Export Format	RDF(S), OWL	RDF(S), OWL	RDF(S), OWL	RDF, N3, TUTLE	RDF, N3, TUTLE
Build-in inference engine	Yes	Yes	Yes	Yes	Yes
Inference engine attached to the tool	Yes	Yes	Yes	No	No
Consistency checking	Yes	Yes	Yes	Yes	Yes
Stability	Stable	Stable	Stable	Abandoned	Stable
Extendibility	Yes, via plugs-in	Yes	Yes	Yes	Yes
Multiple users Support	Yes	No	No	No	No
Ontology library	Yes	Yes	Yes	Yes	Yes
Graphical User Interface	Supported	Not Supported	Supported	Not Supported	Supported
Web Support	Yes	No	No	No	No
OWL editor	Yes	Yes	Yes	No	No

Reasoners	Yes	Yes	Yes	Yes	Yes
Implemented in	Java	Java	Java	C#	C#
Backup management	No	No	No	No	No
Exception Handling	No	Yes	Yes	Yes	Yes
Operating System Support	Cross platform	Cross platform	Cross platform	Cross platform	Windows

## 5.5 Discussion

The comparative results of the evaluation of Protégé, Jena, RDF4J, SemWeb and dotNetRDF tools based on various criteria such the architecture of the tool, interoperability with other tool, import and export capabilities and many more.

It appears that the storage mode in Jena, Protégé and RDF4J are similar whereas SemWeb.NET and dotNetRDF use statements and graphs to store Ontology. Except for Protégé, RDF4J and dotNetRDF supports both command line and graphical user interfaces; this may help novice programmers to create, store, edit and query ontologies, whereas, Jena and SemWeb.NET only offer command line interface which requires skilled users who understand the syntax and the semantics of those libraries. Protégé, Jena, RDF4J and dotNetRDF are stable as they are frequently updated while SemWeb.NET seems to have been abandoned since it was last updated in 2010.

Except Protégé which offers a web based version, Jena and RDF4J are client/Server and standalone applications while SemWeb and dotNetRDF are provided as desktop applications. All the open source tools were able to import and export ontologies of different formats. However, dotNetRDF and SemWeb can import and export only small ontologies of RDF format. Although all the five tools can be extended, none of them provides a back up management feature. Jena and RDF4J use less time to execute and respond to queries compared to Protégé, SemWeb and dotNetRDF.

Jena, RDF4J, SemWeb.NET and dotNetRDF uses build in reasoners while Protégé use build in reasoners as well as other reasoners as plug-ins.

## 5.6 Conclusion

In this chapter, the dataset used in the experiments was presented. Various metrics including loading time, query execution time and query response time were used in running our experiment while recording the output values obtained. Thereafter, we conducted a comparative study on three

open source platforms, namely, Protégé, Jena, RDF4J and two dot NET libraries SemWeb.NET and dotNetRDF. These tools were used to parse and process ontologies of different sizes and format. Thereafter, the results were analyzed and used to compare the performance of the tools. The experiment shows that SemWeb and dotNetRDF performs well on medium size ontologies and can only process ontologies of RDF format. On the other hand, Protégé, Jena and RDF4J performs better in terms of query execution time and query response time on small and large ontologies. Therefore, there is much work to be done in the dot NET environment as they are still behind compared to open source environments.

## **CHAPTER 6. CONCLUSION AND FUTURE WORK**

### **6.1 Summary of the Study**

This study compared and analysed existing Semantic Web platforms for developing, storing and querying ontologies in both dot net and open source environments. In this chapter, Semantic Web Technologies was applied to develop Ontology in Microsoft dot NET. SemWeb.NET and dotNetRDF libraries are used to create RDF graphs of a selected Ontology. Thereafter, the features of two platforms were compared based on a number of metrics or criteria.

The experimental results show that there is a lack of features that can support Graphic User Interface in SemWeb.NET unlike dotNetRDF which provide rdfEditor a tool to edit ontologies and SPARQL GUI that enables the visual query of ontologies. Both libraries do not process ontologies of OWL format and their memory sizes are very small to store big ontologies. Also, in this study we conducted a comparative study on three open source platforms, namely, Protégé, Jena, RDF4J and two dot NET libraries SemWeb.NET and dotNetRDF. These tools were used to parse and process ontologies of different sizes and format.

Various metrics such as loading time, Query Execution Time, Query Response Time and Storage Capacity were empirically measured and used to determine the performance of each tool. Thereafter, the results were analyzed and used to compare the performance of the tools. The experiment shows that SemWeb and dotNetRDF performs well on medium size ontologies and can only process ontologies of RDF format. On the other hand, Protégé, Jena and RDF4J performs better in terms of query execution time and query response time on small and large ontologies. Therefore, there is too much work to be done in dot NET environment as they are still behind compared to open source environments.

### **6.2 Limitations and Recommendations for Future Work**

While conducting this study many challenges and limitations were encountered including:

- Both dotNetRDF and SemWeb.NET were not able to process ontologies at OWL level and their internal memory storage was very limited. However, small and medium ontologies were parsed and processed in both tools.
- The study used only two dot net and three open source Semantic Web platforms, namely, dotNetRDF, SemWeb.NET, Protégé, RDF4J and Jena API in the experiments. Further



research could focus on other existing Semantic Web platforms for storing and querying ontologies and evaluate their performances at large scale.

Future ideas to consider to extend this study includes: (1) developing a Graphical User Interface (GUI) to interact with ontologies in dot net environment , (2) developing plug-ins i.e. libraries in dot net environment for storing and querying ontologies in relational databases and (3) Providing a framework for developing, storing and querying OWL files in dot net environment.

### **6.3 Conclusion**

In this chapter Semantic Web Technologies was applied to develop Ontology in Microsoft dot NET and open source environments. SemWeb.NET and dotNetRDF libraries are used to create RDF graphs of a selected Ontology. Thereafter, the features of two dot net and three open source platforms were compared based on several metrics or criteria. The experimental results showed that there is a lack of features that can support Graphic User Interface in SemWeb.NET unlike dotNetRDF which provides rdfEditor as a tool to edit ontologies and SPARQL GUI that enables the visual query of ontologies. Both libraries do not process ontologies of OWL format and their memory sizes are very small to store big ontologies. Five ontologies of different sizes are used in the experiments. The experimental results showed that the open-source platforms provided more facilities for creating, storing and processing ontologies compared to the dot NET-based tools. Furthermore, the experiments revealed that Protégé and RDF4J open-source and dotNetRDF platforms provide both graphical user interface (GUI) and command line interface for Ontology processing, whereas, Jena open-source and SemWeb.NET are command line platforms. Moreover, the results showed that the open-source platforms are capable of processing multiple ontologies' files formats including RDF and OWL formats, whereas, the dot NET-based tools only process RDF ontologies. Finally, the experiments showed that the dot NET-based platforms have limited memories size as they failed to load, and query larges ontologies compared to open-source environments. Therefore, there is much work to be done in the dot NET environment as they are still behind compared to open source environments. This work will serve as guidelines for dot net and open source developers (Mahoro & Fonou-Dombeau, 2019; Mahoro & Fonou-Dombeau, 2020).

## REFERENCES

- ADRIAN, W. T., LIGEŻA, A., NALEPA, G. J. & KACZOR, K. 2014. Distributed and Collaborative Knowledge Management Using an Ontology-Based System. *In Proceedings of 1<sup>st</sup> IFIP International Workshop on Artificial Intelligence for knowledge Management (AI4KM)*. Montpellier, France, 28 August, pp.112-130.
- ALATRISH, E. 2013, "Comparison some of Ontology Editors. Management Information Systems," Vol.8, No.2, pp.018-024.
- AMATO, F., COLACE, F., GRECO, L., MOSCATO, V. & PICARIELLO, A. 2016. Semantic Processing of Multimedia Data for E-government Applications. *Journal of Visual Languages & Computing*, Vol.32, pp.35-41.
- AOKI-KINOSHITA, K. F., AOKI, N. P., FUJITA, A., FUJITA, N., KAWASAKI, T., MATSUBARA, M., OKUDA, S., SHIKANAI, T., SHINMACHI, D., SOLOVIEVA, E., SUZUKI, Y., TSUCHIYA, S., YAMADA, I. & NARIMATSU, H. 2017. Latest Developments in Semantic Web Technologies Applied to the Glycosciences. *Perspectives in Science*, Vol. 11, pp.18-23.
- APUKE, OBERIRI. (2017). Quantitative Research Methods: A Synopsis Approach. *Arabian Journal of Business and Management Review (kuwait Chapter)*. Vol. 6. 40-47. 10.12816/0040336.
- BASTIAN, E., MATTHIAS, J. & WERNER, Q. 2017, "Ontology-Based Big Data Management" *Journal System*, Vol.5, No.45, pp.1-14.
- BEISSWANGER, E. S., SCHULZ B., STENZHORN, H. & HAHN, U. 2008. "BIOTOP: An upper domain Ontology for the life sciences: a description of its current structure, contents, and interfaces to OBO ontologies", *Journal of Applied Ontology- Towards a meta Ontology for the Biomedical Domain*, Vol. 3, No. 4, pp.205-212.
- BERNERS-LEE, T., HANDLER, J. & LASSILA, O. 2001. The Semantic Web. *Scientific American*, May 2001, 29-37
- BROEKSTRA, J., KAMPMAN, A. & VAN HARMELEN, F. 2002. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. *In Proceedings of the First International Semantic Web Conference (IWC2002)*, Sardinia, Italy, 9-12 June, pp. 1-16.

