

A Semantic Approach to Intelligent and Personal Tutoring System

By
Maria Sette

Submitted in partial fulfillment
Of the requirements for the degree of
Doctor of Professional Studies
In Computing

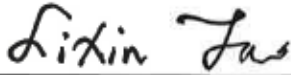
At
School of Computer Science and Information Systems
Pace University

January 2017

Copyright © 2016 Maria Sette

All rights reserved.

We hereby certify that this dissertation, submitted by **Maria Sette**, has successfully satisfied all the dissertation requirements for the degree of *Doctor of Professional Studies in Computing*.



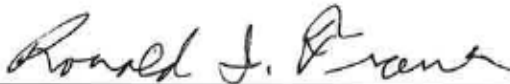
Dr. Lixin Tao
Chairperson of Dissertation Committee

January 25, 2017
Date



Dr. Charles Tappert
Dissertation Committee Member

January 25, 2017
Date



Dr. Ronald Frank
Dissertation Committee Member

January 25, 2017
Date

School of Computer Science and Information Systems

Pace University 2017

Abstract

A Semantic Approach to Intelligent and Personal Tutoring System

By

Maria Sette

Submitted in partial fulfillment
Of the requirements for the degree of
Doctor of Professional Studies
In Computing

January 2017

Cyberlearning presents numerous challenges such as the lack of personal and assessment-driven learning, how students are often puzzled by the lack of instructor guidance and feedback, the huge volume of diverse learning materials, and the inability to zoom in from the general concepts to the more specific ones, or vice versa. Intelligent tutoring systems are needed to improve the Cyberlearning quality. One of the major difficulties is knowledge representation. The current industry standard is to use Web Ontology Language (OWL) for representing knowledge structure. But OWL only supports one "first-class" relation, "is-a", between the concepts, and different knowledge areas usually need different custom relations to describe the relations among the concepts. For example "part-of" and time dependency are important relations to represent most engineering knowledge bodies. OWL is limited to object properties to emulate such custom relations, leading to awkward knowledge representation hard for domain experts to code, validate and use such knowledge bases. This research uses Pace University's extension to OWL, named Knowledge Graph (KG), to support knowledge representation with custom relations. The instructors can use Pace University extended Protégé IDE to declare and apply custom relations in a single document. The instructors teaching experience is also coded in the KG to better support custom learning order by students with different backgrounds. The prototype of a knowledge-driven tutoring system was designed and implemented to illustrate how the KG supports integrated assessments; using assessment results to custom student learning order or material; and let the students freely navigate in the knowledge space from general to specific or the opposite, and following various custom relations. A web technology tutorial is used to validate the design and effectiveness of this approach.

Acknowledgements

This journey has been a very challenging and exhausting path. At times I felt defeated and other times I struggled to finish. I am very thankful to have been surrounded by people who provided the encouragement, council, and strength to see the light at the end of this journey.

This research paper is in honor and loving memory of my father Mr. Michael Sette. He passed away in 1995 never had the opportunity to see my great achievements. Of course my dear mother, Mrs. Agata Sette who has given me the strength, hope, and faith, in believing and guiding me toward opportunities she never had but was able to give me this special gift. Also my beloved and special dog's Morgan Sette who was my best friend and the love of my life who passed away in 2012 during my doctoral program and my little angel Madison Sette a gift from GOD.

I dedicated this life achievement to the people who have been most important and instrumental in guiding my life both by their support and their example. This achievement would not have been made possible without the support of all my relationships with those from Pace University.

First I would especially like to thank my advisor, Dr. Lixin Tao for his wise counsel, guidance, and patience with me through this process. His keen insights and ability to appreciate the complexities and challenges of technology in a professional setting has been a source of inspiration and the encouragement that I needed to see this through. To Dr. Fred Grossman, for having the forethought to craft such a program at Pace University, I am truly thankful for having the opportunity to pursue this milestone in my life.

And finally, I would also like to extend my thanks to Christine Martinez and Ning Jiang for their countless hours of support and mentoring. Both have been my light to a dark tunnel. This program has taught me a lot about who I was from the beginning and who I have become at the end. I am very proud to be part of this establishment.

Table of Contents

Abstract	iv
List of Figures	7
List of Tables	10
Chapter 1 Introduction	11
1.1 Cyberlearning Challenges	11
1.1.1 Unlimited Choices of Learning Materials	14
1.1.2 Lack of Instructor Personal Guidance.....	16
1.2 Knowledge Representation for Adaptive Student Navigation	16
1.2.1 An Intelligent Tutoring System Based on Knowledge Representation	18
1.2.2 Personalized Learning Profiles.....	19
1.2.3 Knowledge Learning Object Navigational Features	22
1.2.4 Generating Learning Sequences.....	23
1.2.5 Assessment Based Learning.....	24
1.2.6 Instructor Teaching Experience Representation	24
1.2.7 Studying with Alternative Materials.....	25
1.3 Problem Statement	26
1.3.1 Research Problem	26
1.3.2 Solution Methodology	27
1.3.3 Main Objective and Contributions	28
1.4 Dissertation Roadmap	30
Chapter 2 Literature Survey of Knowledge Representation Methods and Web-Based Technologies	31