- BURANARACH, M., SUPNITHI, T., THEIN, Y. M., RUANGRAJITPAKORN, T., RATTANASAWAD, T., WONGPATIKASEREE, K., LIM, A. O., TAN, Y. & ASSAWAMAKIN, A. 2016: An Ontology Application Management Framework for Simplifying Ontology-Based Semantic Web Application Development. *International Journal of Software Engineering and Knowledge Engineering*, Vol.26, No.1, pp.115-145.
- CARBON, S., DIETZE, H., LEWIS, S. E., MUNGALL, C. J. & MUNOZ-TORRES, M. C. 2017. Expansion of the Gene Ontology knowledgebase and resources. *Nucleic Acids Research*, 45, D331-D338.
- CHANDRASEKARAN, B., JOSEPHSON, J.R. BENJAMINS, V. R 1999.What are ontologies and why do we need them? *IEEE Intelligent Systems and their application*. Vol.14 No.1, pp.20-26.
- CHANDRASEKARAN, B., JOSEPHSON, J.R., BENJAMINS, V.R. 1999. “What are ontologies and why do we need them?” *IEEE Intelligent Systems and their application*, Vol.14 No.1, pp.20-26.
- D'AQUIN, M. & NOY, N. F. 2012. Where to publish and find ontologies? A survey of Ontology libraries. *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol.11, pp. 96-111.
- DARWISH, A. 2011. “The Impact of new Web2.0 Technologies in Communication, Development and Revolutions of Societies,” *Journal of Advances in Information Technology*. Vol.2, No.4, pp.204-216.
- DUINEVELD, A. J., STOTER, R., WEIDEN, M. R., KENEP, B. & BENJAMINS, V. R. 2000. “Wondertools? A comparative study of ontological engineering tools,” *International Journal of Human-Computer Studies*, Vol.52, No.6, pp.1111–1133.
- FLUIT, C., SABOU, M. & VAN HARMELEN, F. 2003. Ontology-Based Information Visualization. *Visualizing the Semantic Web – XML – based International and Information Visualization*, Vladimir Geroimenko and Chaomei Chen (eds), Springer

FONOU-DOMBEU, J.V & HUISMAN, M. 2011. Combining Ontology Development Methodologies and Semantic Web Platforms for E-government Domain Ontology Development. *International Journal of Web and Semantic Technology (IJWeST)* Vol .2, No.2, pp .12-25.

FONOU-DOMBEU, J.V, 2010. A Conceptual Ontology for E-government Monitoring of Development Projects in Sub Saharan Africa, *In Proceedings of the Information Society Technologies of Africa (IST-Africa)* Durban, South Africa, 19-21 May, pp. 1-8.

GARCÍA-CASTRO, R. & GÓMEZ-PÉREZ, A. 2005. Guidelines for Benchmarking the Performance of Ontology Management APIs. (ISWC) 4th International Semantic Web Conference.

GKOUTOS, G. V., SCHOFIELD, P. N. & HOEHNDORF, R. 2017. The anatomy of phenotype ontologies: principles, properties and applications Briefings in Bioinformatics 1-14.

GRUBER, T. 1993, “A translation Approach to Portable Ontology Specifications,” *International Journal of Knowledge Acquisition for Knowledge-based Systems*, Vol.5, No.2, pp. 199-220.

GRUBER, T. 1993. A translation Approach to Portable Ontology Specifications. *International Journal of Knowledge Acquisition for Knowledge-based Systems*, Vol.5, No.2, pp 199-220.

HIREMATH, B. K. & KENCHAKKANAVAR, A. Y. 2016 “An Alteration of the web 1.0, web 2.0 and web 3.0,” *Imperial Journal of Interdisciplinary Research (IJIR)*, India, Vol. 2, No. 4, pp. 705-710.

HORROCKS, I. 2008. Ontologies and the Semantic Web. *Communication of the ACM*, Vol.51, No.12, pp.58-67.

<https://github.com/bpellens/owldotnetapi>

<https://github.com/dotnetrdf/dotnetrdf/wiki/UserGuide-Getting-Started>, (Last Accessed May 30, 2016). J. Tauberer, “SemWeb.NET: Semantic Web/RDF Library for C#.NET,” Available at:

<https://github.com/JoshData/semweb-dotnet>, (Last Accessed: Aug 12, 2014).

<https://github.com/mdesalvo/RDFSharp>

- HUANG, Y. & DENG, G. 2010. Research on Storage of Geo-Ontology in Relational Database. *2<sup>nd</sup> International Symposium on Information Engineering and Electronic Commerce*, 23-25 July pp. 1-4.
- IACOB, A. 2009. .Net Framework RDF APIs [Online]. Available: [https://www.slideshare.net/andrei\\_i/net-rdf-apis](https://www.slideshare.net/andrei_i/net-rdf-apis) [Accessed 20/07 2017].
- ISLAM, N., SIDDIQUI, M. S. & SHAIKH, A. Z. 2010. TODE: A Dot Net based Tool for Ontology Development and Editing. *2<sup>nd</sup> International Conference on Computer Engineering and Technology (ICCET)*, Chengdu, China. pp. 229-233.
- JAMES R. EVANS, MATTHEW W. FORD, SUZANNE S. MASTERSON, HARRY S. HERTZ. 2012. beyond performance excellence: research insights from Baldrige recipient feedback. *Total Quality Management & Business Excellence* Vol. 23, No.5-6, pp. 489-506.
- JOVANOVIĆ, J., GAŠEVIĆ, D., KNIGHT, C. & RICHARDS, G. 2007. Ontologies for effective use of context in e-learning settings. *Educational Technology and Society* 10, 47-59.
- KAUR, L. & MISHRA, A. 2017. Software Component and the Semantic Web: An in-Depth Content Analysis and Integration History. *Journal of Systems and Software*, Vol.125, pp.152-169.
- KIONG, Y.C., PALANIAPPAN, S. & YAHAYA, N.A. 2009 Health Ontology Generator: Design and Implementation. *International Journal of Computer Science and Network Security*, Vol.9, No.2, pp.104–112.
- KONSTANTINIDIS, S. T., WHARRAD, H., WINDLE, R. J. & BAMIDIS, P. D. 2017. Semantic Web, Reusable Learning Objects, Personal Learning Networks in Health: Key Pieces for Digital Health Literacy. *Informatics Empowers Healthcare Transformation*. Vol.238, pp.219-222.
- LABIB, A. E., CANÓS, J. H. & PENADÉS, M. C. 2017. On the way to learning style models integration: A Learner's Characteristics Ontology. *Computers in Human Behavior*, 73, 433-445.
- LAMBRIX, P., HABBOUCHE, M. & PEREZ, M. 2003. Evaluation of Ontology Development Tools for Bioinformatics. *Bioinformatics*, Vol.19, pp.1564-1571.
- LIVINGSTONE, S. 2012. Critical Reflections on the Benefits of ICT. *In Education. Oxford Review of Education*, Vol.38, No.1, pp. 9-24.

- LU, J., RUAN, D. & ZHANG, G. 2007. E-Service Intelligence: *An Introduction, Studies in Computational Intelligence (SCI)*, Vol.37, pp.1-33.
- MARCO. L.C. “The Design Research Methodology as Framework for Development of a Tool for Engineering Design Education. *International Conference on Engineering and Product Design Education 2&3, Norway September 2010*
- MARSHALL PA. Human subjects protections, institutional review boards, and cultural anthropological research. *Anthropol Q* 2003;76(2):269-85.
- MARTINS, C., OLIVEIRA, T. & POPOVIČ, A. 2014. Understanding the Internet Banking Adoption: A unified theory of acceptance and use of technology and perceived risk application. *International Journal of Information Management*, Vol.34, No.1, pp. 1-13.
- MATTHEWS, A. 2007. Semantic web visual designer for visual studio.NET [Online]. Available: <https://aabs.wordpress.com/2007/10/24/semantic-web-visual-designer-for-visual-studio-net/> [Accessed 25/07 2018].
- MAZILU, L.-A. & PINTILLIE, R. A.-S. 2009. RDF APIs Using .NET Framework SemWeb & dotNetRDF [Online]. Available: <https://www.slideshare.net/andreimazilu/net-rdf-apis-2618742> [Accessed 25/06/2017 2018].
- MEENACHI, N. M., & BABA, S. M. 2012. Web Ontology Editors for Semantic Web A Survey. *International Journal of Computer Applications*. Vol.53, No.12, pp. 12-16.
- NOY, N. F. & MUSEN, M. A. 2002. Evaluating Ontology-mapping tools: Requirements and Experience. *In Proceedings of the EKAW Workshop on Evaluation of Ontology Tools*. Siguenza, Spain, pp.1-14.
- OATES, B.J. (2005). *Researching information systems and computing*. Sage.
- OCHS, C., PERL, Y., GELLER, J., ARABANDI, S., TUDORACHE, T. & MUSEN, M. A. 2017. “An empirical analysis of Ontology reuse in Bio Portal,” *Journal of Biomedical Informatics* Vol.71, pp.165-177.
- OCHS, C., PERL, Y., GELLER, J., ARABANDI, S., TUDORACHE, T. & MUSEN, M. A. 2017. An empirical analysis of Ontology reuse in Bio Portal. *Journal of Biomedical Informatics*, Vol.71, pp.165-177.

OPEN ANZO NOTES (2015), <http://www.openanzo.org/projects/openanzo/wiki/DBLayout> [accessed 11-November -2015]

PAK, J. & ZHOU, L. 2011. A Framework for Ontology Evaluation. *Exploring the Grand Challenges for Next Generation E-Business*: SHARMAN, R., RAO, H. R. & RAGHU, T. S. (eds.) Springer

RAFFAT, S.K., SIDDIQUI, M.S, SHAIKH, Z.A. & MEMON, A.R., A. 2012 Scientific Classification Technique. *International Journal of Engineering Research and Applications*. Vol.44, pp.63-68.

RIO, M., RIEL, A. & BRISSAUD, D. 2017. Design to Environment: Information Model Characteristics. *Procedia CIRP*, Vol.60, pp.494-499.

ROULEAU, G., GAGNON, M.-P. & CÔTÉ, J. 2015. Impacts of information and communication technologies on nursing care: *an overview of systematic reviews (protocol)*. *Systematic Reviews*, Vol.4, No.1, pp.1-8.

RUTA, M., SCIOSCIA, F., PINTO, A., GRAMEGNA, F., IEVA, S., LOSETO, G. & SCIASCIO, E. D. 2017. A CoAP-based framework for collaborative sensing in the Semantic Web of Things. *Procedia Computer Science*, Vol.109, pp. 1047-1052.

SANTOS, P. M. & ROVER, A. J. 2016. Knowledge Representation through Ontologies: an Application in the Electronic Democracy Field. *Perspectivas em Ciência da Informação*, Vol.21, No.3, pp.22-49.

SINGH, A. & ANAND, P. 2013 “State of Art in Ontology Development Tools,” *International Journal of Advances in Computer Science and Technology*, Vol.2, No.7, pp.96-101.

SLATER, L., GKOUTOS, G. V., SCHOFIELD, P. N. & HOEHNDORF, R. 2015. Using Aber-OWL for Fast and Scalable Reasoning over BioPortal Ontologies. *In Proceedings of International Conference on Biomedical Ontologies (ICBO)*, July, pp.72-76.

SLIMANI, T. 2015. Ontology Development: A Comparing Study on Tools, Languages and Formalisms. *Indian Journal of Science and Technology*, Vol.8, No.24, pp.1-12.

- SPLENDIANI, A., BURGER, A., PASCHKE, A., ROMANO, P. & MARSHALL, M. S. 2011. Biomedical Semantics in the Semantic Web. *Journal of Biomedical Semantics*, Vol.2, No.1, pp.1-9.
- SUN, D., JUNG, H., HWANG, C. & KIM, H. 2011. Accessing Information Sources using Ontologies. *International Journal of Computers, Communications & Control*, Vol.6, No.2, pp. 349-366.
- STEGMAIER, F., GRÖBNER, U., DÖLLER, M., KOSCH, H. & BAESE, G. (2009), “Evaluation of current RDF database solutions”, *In Proceedings of the 10<sup>th</sup> International Workshop on Semantic Multimedia Database Technologies (SeMuDaTe)*, 4th International Conference on Semantics And Digital Media Technologies (SAMT), Graz, Austria, 2-4 December, pp. 39-55.
- TAHA, K. 2013 “GOSeek: A Gene Ontology Search Engine using Enhanced Keywords” *In Proceedings of the 35th Annual International Conference of Engineering in Medicine and Biology Society (EMBC)*, Osaka, Japan, 3-7 July, pp.1502 – 1505.
- TAYE, M. 2010 “The State of the Art: Ontology Web-Based Languages: XML Based,” *Journal of Computing*, NY, USA, June 2010, Vol. 2, No. 6, ISSN 2151-9617.
- TAYE, M. M. 2010. Understanding Semantic Web and Ontologies: Theory and Applications. *Journal of Computing*, Vol.2, No.6, pp.182-192.
- TAUBERER, J. 2009 “SemWeb.NET: Semantic Web/RDF Library for C#.NET,” Available at: <https://github.com/JoshData/semweb-dotnet>, (Last Accessed: Aug 12, 2019).
- TEIMOURIKIA, M. & FUGINI, M. 2017. Ontology Development for Run-Time Safety Management Methodology in Smart Work Environments Using Ambient Knowledge. *Future Generation Computer Systems*, Vol.68, pp.428-441.
- UTHAYAN, K. R. & ANANDHA MALA, G. S. 2015. Hybrid Ontology for Semantic Information Retrieval Model Using Keyword Matching Indexing System. *The Scientific World Journal*.
- VEGETTI, M., ROLDÁN, L., GONNET, S., LEONE, H. & HENNING, G. 2016. A Framework to Represent, Capture, and Trace Ontology Development Processes. *Engineering Applications of Artificial Intelligence*, vol.56, pp. 230-249.

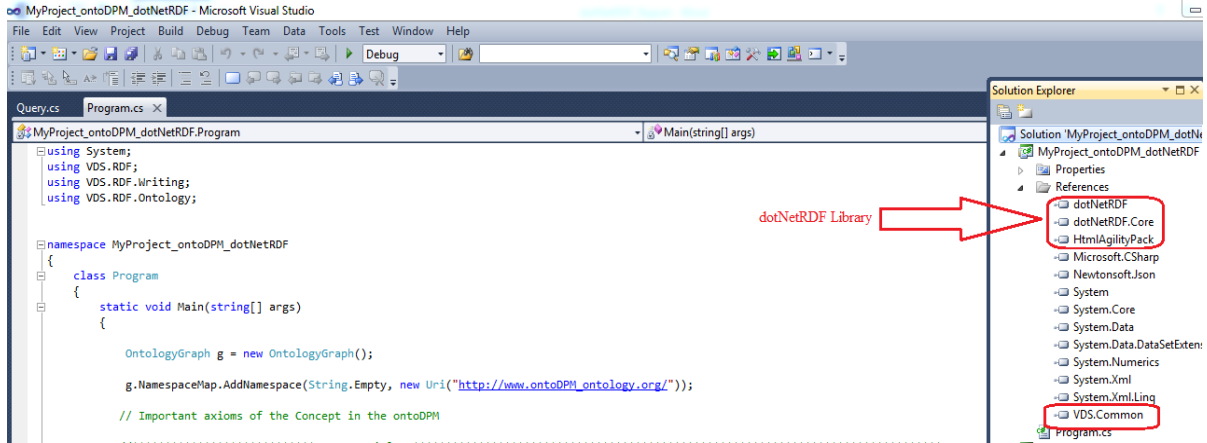


- VESSE, R. 2010 “dotNetRDF Documentation,” Available at: <https://github.com/dotnetrdf/dotnetrdf/wiki/UserGuide-Getting-Started>, (Last Accessed May 30, 2019).
- XIAO, Y., XIAO, M. & ZHAO, H. An Ontology for e-Government Knowledge Modeling and Interoperability. 2007 International Conference on Wireless Communications, Networking and Mobile Computing, 21-25 Sept. 3605-3608.
- YOUN, S. & MCLEOD, D. 2006. Ontology development tools for Ontology-based knowledge management Hershey, PA, USA: Idea group Inc.
- ZANG, J. 2007, “Ontology and the Semantic Web. Proceedings of the North American Symposium on Knowledge Organization,” Vol.1, Available: <http://dlist.sir.arizona.edu/1897/>
- ZENUNI, X., RAUFI, B., ISMAILI, F. & AJDARI, J. 2015. State of the Art of Semantic Web for Healthcare. *Procedia - Social and Behavioral Sciences*. Vol.195, pp.1990-1998.

## APPENDIX A: FULL CODE OF ONTOLOGY DEVELOPMENT IN DOT NET ENVIRONMENT

### a) Creating RDF/XML file in dotNetRDF Library

We created a blank project called “MyProject\_ontoDPM\_dotNetRDF”, the Project is created under the Visual C# 4.0 framework (Client Profile) created by default once you open Visual Studio2010. After the creation of a project, dotNetRDF, the library to manipulate Ontology is added to the projects. After the library is added to the solution of the project, it appears under the Solution Explorer tab which is under the References folder as demonstrated in – Figure A.1



**Figure A. 1:** Adding dotNetRDF Library to the Project

The table presented below shows the full code of creation of a graph object where the Ontology triples will be stored. All necessary namespaces such as “VDS.RDF, VDS.RDF. Writing and VDS.RDF. Ontology” have to be imported in class Program as they contain all classes used to create graphs, triples (subjects, predicates and objects) and RDF instances (Individuals).

The following table is a sample of RDF/XML format, the output from the Semantic Web OntoDPM application generated by dotNetRDF Library.

**Table A.1:** RDF/XML file output of OntoDPM Ontology in dotNetRDF

---

```
<? Xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rdf:RDF [
```

---

---

```

<!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#>
<!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#>
<!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#>
<!ENTITY owl 'http://www.w3.org/2002/07/owl#>
]>
<rdf:RDF
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns="http://www.ontodpm_Ontology.org/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="http://www.ontodpm_Ontology.org/AcademicInstitution">
    <isA rdf:resource="http://www.ontodpm_Ontology.org/Stakeholder" />
  </rdf:Description>
  <rdf:Description rdf:about="http://www.ontodpm_Ontology.org/CivilSociety">
    <isA rdf:resource="http://www.ontodpm_Ontology.org/Stakeholder" />
  </rdf:Description>
  <rdf:Description rdf:about="http://www.ontodpm_Ontology.org/CommunityBaseOrganization">
    <isA rdf:resource="http://www.ontodpm_Ontology.org/Stakeholder" />
  </rdf:Description>
  <rdf:Description rdf:about="http://www.ontodpm_Ontology.org/CommunityBasedOrganization">
    <owns rdf:resource="http://www.ontodpm_Ontology.org/Community" />
  </rdf:Description>
  <rdf:Description rdf:about="http://www.ontodpm_Ontology.org/CommunityLeader">
    <resides rdf:resource="http://www.ontodpm_Ontology.org/Community" />
  </rdf:Description>
  <rdf:Description rdf:about="http://www.ontodpm_Ontology.org/CommunityWorker">
    <affiliates rdf:resource="http://www.ontodpm_Ontology.org/Community" />
    <isA rdf:resource="http://www.ontodpm_Ontology.org/Person" />
  </rdf:Description>
  <rdf:Description rdf:about="http://www.ontodpm_Ontology.org/Consultant">
    <isA rdf:resource="http://www.ontodpm_Ontology.org/PrivateCompany" />
  </rdf:Description>
  <rdf:Description rdf:about="http://www.ontodpm_Ontology.org/Contractor">
    <isA rdf:resource="http://www.ontodpm_Ontology.org/PrivateCompany" />
  </rdf:Description>
  <rdf:Description rdf:about="http://www.ontodpm_Ontology.org/DataCollection">
    <isIndividualOf rdf:resource="http://www.ontodpm_Ontology.org/FocusGroup" />
    <isIndividualOf rdf:resource="http://www.ontodpm_Ontology.org/SiteObservation" />
    <isIndividualOf rdf:resource="http://www.ontodpm_Ontology.org/Survey" />
  </rdf:Description>
  <rdf:Description rdf:about="http://www.ontodpm_Ontology.org/DeliveryActivity">
    <isIndividualOf rdf:resource="http://www.ontodpm_Ontology.org/Discussion" />

```

---

## b) Querying RDF/XML file in dotNetRDF Library

- Command Line Interface

The full code of querying RDF/XML file of OntoDPM model from application generated by dotNetRDF Library is presented in table A.3.