2.1	Forms of Knowledge Representation Methods	31
2.1.1	Semantic Networks.....	31
2.1.2	Frame-Based	36
2.1.3	Rules-Based	42
2.1.4	Logic-Based.....	44
2.1.5	Ontology-Based.....	46
2.1.6	Knowledge Graph	50
2.2	Learning Objects	54
2.3	Web-Based Technologies	55
2.3.1	JAVA.....	55
2.3.2	JavaScript.....	56
2.3.3	JavaServer Pages (JSP)	57
2.3.4	AJAX	58
2.3.5	Java Servlets	59
2.3.6	Tomcat.....	60
2.3.7	Spring MVC.....	67
2.3.8	Apache Jena.....	73
2.3.9	XML Parsers.....	74
Chapter 3	An Intelligent Tutoring System Framework Empowered with	
	Knowledge Representation	77
3.1	Cyberlearning Challenges	77
3.1.1	Overwhelming Study Resources And Lack Of Personalized Guidance	78
3.1.2	Lack Of Instructor Experience	79
3.1.3	Lack Of Assessment Based Study	80
3.2	Intelligent Tutoring System Solution Framework Overview	81

3.2.1	Concepts and Proposed Solution Framework.....	81
3.2.2	Knowledge Representation.....	82
3.2.3	Instructor Experience Representation.....	86
3.2.4	Knowledge Graph Navigation.....	87
3.2.5	Integrated Assessment.....	88
3.2.6	Web Tutorial Ontology.....	88
3.2.7	Evaluation.....	91
3.2.8	Documentation.....	91
3.2.9	Conclusion.....	92
3.3	Knowledge Representation of Intelligent Tutoring System Solution	92
3.3.1	Knowledge Representation.....	92
3.3.2	Capturing Instructor Experience in Learning Orders.....	97
3.3.3	Capturing Instructor's Adaptive Learning Topics.....	99
3.3.4	Capturing Instructor's Integrated Assessment.....	101
3.3.5	Conceptualizaion of Web Tutorial Ontology.....	102
3.4	Tutoring System Life Cycle Based on Knowledge Representation	106
3.5	Tutoring System Architecture.....	107
3.5.1	Functional and System Design Requirements.....	107
3.5.2	System Architecture.....	108
3.6	Tutorial System Use Cases	110
3.6.1	Main Use Case.....	111
3.6.2	Instructors Managing Course Content Use Case.....	115
3.6.3	Student Login Process Use Case.....	118
3.6.4	Login Use Case.....	120
3.6.5	Registration Use Case.....	122

3.6.6	Login Learning Order Use Case	124
3.6.7	Knowledge Space Exploration Use Case	125
3.6.8	Quiz Module Overview	127
3.6.9	Conclusion	131
3.7	Pace Protégé Extension for Knowledge Representation	131
3.8	PaceJena for Knowledge Navigation	136
3.9	Knowledge Graph Representation for Navigation	136
3.10	Summary	137
Chapter 4 Intelligent Tutoring System Implementation Highlights and		
Adaptation to Other Subjects		139
4.1	Introduction to Tutorial Web-Based System	139
4.2	Overview of Tutorial Web-Based Architecture	140
4.3	Open Source Web-Based Technologies	141
4.4	Web Tutorial Knowledge Representation	145
4.5	Web Tutorial File Structure Representation	152
4.6	Web Tutorial Learning Order Specification	155
4.7	Web-Based Technology Implementation.....	159
4.7.1	Tomcat Web Server	160
4.7.2	Tutorial POM Descriptor.....	163
4.7.3	Tutorial Deployment Descriptor	167
4.7.4	Tutorial Web Dispatcher Customization	174
4.7.5	Tutorial Application File Structure.....	180
4.7.6	Tutorial Annotation Web-Based Controllers	182
4.8	Adaptive Tutorial Navigational Features.....	186
4.8.1	Adaptive Navigational Knowledge Space.....	186

4.8.2	Adaptive Navigational Knowledge Space Guidance	187
4.8.3	Adaptive Navigational Knowledge Space Hiding	187
4.8.4	Adaptive Navigational Knowledge Organization	188
4.8.5	Adaptive Navigational Knowledge Assessment.....	189
4.9	Managing Student Learning Profiles	189
4.10	Managing Knowledge Object Collection and Organization.....	191
4.11	Managing Students Learning Topic: Editing, Saving and Update Process ..	193
4.12	Managing Instructors New Subject Topics	195
4.13	Quiz Development and Integration	196
4.13.1	Challenge Quiz Overview	196
4.13.2	Students Knowledge Exploration	198
4.13.3	Instructor's Integrated Assessment	205
4.14	Process to Developing a Tutorial System for a New Subject.....	206
4.14.1	Knowledge Representation.....	206
4.14.2	Learning Order Specification	210
4.14.3	Knowledge Object Collection and Organization.....	212
4.14.4	Managing New Topics for Editing, Saving, and Deletion	214
4.14.5	Navigating the Knowledge Space.....	216
4.14.6	New Quiz Development and Integration.....	217
4.15	Summary.....	217
Chapter 5	Validating System Effectiveness through Prototype Application	
	218	
5.1	Evaluation Process of the Tutorial System Prototype.....	218
5.1.1	Participates	219
5.1.2	Procedures and Instruments	220