**Table A.2:** The full code of querying RDF file of Juho Ontology

---

```

using System;
using VDS.RDF;
using VDS.RDF.Parsing;
using VDS.RDF.Query;
using VDS.RDF.Storage;
using VDS.RDF.Writing;
using VDS.RDF.Query.Datasets;
using System.Diagnostics;
namespace TripleStoreOntoDPM
{
    class Program
    {
        static void Main(string[] args)
        {
            Stopwatch myTimer = new Stopwatch();
            myTimer.Start();
            TripleStore store = new TripleStore();
            store.LoadFromFile(@"C:\ontologies\juho.rdf");
            ISparqlDataset ds = new InMemoryDataset(store);
            //Execute a raw SPARQL Query
            //Should get a SparqlResultSet back from a SELECT query
            Object results = store.ExecuteQuery("SELECT * WHERE { ?s ?p ?o }");
            if (results is SparqlResultSet)
            {
                //Print out the Results
                SparqlResultSet rset = (SparqlResultSet)results;
                foreach (SparqlResult result in rset)
                {
                    Console.WriteLine(result.ToString());
                }
            }

            //Use the SparqlQueryParser to give us a SparqlQuery object
            //Should get a Graph back from a CONSTRUCT query
            SparqlQueryParser sparqlparser = new SparqlQueryParser();
            SparqlQuery query = sparqlparser.ParseFromString("CONSTRUCT { ?s ?p ?o } WHERE { ?s ?p ?o }");

            results = store.ExecuteQuery(query);
            if (results is IGraph)
            {
                //Print out the Results
                IGraph g = (IGraph)results;
                foreach (Triple t in g.Triples)
                {
                    Console.WriteLine(t.ToString());
                }
                myTimer.Stop();
                Console.WriteLine("Query Response Time taken " + myTimer.Elapsed);
                Console.WriteLine("Query Execution Time taken " + query.QueryExecutionTime.ToString());
                Console.ReadLine();

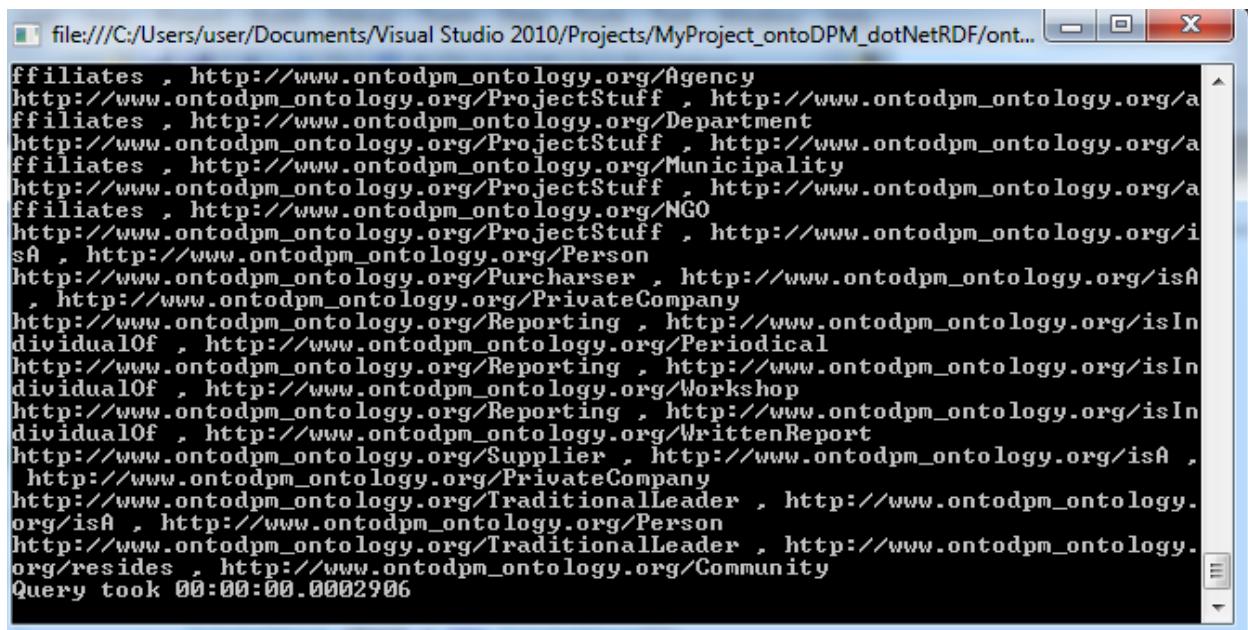
                TurtleWriter twriter = new TurtleWriter();
                twriter.Save(g, "Output.ttl");

                NTriplesWriter ntwriter = new NTriplesWriter();
                ntwriter.Save(g, "Output.nt");

                RdfXmlWriter rdfxmlwriter = new RdfXmlWriter();
                rdfxmlwriter.Save(g, "Output.rdf");
            }
        }
    }
}

```

---



```
ffiliates , http://www.ontodpm_ontology.org/Agency
http://www.ontodpm_ontology.org/ProjectStuff , http://www.ontodpm_ontology.org/a
ffiliates , http://www.ontodpm_ontology.org/Department
http://www.ontodpm_ontology.org/ProjectStuff , http://www.ontodpm_ontology.org/a
ffiliates , http://www.ontodpm_ontology.org/Municipality
http://www.ontodpm_ontology.org/ProjectStuff , http://www.ontodpm_ontology.org/a
ffiliates , http://www.ontodpm_ontology.org/NGO
http://www.ontodpm_ontology.org/ProjectStuff , http://www.ontodpm_ontology.org/i
sA , http://www.ontodpm_ontology.org/Person
http://www.ontodpm_ontology.org/Purchaser , http://www.ontodpm_ontology.org/isA
, http://www.ontodpm_ontology.org/PrivateCompany
http://www.ontodpm_ontology.org/Reporting , http://www.ontodpm_ontology.org/isIn
dividualOf , http://www.ontodpm_ontology.org/Periodical
http://www.ontodpm_ontology.org/Reporting , http://www.ontodpm_ontology.org/isIn
dividualOf , http://www.ontodpm_ontology.org/Workshop
http://www.ontodpm_ontology.org/Reporting , http://www.ontodpm_ontology.org/isIn
dividualOf , http://www.ontodpm_ontology.org/WrittenReport
http://www.ontodpm_ontology.org/Supplier , http://www.ontodpm_ontology.org/isA ,
http://www.ontodpm_ontology.org/PrivateCompany
http://www.ontodpm_ontology.org/TraditionalLeader , http://www.ontodpm_ontology.
org/isA , http://www.ontodpm_ontology.org/Person
http://www.ontodpm_ontology.org/TraditionalLeader , http://www.ontodpm_ontology.
org/resides , http://www.ontodpm_ontology.org/Community
Query took 00:00:00.0002906
```

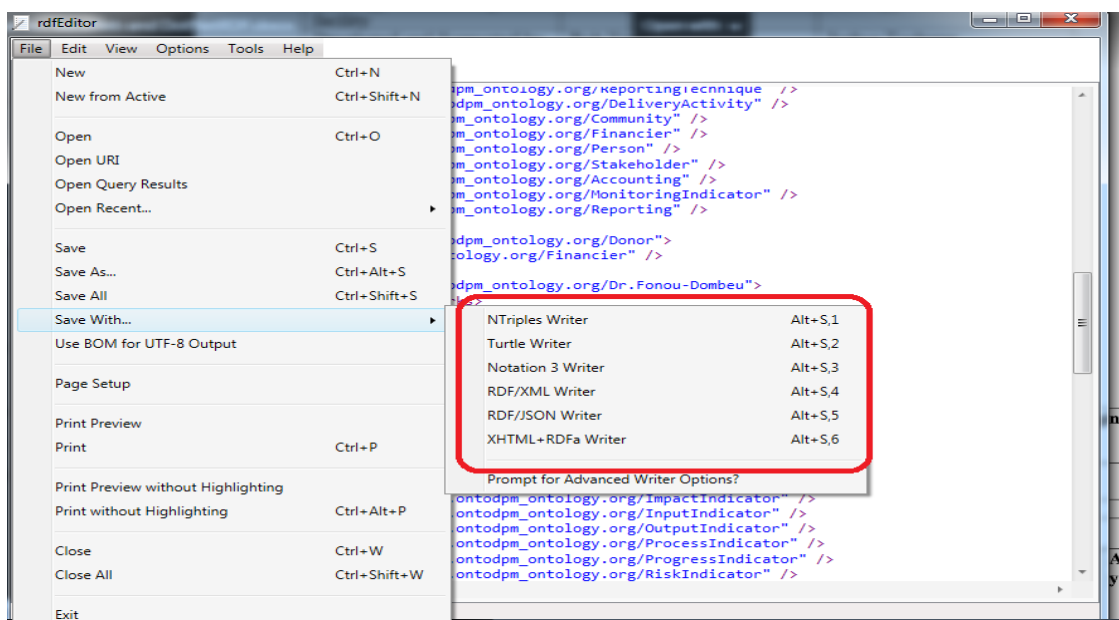
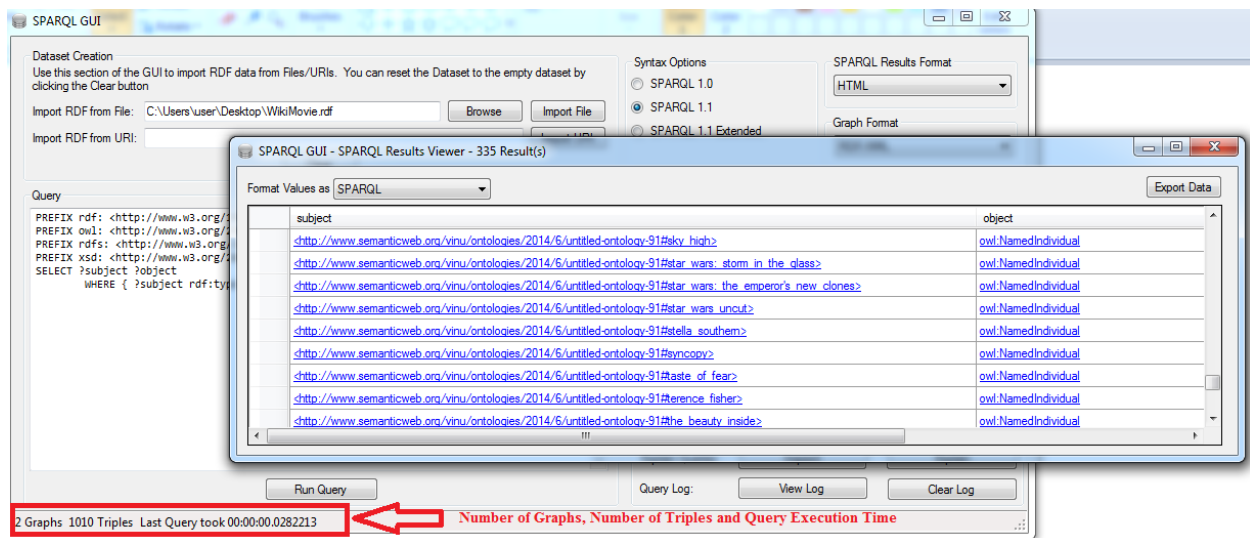
**Figure A. 2:** Sample Code of Query Execution Time in dotNetRDF Library

- Graphical User Interface

Also dotNetRDF provides other option of querying Ontology using Graphic User Interface (SPARQL GUI) which is the dotNetRDF build-in Tool to visually query ontologies. This Graphical User Interface is simple to use and straightforward as the interface provides all necessary information in terms of querying and display the query results in any format supported by dotNetRDF library.

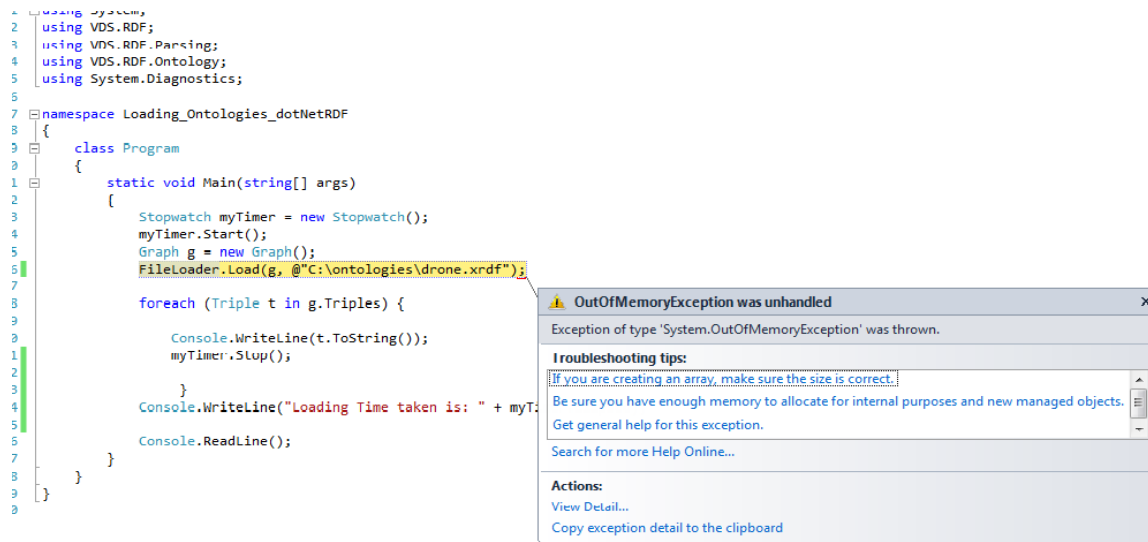
**Figure A. 3:** Querying WikiMovie Ontology in SPARQL GUI dotNetRDF

As shown in Figure 5.17 dotNetRDF provides the option of importing and exporting ontologies to/from different formats such as NTriples, Turtle, Notation3, RDF/XML etc.

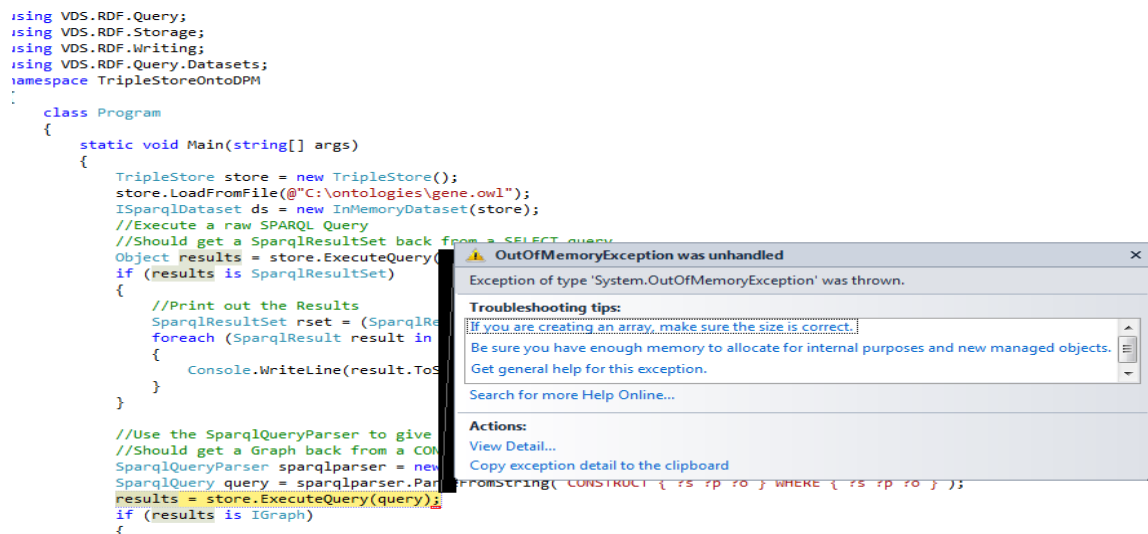


**Figure A. 4:** Export and Import features of dotNetRDF

dotNetRDF failed to load and query Drug and Gene Ontologies because of its original size which was much bigger to be processed in dotNetRDF library as depicted in Figures A. 5 and A. 6



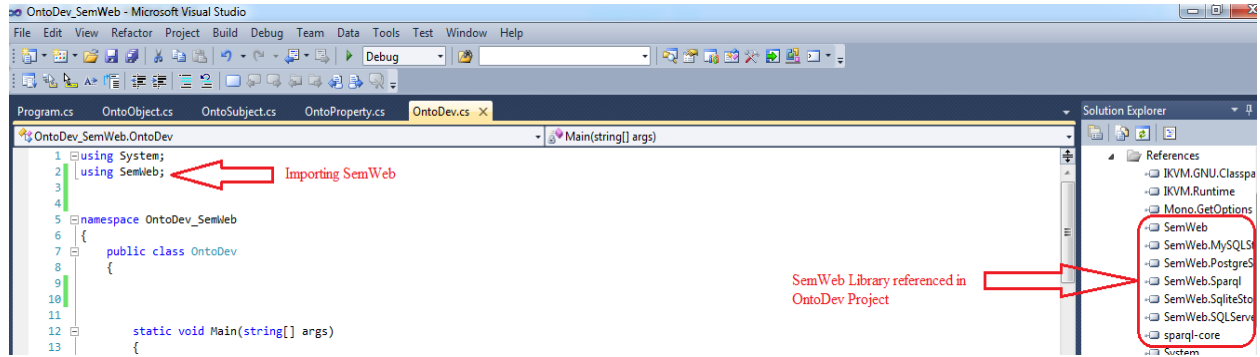
**Figure A. 5:** Out of memory exception while loading drug Ontology in dotNetRDF



**Figure A. 6:** Failure occurred while querying Gene Ontology in dotNetRDF

### c) Implementation of OntoDPM Ontology in SemWeb.NET

The SemWeb.NET library's namespaces includes: SemWeb, SemWeb.MySQLStore, SemWeb.Sparql are added under references section in the OntoDev project within Microsoft dot NET using C# language as shown in Figure A .7.



**Figure A. 7:** Adding SemWeb Library to the OntoDev Project

After adding the SemWeb.NET library to build RDF triples of OntoDPM Ontology, we created three main classes including OntoSubject.cs, OntoProperty.cs and OntoObject.cs were created under OntoDev\_SemWeb solution in OntoDev project as shown in Figure A. 8.

### Querying RDF/XML file in SemWeb.NET Library

The Ontology have been queried out using Simple Protocol and RDF Query Language (SPARQL). SPARQL is the standard query language for the Semantic Web which can be used to query over large volumes of RDF data and is W3C Recommendation. The SemWeb.NET library provides a great mechanism for querying Ontology files stored in MemoryStore using build-in SPARQL language.

**Table A. 3:** Full code for querying Ontology in SemWeb.NET Library

---

```

using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
using SemWeb;
using SemWeb.Query;
namespace QueryApp
{
    public class Example
    {
        public static void Main(string[] argv)
        {
            if (argv.Length < 3)
            {
                Console.WriteLine("Usage: query.exe format queryfile datafile");
                return;
            }

            string format = argv[0];
            string queryfile = argv[1];
            string datafile = argv[2];

```

---



---

```

Query query;

if (format == "rsquary")
{
    // Create a simple-entailment "RSquary" query
    // from the N3 file.
    query = new GraphMatch(new N3Reader(queryfile));
}
else
{
    // Create a SPARQL query by reading the file's
    // contents.
    query = new SparqlEngine(new StreamReader(queryfile));
}

// Load the data file from disk
MemoryStore data = new MemoryStore();
data.Import(new N3Reader(datafile));

// First, print results in SPARQL XML Results format...

// Create a result sink where results are written to.
QueryResultSink sink = new SparqlXmlQuerySink(Console.Out);

// Run the query.
query.Run(data, sink);

// Second, print the results via our own custom QueryResultSink...
query.Run(data, new PrintQuerySink());
}

public class PrintQuerySink : QueryResultSink
{
    public override bool Add(VariableBindings result)
    {
        foreach (Variable var in result.Variables)
        {
            if (var.LocalName != null && result[var] != null)
            {
                Console.WriteLine(var.LocalName + " ==> " + result[var].ToString());
            }
            Console.WriteLine();
        }
        return true;
    }
}

```

---

The capacity in terms of querying large ontologies in SemWeb.NET library is very limited as it fails to query out Drug Ontology file which has the size of 473 Mb as depicted in Figure A. 11.