5.1.3	Comparing and Contrasting Different Learning Methods	222
5.1.4	Students Feedback.....	223
5.2	Analyzing Quiz Results.....	225
5.3	Conclusion	233
Chapter 6	Conclusion	234
6.1	Major Research Contributions.....	235
6.2	Potential Future Work	236
Appendix A	237
Appendix B	246
Appendix C	260
Appendix D	262
Appendix E	263
Appendix F	265
References	267

List of Figures

Figure 1 PaceJena Tutoring System Architecture	13
Figure 2 Block Diagram of an Adaptive E-Learning System.....	17
Figure 3 Proposed PaceJena Tutoring System Life Cycle	19
Figure 4 PaceJena Tutoring System Login Page	20
Figure 5 PaceJena Tutoring System User Registration	21
Figure 6 PaceJena Tutoring System Learning Path Navigation	23
Figure 7 The “is-a” relationship between class and superclass	32
Figure 8 The “is an instance of” relationship between instance and class	32
Figure 9 The “is a part of” relationship between part and whole	32
Figure 10 The “has” relationship between object and attribute.....	33
Figure 11 Inheritance of semantic network	33
Figure 12 Repeated “is-a” link with different meanings	35
Figure 13 The “type-of” and “subtype-of” links.....	36
Figure 14 Representation of Semantic Network Frame	38
Figure 15 Diagrammatic form of frame-based system.....	39
Figure 16 Semantic Network Edges.....	45
Figure 17 Representation of Individuals	48
Figure 18 Representation of Properties	49
Figure 19 Representation of Classes	49
Figure 20 Example of Triple Representation.....	50
Figure 21 Pet Relations Using “is-a” Predicate	50
Figure 22 Hand Relations Using “partOf” Relation.....	53
Figure 23 Tomcat Architecture with Main Components [37]	61
Figure 24 Front Controller Design Pattern [14].....	69
Figure 25 Spring MVC Design Pattern [14].....	69
Figure 26 Spring MVC Architecture [14]	70
Figure 27 Spring Web Configuration Hierarchy.....	70
Figure 28 Spring MVC Request Life Cycle	73
Figure 29 DOM Parser Workflow.....	74
Figure 30 SAX Parser Workflow	75
Figure 31 Solution Framework of PaceJena Tutoring System	85
Figure 32 Knowledge Graph Representation of Extended and Custom Relations	93
Figure 33 Learning Order for Beginner.....	98
Figure 34 Learning Order for Intermediate	99
Figure 35 Web Tutorial Ontology Scope of Knowledge Representations	103
Figure 36 Pace Protege Version 5.0 - Adding Relation	104
Figure 37 Pace Protege OWLVIZ Knowledge Graph with Custom Relations	105
Figure 38 PaceJena Tutoring System Architecture	108

Figure 40 Screen Capture of the PaceJena Tutoring System.....	111
Figure 41 Example of Knowledge Graph Based on Custom Relation “partOf”	112
Figure 43 Instructor / Teacher Login Use Case	115
Figure 44 Instructor / Teacher Editing Course Content Use Case	116
Figure 45 Instructor / Teacher Saving Course Content Use Case.....	116
Figure 46 A Snapshot of Proposed Web Tutorial Ontology In Pace Protégé	117
Figure 47 Student Login Process Use Case	119
Figure 48 Student Login Process Use Case	120
Figure 49 Interaction Diagram for Login Use Case	121
Figure 50 Login Use Case	121
Figure 50 Interaction Diagram for Registration Process	123
Figure 52 Registration Use Case.....	123
Figure 53 Interaction Diagram for Login Learning Order Process.....	124
Figure 54 Login Learning Order Use Case.....	125
Figure 55 Knowledge Space Exploration Use Case.....	126
Figure 56 Instructor / Teacher Editing Quiz Assessment Use Case	128
Figure 57 Quiz Diagram	129
Figure 58 Student Quiz Assessment Use Case	130
Figure 59 Launching Pace Protege from Command Line	132
Figure 60 Pace Protege GUI Interface with Custom Relations Features	133
Figure 61 Pace Protege GUI Interface with Custom Relations and Related To Feature	134
Figure 62 Pace Protege GUI Interface with Customized OWLViz Knowledge Graph Features	135
Figure 63 Tutoring System Architecture	141
Figure 63 SubClass.....	146
Figure 64 Sample of Individual Classes and Members	147
Figure 30 Sample of Knowledge Graph of Individual Classes and Members	147
Figure 31 Transitive and Inverse Property of “part of” Relations	148
Figure 67 Partial Knowledge Graph Representation of “include and “partOf” Relation	149
Figure 68 Partial Generalization and Specialization Relations.....	150
Figure 32 Partial Knowledge Graph of Generalization and Specialization Relations	150
Figure 33 Represents the IS-A Relation.....	151
Figure 34 Sample Knowledge Representation of Include Relation	151
Figure 72 New Custom Relation ImplementedBy	152
Figure 73 Knowledge Graph of New Custom Relation ImplementedBy.....	152
Figure 35 Declarations of the Standard Prefix Names and Custom Namespaces	154
Figure 75 Knowledge Representation of Customized New Relations	155
Figure 76 Knowledge Representation of Learning Order Structure	157
Figure 77 Defining Learning Order for Beginner	158
Figure 78 Defining Learning Order for Intermediate.....	158
Figure 36 A Sample of Tutoring System Learning Order Diagram.....	159
Figure 80 Tomcat Server Configuration.....	161
Figure 81 Overview POM Descriptor Configuration.....	163
Figure 82 Sample of POM Dependencies	164
Figure 83 Sample of POM.XML File	166
Figure 84 Tutoring System Deployment Descriptor: Web.XML File	169