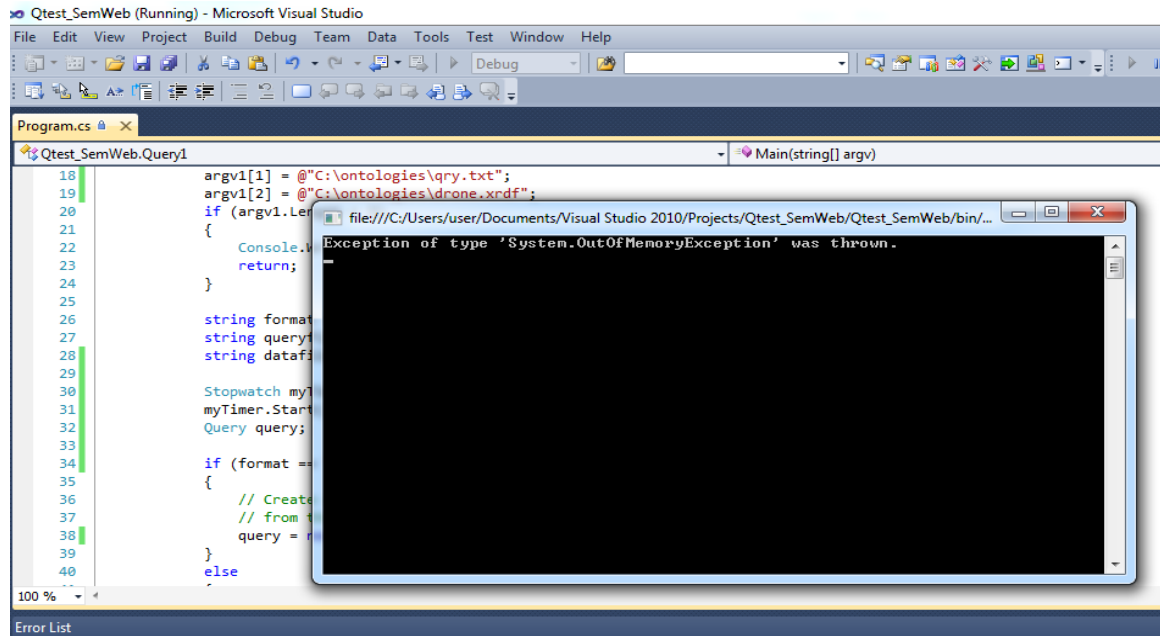


Figure A. 8: Fatal Error while Querying Drug Ontology in SemWeb.NET

## APPENDIX B: FULL CODE OF ONTOLOGY DEVELOPMENT IN OPEN SOURCE ENVIRONMENT

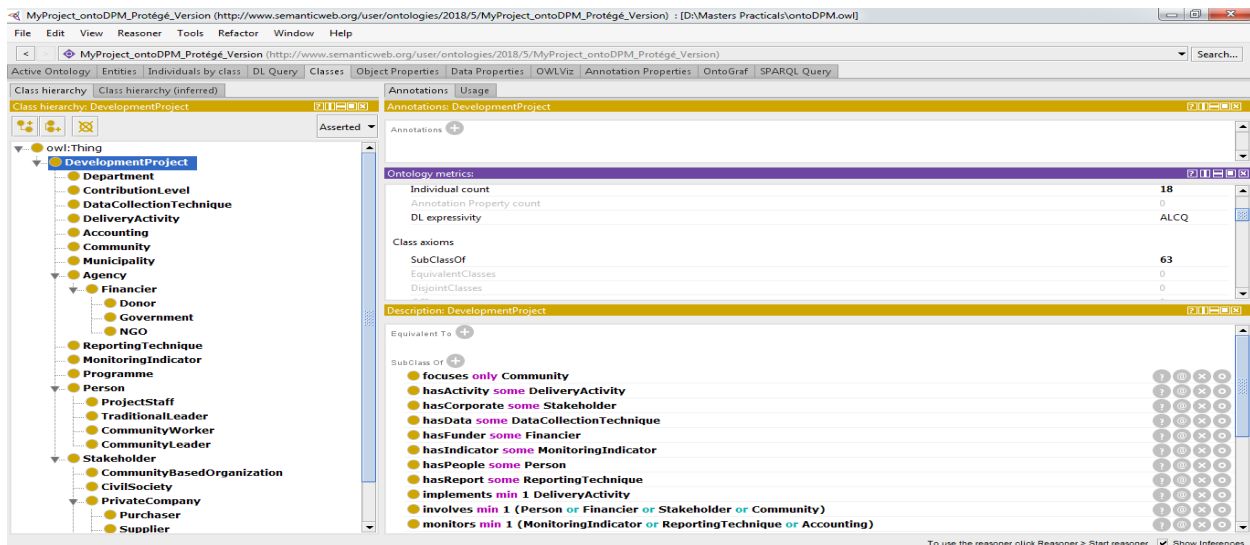
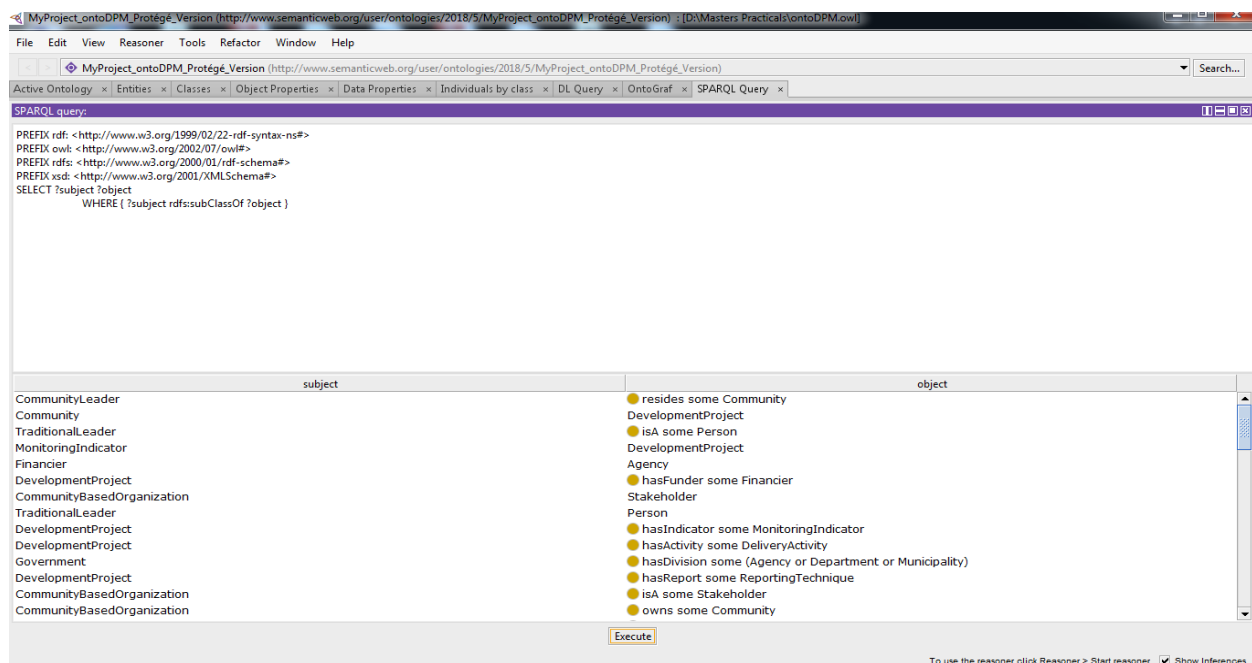
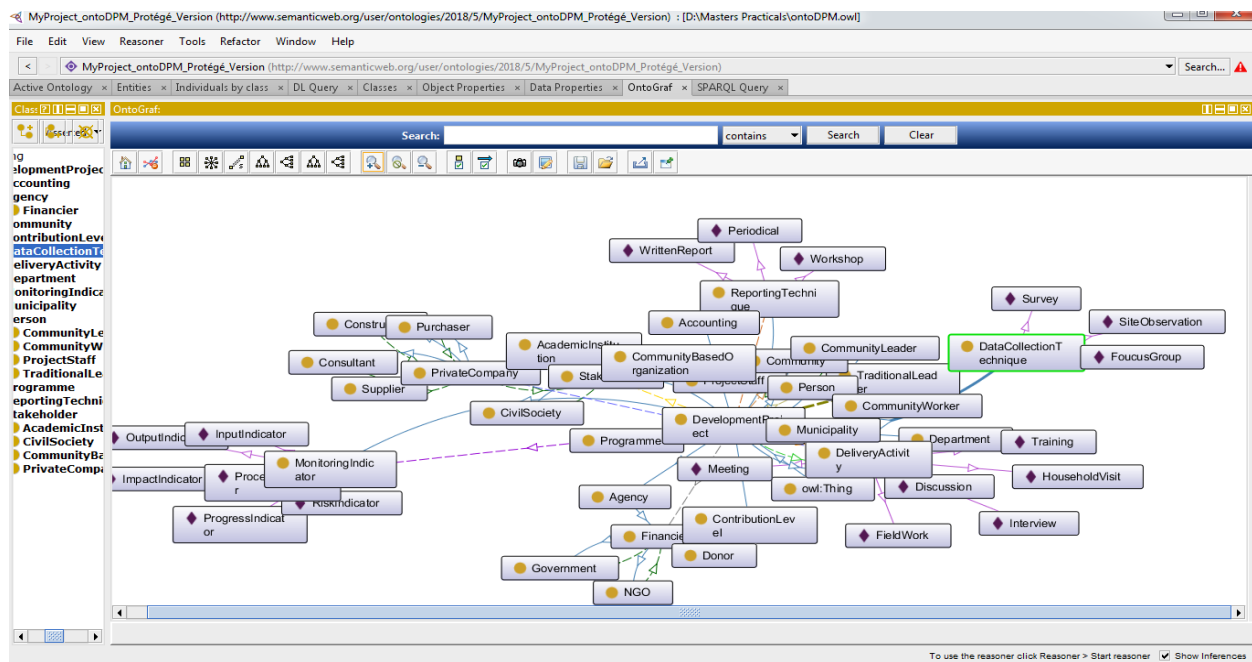
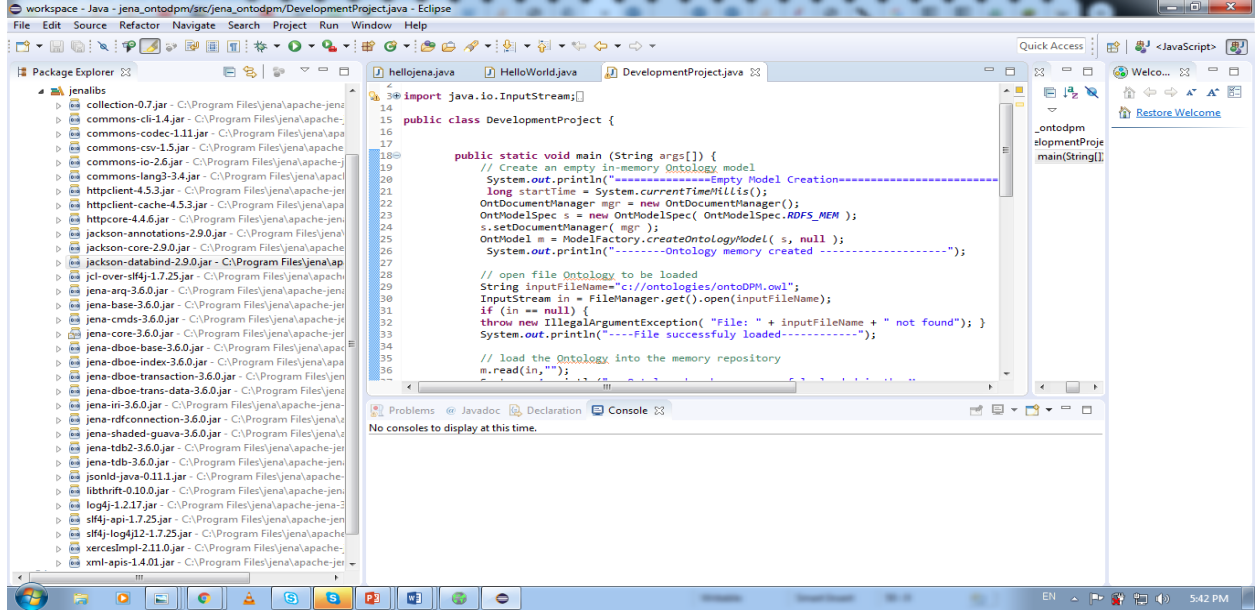