Figure 85 Servlet and Servlet Mapping	170
Figure 86 SpringDispatcher – servlet.xml File	172
Figure 87 Context Parameter	173
Figure 88 Spring-Database.xml File	173
Figure 90 Tutoring System DispatcherServlet Diagram	175
Figure 92 Tutoring System Application File Structure	181
Figure 93 User Controller	182
Figure 94 Home Controller	183
Figure 95 Learning Topic Controller	184
Figure 96 Owl Navigator Controller	184
Figure 97 MYSQL USER Database A	189
Figure 98 MYSQL USERS Database B	190
Figure 99 Sample of Customized Master Web page Template	192
Figure 100 Summernote Editor Tool [44]	194
Figure 101 Tutoring System File Structure in Ubuntu	195
Figure 102 Quiz Question Database	197
Figure 103 Quiz Scores	198
Figure 104 PaceJenaAuthenticationSuccessHandler.Java	199
Figure 105 SecurityConfiguration.java	200
Figure 106 A Sample of User Flow Diagram	200
Figure 107 Capturing Generalization in PaceJena Tutoring System	204
Figure 108 Capturing Specialization in PaceJena Tutoring System	204
Figure 109 Resetting the Running Environment	206
Figure 110 Defining Classes	207
Figure 111 Create a New Relation “partOf”	207
Figure 112 Defining Relations between Classes	208
Figure 113 Declaring Relations between Concepts	208
Figure 114 Declare Hand is partOf Body	209
Figure 115 Visualization of the OWL	209
Figure 116 Defining Learning Order of Individuals	211
Figure 117 A Sample PaceJena Code for Learning Order	211
Figure 118 Implement New Ontology File	213
Figure 119 Replacing the webTutorial.owl in PaceJenaAuthenticationSuccessHandler.java with New Subject File	213
Figure 120 Administrator Login Access	214
Figure 121 Click on Editing Mode	214
Figure 122 Managing Editing Mode Features	215
Figure 123 Learning Topic Database Table	216
Figure 124 Load XML file into MYSQL Database	217
Figure 125 Analyzing and Comparing Pre-Quiz and Paper Quiz Results Chart	228
Figure 126 Analyzing and Comparing Pre-Quiz and Web-Based Quiz Results Chart	229
Figure 127 Analyzing and Comparing Paper and Web Based Quiz Results Chart	230
Figure 128 Analyzing and Comparing Beginner Quiz Results Chart	231
Figure 129 Analyzing and Comparing Intermediate Quiz Result Chart	232

List of Tables

Table 1 Frame Example of the Book “Semantic Web”	40
Table 2 Frame Example of Personal Data	40
Table 3 Frame Example of “Dell Optiplex”	41
Table 4 Tomcat Default Directory Structure	63
Table 5 Web Tutorial Ontology Key Terms	102
Table 6 Example of Key Terms	112
Table 7 Open Source Web Technology.....	141
Table 8 Knowledge Representation of Learning Order	156
Table 9 Tomcat Webapp Tutoring System Directory Structure	162
Table 10 Elements of POM.XML File Structure	165
Table 11 Deployment Descriptor Elements.....	167
Table 12 Tutoring System Elements	175
Table 13 USERS Database Table Information	190
Table 14 Analyzing and Comparing Pre-Quiz and Paper Quiz Results.....	225
Table 15 Analyzing and Comparing Pre-Quiz and Web-Based Quiz Results	226
Table 16 Analyzing and Comparing Paper Quiz and Web-Based Quiz Results	227

Chapter 1 Introduction

1.1 Cyberlearning Challenges

Today Cyberlearning education has become a very acceptable way to learning course content material, but students are faced with many challenges and are overwhelmed by the choices. Some of the challenges that students encounter is the accessibility and availability of many different types of rich learning object formats, which can be very distracting to a student's learning path and can affect students learning experience and performance. Another challenge to Cyberlearning education is we don't have teacher's present to coach students in learning subject materials. So how can students achieve personalized adaptive learning experience in a Cyberlearning educational environment? Personalized learning services are a key point in the field of online learning as there is no fixed learning path, which is appropriate for all learners. However, traditional learning systems ignore these service requirements and deliver the same learning content to all learners and may not be effective for learners with different backgrounds and abilities. Finally, learning object contains many knowledge concepts, and these concepts are connected through different kinds of relations. Even though people have conducted over a decade of learning objects research to use web base learning resources in learning, they have not looked into knowledge concept reuse. In the last 20 years there has been studies

conducted on learning object's, which is heavily supported by the National Science Foundation. But this study was limited by categorizing, reusing existing learning resource's, they don't have knowledge mapped to integrate these learning objects together for customized adaptive learning environment. Over the past few years there has been an increase in the awareness of a semantically adaptive web-based learning environment [2][17][27][10]. This has been driven by the idea of individualizing tutoring systems for personalized learning. Factors that contribute in this direction include the diversity in the target population participating in learning activities, the diversity in the access media and modalities that one can effectively utilize today in order to access, manipulate or collaborate on educational content or learning activities, alongside with a diversity in context of use of technologies [23]. Some investigations focus on how to improve the organization of representing structure data vocabulary and content for Learning Objects (LOb) [9]. However, this goal cannot be achieved at a massive scale using the traditional approaches. Currently LOb remain an ill-defined concept, which refers to a collection of knowledge points that are combined for being aware of a certain objective [31]. They do not semantically represent knowledge structure relations which links one LOb concept to another LOb concept for web based intelligent tutoring systems, they only do indexing of learning resources. Although there are many tools and various approaches for developing LObs concepts, the objective is to adapt to the learner's needs, assess the learner's performance outcome and to provide feedback recommendations and to adjust learning path and learning order accordingly. Moreover, a web based intelligent tutoring system is a computerized instructional designed application that offers online learning material resources or feedback assessment to learners. Web