Figure B. 1: Screenshot of OntoDPM in Protégé



The above result shows the output of SPARQL Query executed using Protégé which displays “subject” and “object” and also showing the subclass and superclass relationships in OntoDPM Ontology.



**Figure B. 4:** Screenshot of Jar Files in Jena API

**Table B. 1:** Full Code to Load and Query Ontologies in Jena API

```
package jena_ontodpm;

import java.io.InputStream;
import org.apache.jena.Ontology.OntDocumentManager;
import org.apache.jena.Ontology.OntModel;
import org.apache.jena.Ontology.OntModelSpec;
import org.apache.jena.query.Query;
import org.apache.jena.query.QueryExecution;
import org.apache.jena.query.QueryExecutionFactory;
import org.apache.jena.query.QueryFactory;
import org.apache.jena.query.ResultSetFormatter;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.util.FileManager;

public class DevelopmentProject {

    public static void main (String args[]) {
        // Create an empty model to hold Ontology in-memory
        System.out.println("*****Creation of Empty
model*****");
        long startTime = System.currentTimeMillis();
        OntDocumentManager mgr = new OntDocumentManager();
        OntModelSpec s = new OntModelSpec( OntModelSpec.RDFS_MEM );
        s.setDocumentManager( mgr );
        OntModel m = ModelFactory.createOntologyModel( s, null );
        System.out.println("*****Ontology Model has been successfully
Created*****");

        // open and loaded file Ontology in Jena memory
        String inputFileName="c:/ontologies/ontoDPM.owl";
        InputStream in = FileManager.get().open(inputFileName);
        if (in == null) {
            throw new IllegalArgumentException( "File: " + inputFileName + " not found"); }
        System.out.println("*****File successfully
loaded*****");
    }
}
```

---

```

        // load the Ontology into the memory repository
        m.read(in,"");
        System.out.println("===Ontology has been successfully loaded in the Memory
*****");
        long estimatedTime = System.currentTimeMillis() - startTime;
        System.out.println("=====Estimate Loading Time is:"+ estimatedTime
+ "*****");

```

**Table B.2:** The Output Results of Loading and Querying Ontologies in Jena API

---

```

*****Creation of Empty model*****
*****Ontology Model has been successfully Created*****
*****File successfully loaded*****
===Ontology has been successfully loaded in the Memory *****
=====Estimate Loading Time is:3081*****
=====reading a query: =====
-----
-----
-----
| subject |
| object |
=====
=====
=====
|
| <http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#DevelopmentPro
| ject> | _:b0 |
|
| <http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#DeliveryActivi
| ty> |
| <http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#DevelopmentPro
| ject> |
|
| <http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#ProjectStaff>
| | _:b1 |
| | <http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#Donor>
| | _:b2 |
|
| <http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#CommunityWorke
| r> |
| <http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#Person>
|
|
| <http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#CommunityBased
| Organization> |
| <http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#Stakeholder>
|
|
| <http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#AcademicInstit
| ution> |
| <http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#Stakeholder>
|
|
| <http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#ContributionLe
| vel> |
| <http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#DevelopmentPro
| ject> |

```

---

---

```

|
<http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#DevelopmentPro
ject> | _:b3
|
<http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#PrivateCompany
>
<http://www.semanticweb.org/user/ontologies/2018/5/MyProject_ontoDPM_ProtÃ©gÃ©_Version#Stakeholder>
|
-----
-----
=====Query Response Time is:924=====

/*****
****/

// Create a new SPARQL Query
String queryString =
"PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#> " +
"SELECT ?subject ?object " +
"WHERE {" +
" ?subject rdfs:subClassOf ?object " +
" }" +
" LIMIT 10" ;

System.out.println("=====reading a query:
=====");
    long startTime2 = System.currentTimeMillis();

    Query query = QueryFactory.create(queryString);
    // Execution of the query and Display the results
    QueryExecution qe = QueryExecutionFactory.create(query, m);
    org.apache.jena.query.ResultSet results = qe.execSelect();
    // Output query results
    ResultSetFormatter.out(System.out, results, query);
    // Important -free up resources used running the query
    qe.close();
    long estimatedTime2 = System.currentTimeMillis() - startTime2;
    System.out.println("=====End time Time for querying: Query
Response Time is:"+ estimatedTime2 +"=====
}