based intelligent tutoring systems do not offer automatic feedback recommendation for students on which LOBs to recommend, what learning path and learning order learner should follow in order to better understand learning concepts. The failure to recognize the online learners needs will result in a failure on how instructors use Lobs to improve learning needs.

This research addresses this problem and proposes a semantic approach, PaceJena Tutoring System, for achieving intelligent and adaptive learning, which uses Web LOB as an approach to reuse learning resources. Figure 1 illustrates the architecture of the PaceJena Tutoring System. Four tiers are included in the system, which involve client, education, semantic, and data tiers.

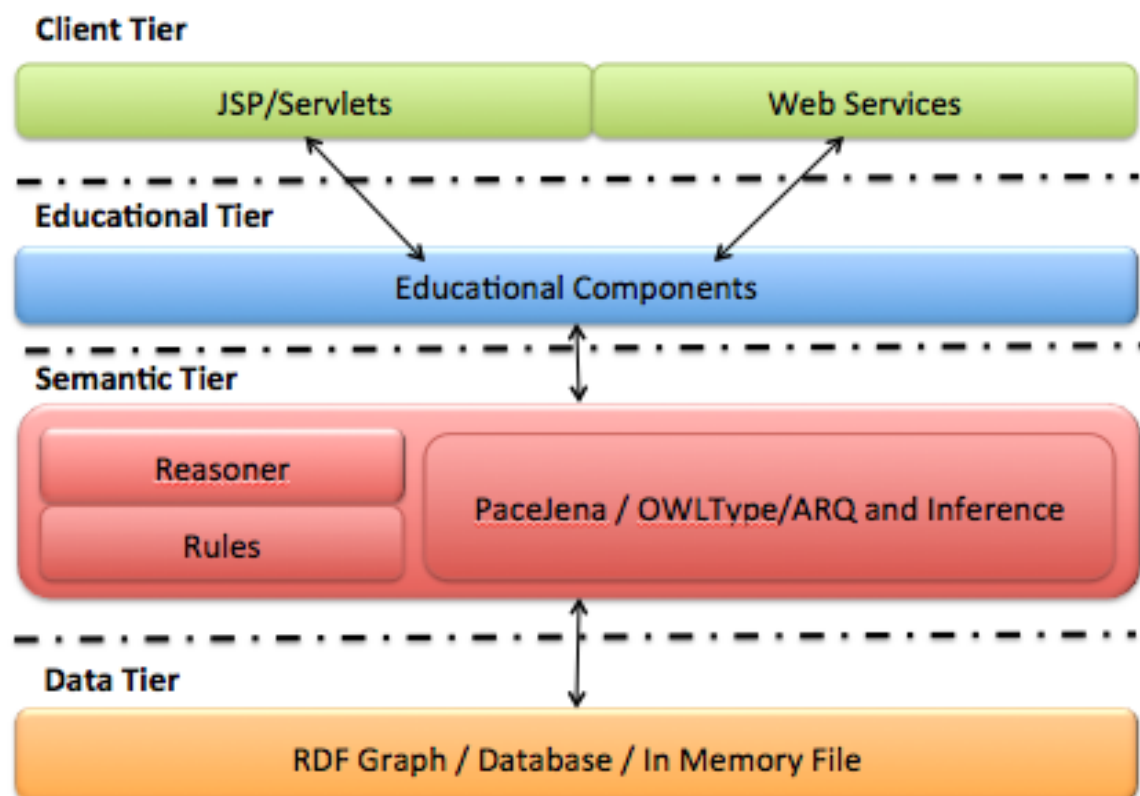


Figure 1 PaceJena Tutoring System Architecture

This research seeks to purpose and improve a methodology and framework to support the representational knowledge structures of LOB resources by means of ontology. Ontology is a formal naming and definition of the types, properties, and inter-relationships of the entities that exist for a particular domain of discourse. But this research goes beyond the “is a” and inheritance relations and also supports the “partOf” relations along with customized relations and to provide the learner with recommended feedback. The recommended feedback consists of evaluating learner’s outcome and semantically mapping learner to learning path using recommended LOB resources. The main contributions of this work include: 1) we use Pace Knowledge Graph to represent knowledge in a subject in order to achieve a customized knowledge structure and enable an intelligent tutoring system. 2) We design and implement a personal tutorial system to support personalized learning.

1.1.1 Unlimited Choices of Learning Materials

In today’s e-learning environment, students are overwhelmed with the amount of learning materials that are available. Student are undecided which learning material is right for them and which learning material they should learn first. Some students learn better from videos, some from content, and others from audio. So many choices to choose from, which is the right choice for me? As technology is evolving, many web-based educational systems have been developed that offer an abundance of learning resources presented in the form of a digital learning objects, such as document files, web pages, audio, video, image files, and much more. Finding the relevant learning resources requires that these

resources be annotated with rich, standardized, and widely used metadata. Several educational metadata standards, such as IEEE LOM (IEEE, 2002) [38], Dublin Core (DCMI, 1990) [37], IMS Consortium (IMS, 2001) [39], and SCORM (ADL, 2004) [43], have emerged to define a standard specification of a learning resource component as a learning object to facilitate accessibility, interoperability, and reusability, the use of different types of metadata has introduced the problem of interoperability across heterogeneous learning resource systems. One of the problems lies on the lack of semantic common standards in the e-learning domain. The abundance of e-learning metadata standards and user-defined metadata does not purpose a globally agreed common standard and still lacks formal semantics. The semantic web provides organizations the ability to annotate learning resources to facilitate knowledge sharing and reuse by others. As a number of learning objects increases, it is expected that many different ontologies can use different e-learning metadata to describe the same or similar sets of learning objects. The related learning objects dispersed over heterogeneous learning resource systems are still difficult to discover and reuse within other courseware, because of the lack of semantic common standards between learning resources ontologies. To access learning objects users need to know the physical locations of the desired learning objects. This leads to the problems of interoperability of heterogeneous learning resource systems. Unfortunately, the various repositories are either closed systems or systems that allow user access only through proprietary interfaces and data formats.

1.1.2 Lack of Instructor Personal Guidance

Traditionally, the lecture based teaching as played an active role and the students a passive one, this has been the main teaching vehicle in many colleges and university systems, military and business training courses. Emerging are the online system methodology approaches, lectures classes are losing their central role while student work is gaining weight, with hours increasingly devoted to online learning. One objectives of the online learning is to transfer knowledge from teachers to learners, and this knowledge has to be interpreted, to be clear and to be retained by learners. Many e-learning environments are challenged with the delivery of how to present learning materials to their students. E-learning environments in colleges and universities tend to teach learning materials in a restricted order for all learners regardless of students learning level. These learning materials also known as learning objects have no intelligences in these types of systems to be able to guide users learning abilities or personalized their learning materials. The assessment follows each learning lesson, which is done by taking a quiz, but the quiz can't evaluate or recommend whether you understood the lesson or whether you should be redirected back to learning topic before go ahead to learn the next topic. Some learners are beginners while others are more advanced. How do we address some of these issues?

1.2 Knowledge Representation for Adaptive Student Navigation

Many successful adaptive e-learning systems do not employ ontologies for knowledge representation. An adaptive student navigation support system is needed to guide students

and generate available learning resource objects, and assist in assessment base learning knowledge to track and compute student's performance level and lead students to a recommended learning path. In e-Learning applications metadata is also fundamental for describing online learning materials and among other information, it can capture the subjects or knowledge domains associated with each document. This information is needed for customizing learning sessions, but it is not sufficient. For locating relevant content, the system needs to know how the subjects are interrelated in an abstract knowledge space and follow the associations to suggest links to the student.

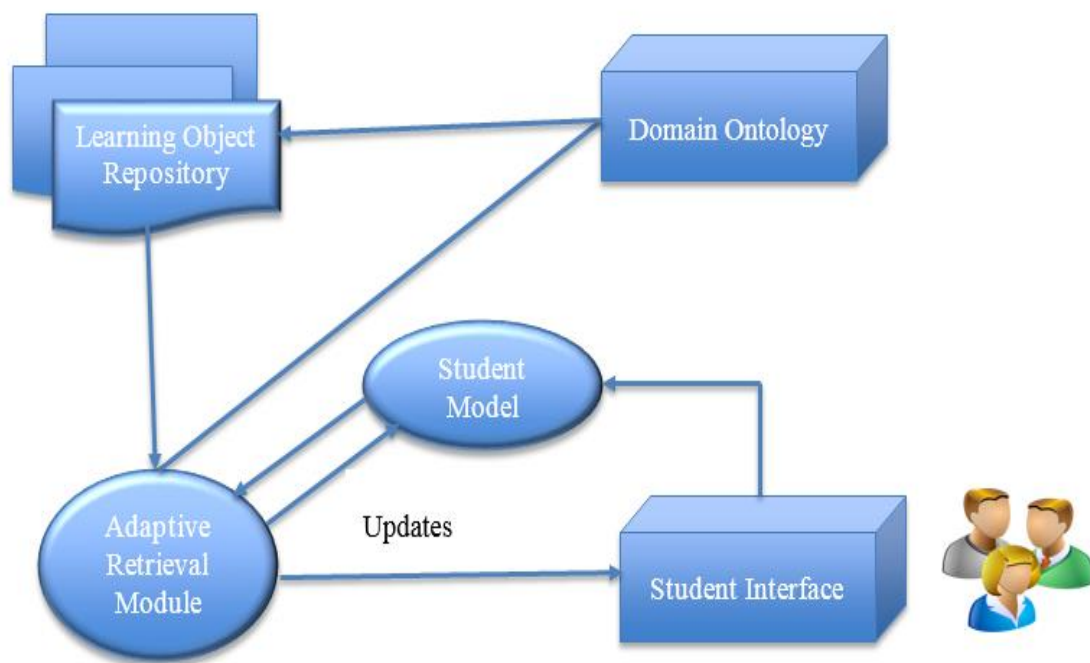


Figure 2 Block Diagram of an Adaptive E-Learning System

The block diagram of a typical adaptive e-learning system is shown in Figure 2. The main modules of an adaptive system are the domain ontology, student model, adaptive retrieval module and the learning object repository. For the development of an adaptive system, the domain ontology plays a crucial role [28], [30]. The ontological structures can be

used for organizing, processing and visualizing subject domain knowledge, marking the topic and coverage of learning objects, and for building learner models in e-learning systems. The domain ontology can be used for concept-based domain specific information retrieval, visualization, and navigation that help learners to get oriented within a subject domain and recommend learning path for their own understanding and conceptual association [3], [18].