```

---

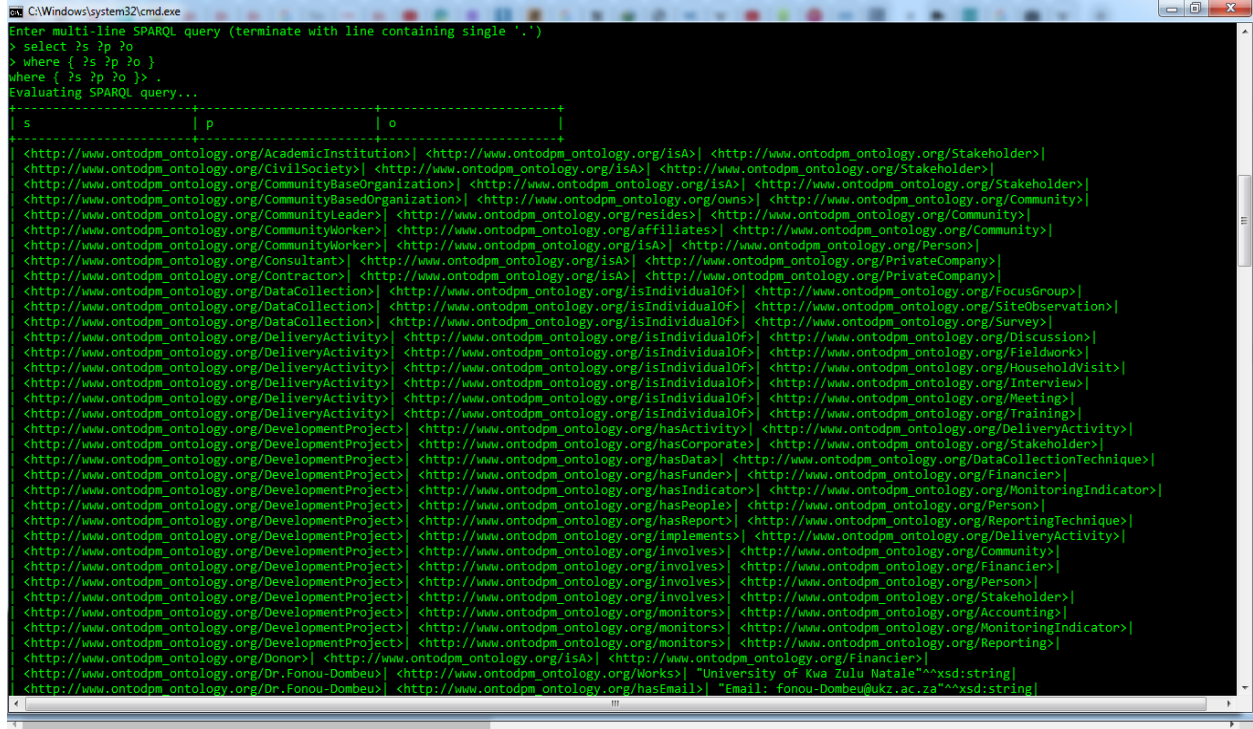


Figure B. 5: Query Result of OntoDPM Ontology in RDF4J

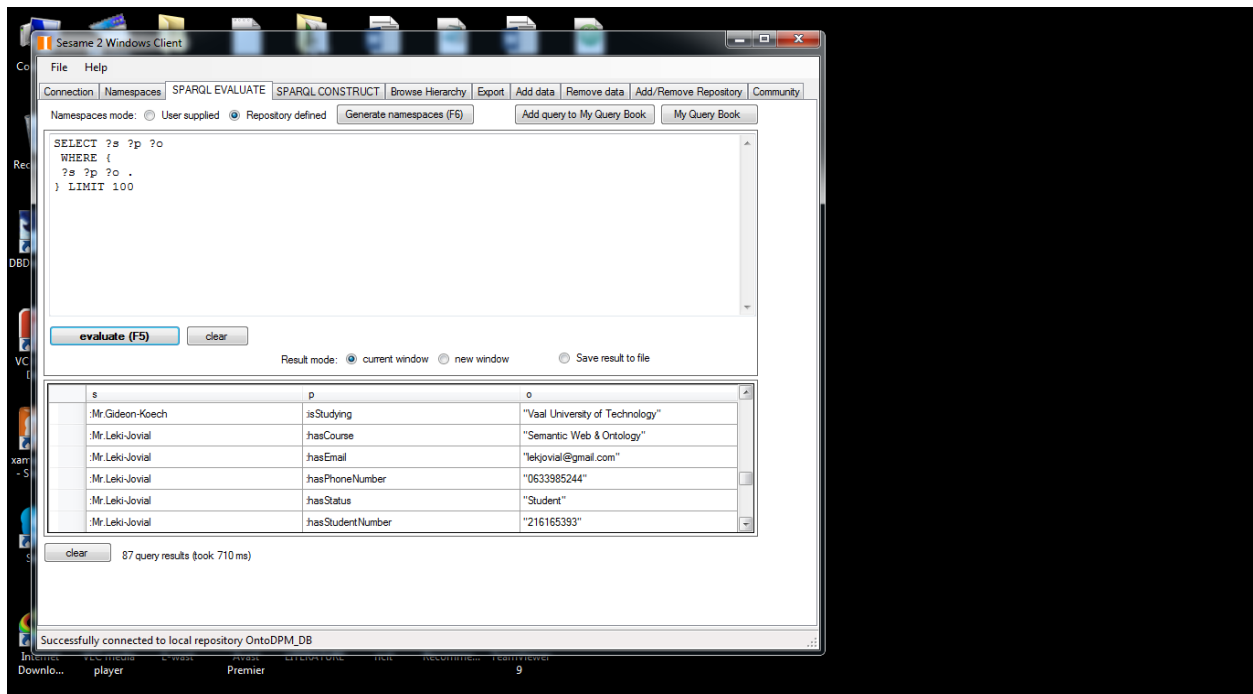
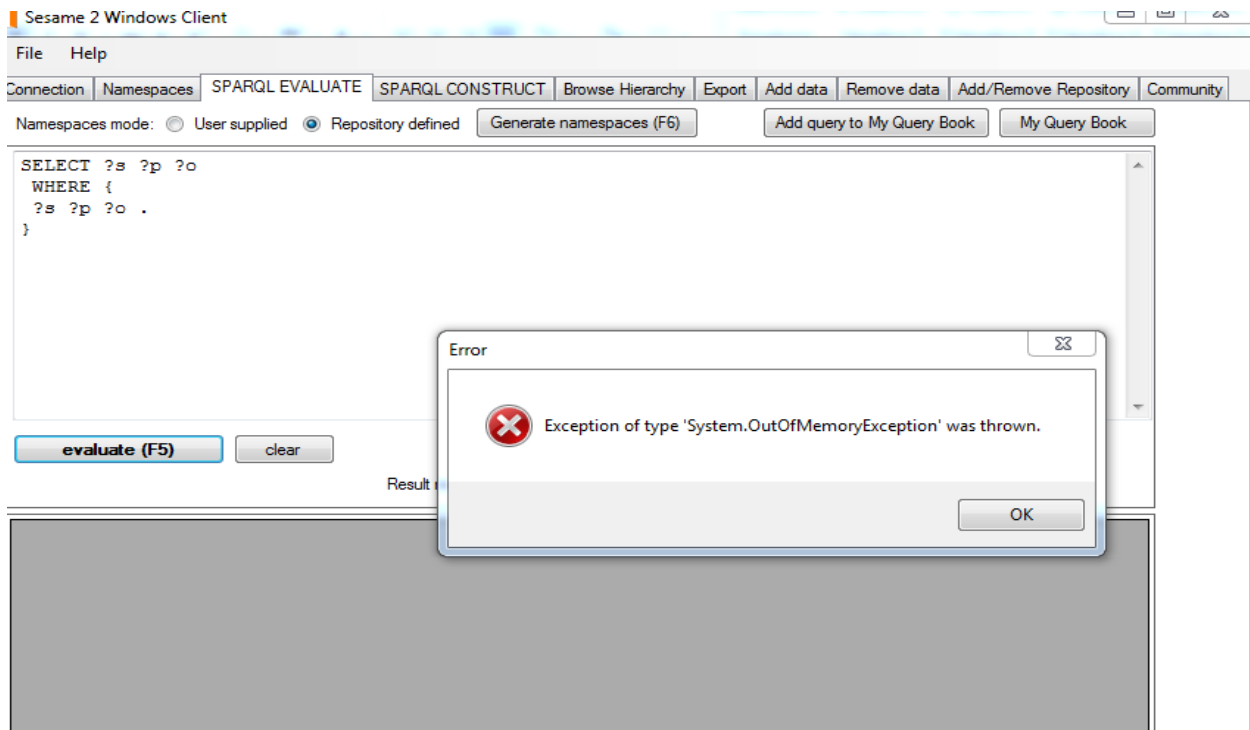


Figure B. 6: Query Result of Individuals of OntoDPM Ontology in RDF4J



**Figure B. 7:** Fatal Error while Querying Gene Ontology in RDF4J (Sesame SDK)



## **APPENDIX C: FULL TABLE ON COMPARISON OF DOT NET AND OPEN SOURCE PLATFORMS FOR ONTOLOGY DEVELOPMENT**

The first column presents the name of Ontology, in the second column named storage indicates the mode used by those platforms to store Ontology files. A third column indicates tools which are open source. The fourth column named extensibility, shows the capability of adding more features supported by platforms. The fifth column provides the architecture type available for each platform while last two columns shows if platforms support the import and export of ontologies from different sources.

Tools	Storage	Availability	Open source	Extensibility	Architecture	Import Language	Export Language
Protégé	In-memory, Files & DBMS	Free	Yes	Yes	Standalone & ClientServer	RDF(S), OWL	RDF(S), OWL, CLIPS
Jena API	In-memory, Files & DBMS	Free	Yes	Yes	Standalone & ClientServer	RDF(S), OWL	RDF(S), OWL
RDF4J	In-memory, Files & DBMS	Free	Yes	Yes	Standalone & ClientServer	RDF(S), OWL	RDF(S), OWL
IsaViz	Files	Free	Yes	Yes	Standalone	RDF(S), N-Triple	RDF(S), N-Triple
OilEd	Files	Free	Yes	No	Standalone	RDF(S), DAML+ OIL	RDF(S), DAML+ OWI
Swoop	Files	Free	Yes	Yes	Standalone	RDF(S), OWL	RDF(S), OWL
OBO-Edit	Files	Free	Yes	Yes	Standalone	OBO File format, OWL	OBO File format, OWL
Hozo	Files	Free	Yes	No	Standalone & ClientServer	RDF(S), subset of OWL	OWL, RDF(S)
OntoBuilder	Files	Free	Yes	Yes	Standalone & ClientServer	RDF(S), OWL WSDL	RDF(S), Microsoft Biz Talk
Ontosaurus	Files	Free	Yes	No	ClientServer	LOOM, IDL KIF, C++	LOOM, IDL KIF, C++
Apollo	Files	Free	Yes	Yes	Standalone	Apollo Meta-language	OCML
Grafoo	Files	Free	Yes	Yes	Standalone	OWL2, Tuttle OWL/XML	OWL2, Tuttle OWL/XML
KAON	DBMS	Free	Yes	No	Standalone	RDF(S)	RDF(S)
pOWL	Files	Free	Yes	Yes	N-tier Architecture	RDF(S), N-triple	RDF(S), N-triple
WSMO Studio	Files	Free	Yes	Yes	Standalone	WSML-XML, RDF(S), OWL	WSML-XML, OWL-D
Neon Toolkit	Files	Free	Yes	Yes	Standalone	RDF(S), OWL	RDF(S), OWL
SemWeb.NET	In-memory, Files & DBMS	Free	Yes	Yes	Standalone	RDF	RDF
LinqToRdf	In-memory, Files & DBMS	Free	Yes	Yes	Standalone	RDF(S), OWL	RDF
dotNetRDF	In-memory, Files & DBMS	Free	Yes	Yes	Standalone	RDF	RDF(S), OWL
RDFSharp	Files	Free	Yes	Yes	Standalone	RDF(S), N-Triple	RDF(S), N-Triple
OwlDotNetApi	Files & DBMS	Free	No	Yes	Standalone	RDF(S), OWL	RDF(S), OWL
dotSesame	In-memory, Files & DBMS	Free	Yes	Yes	Standalone	RDF(S), OWL	RDF(S), OWL
BrightstarDB	Files & DBMS	Commercial	No	Yes	Standalone	RDF(S), OWL, N-Triple	RDF(S), OWL

Rowlex	Files& DBMS	Commercial	No	Yes	Standalone	RDF(S), OWL	RDF(S), OWL
TODE	Files& DBMS	-	No	Yes	Standalone	RDF(S), N-Triple, OWL	RDF(S), N-Triple OWL