1.2.1 An Intelligent Tutoring System Based on Knowledge Representation

In this section we will describe the desired features that support our intelligent tutoring system, which is based on our semantic approach to personalized learning profiles, knowledge navigation, generating learning object sequences, assessment based learning, teaching experience representation, and studying with alternative materials. In Figure 3, Proposed PaceJena Tutoring System Life Cycle illustrates the different phases for this system. The student content is where the student initiates his/her personalized learning level. Once learning level has been established the tutoring system will assign the student with the learning order. Student will have the ability to navigate the learning objects from the knowledge structure for this learning content. The instructor content contains the resources needed for the instructor to manage student's performance and learning outcomes. The semantic technology content contains the Web Tutorial Ontology, which is a knowledge graph and knowledge structure, which gets read into the PaceJena program. The PaceJena program is where the functions and methods drive the knowledge graph. PaceJena also contains the Sax Parser, which will parser the knowledge data. The assessment content is where student will be challenged with a quiz after each topic has

been completed. This quiz will evaluate and deliver feedback to students on how they should proceed with their learning strategy.

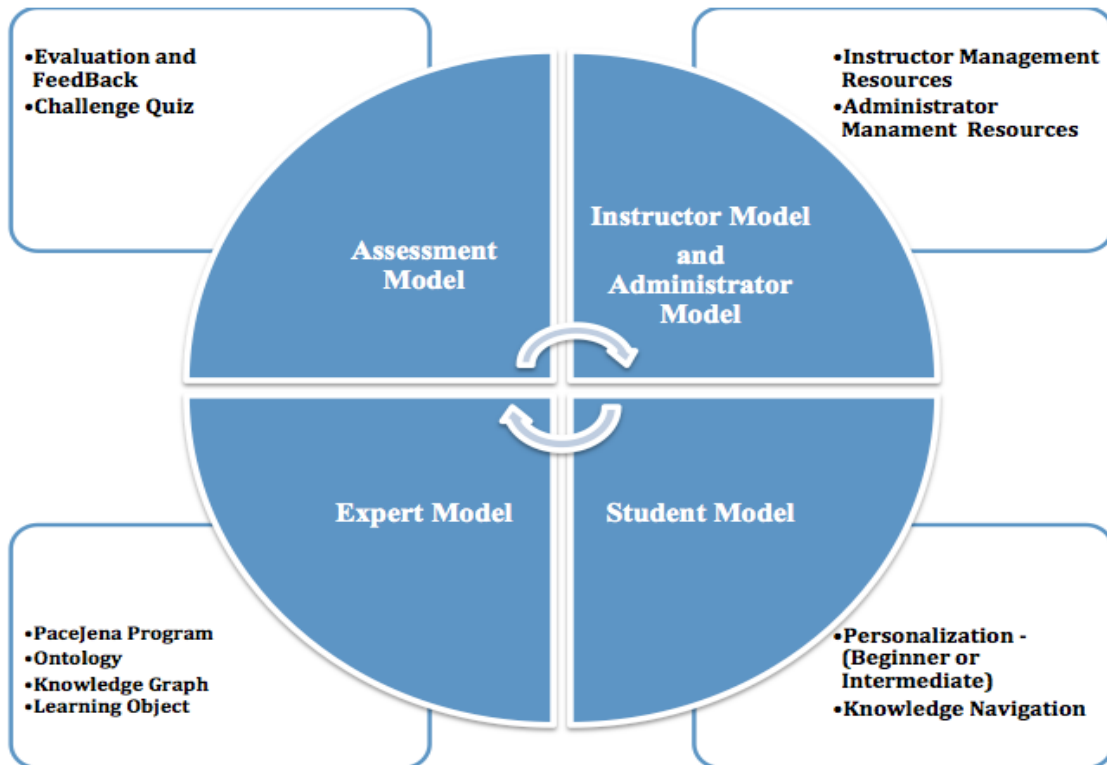


Figure 3 Proposed PaceJena Tutoring System Life Cycle

1.2.2 Personalized Learning Profiles

Research as argued [36] that tutoring systems should provide a personalized learning environment. In this subsection, we show how PaceJena Tutoring System adapts to the individual users profile. PaceJena Tutoring System contains two learning levels of learning for the purpose of this dissertation we have chosen a “Beginner” level and an “Intermediate” level. Personalized student profile learning levels are pre-defined during the new user account registration form. Students who wish to use the PaceJena Tutoring System must register on the PaceJena Tutoring System user registration page. The

student can create his/her registration in Figure 4 by first going to the user login screen.

On this page the student will see a sentence and link called “Not Yet Registered!!! Register Here”. This link will bring the student to the registration form page.



The image shows a screenshot of the PaceJena Tutoring System login page. At the top left is the Pace University logo with the tagline "Work toward greatness." To the right of the logo is a graphic of a human head in profile, composed of a wireframe mesh, set against a background of blue and white data points. Below the header is a dark blue bar with the text "PACEJENA TUTORING SYSTEM" in white. The main content area is white and contains a light gray rounded rectangle with a login form. The form has two input fields: "User ID" and "Password". Below these fields is a blue "Sign in" button. At the bottom of the form is a link that says "Create account".

Figure 4 PaceJena Tutoring System Login Page

The screenshot shows the user registration interface for the PaceJena Tutoring System. At the top left is the Pace University logo with the tagline "Work toward greatness." To the right is a graphic of a human head with a digital grid overlay. Below this is the title "PACEJENA TUTORING SYSTEM". The registration form is centered and contains the following fields:

- User ID:
- Email:
- First Name:
- Last Name:
- Password:
- Confirm Password:
- Learning Level:

A blue "Register" button is located at the bottom of the form.

Figure 5 PaceJena Tutoring System User Registration

The registration page in Figure 5 contains the following registration information; user must enter a “Student ID,” “Last Name,” First Name,” Preferred UserID,” “Email Address,” “Password,” “Password (Repeated),” “Select Learning Level:,” (We are only using Beginner and Intermediate for this dissertation.) student must use the down arrow to select the learning level of their choice and finally the student will click on the submit button. The system will register this information and generated the student’s profile in the PaceJena Tutoring System database. Students with learning level “Beginner” will learn the learning material in sequential order from the very beginning. Students with intermediate will learn in a different learning order then beginner.

1.2.3 Knowledge Learning Object Navigational Features

The knowledge learning object navigational in figure 6 features are activated when the user creates his/her registration account. When user logs into the PaceJena Tutoring System for the first time, the knowledge learning objects navigational menu will appear on the left pane side, it will be based on the users personalized registration profile for learning level and learning order. This knowledge learning objects navigational menu is used to navigate through the learning object concepts under consideration during the learning process. The learning order for “Beginner” is the recommended learning order for this level of learning. Students have the ability to learn the learning objects in any sequence but it is recommended that they learn according to the learning role for “Beginner”. The same concept goes for also the “Intermediate” learning role. The knowledge navigation learning object concepts will adapt to the learner’s learning assessment performance.

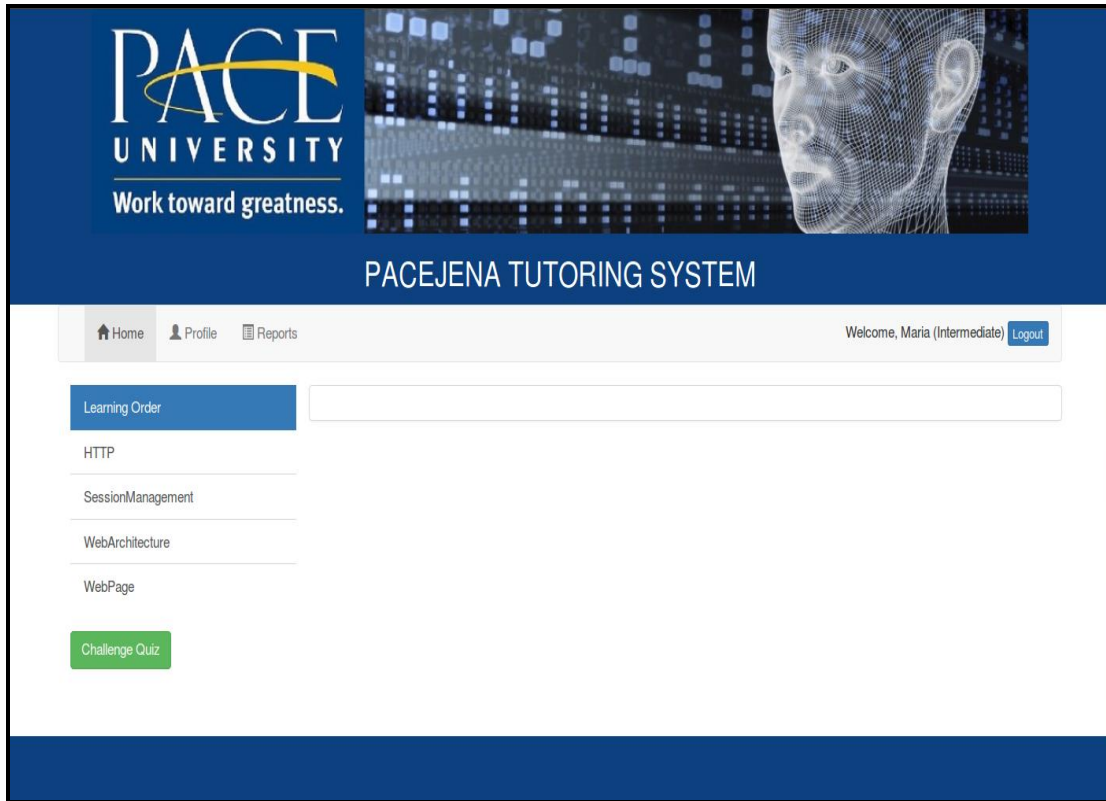


Figure 6 PaceJena Tutoring System Learning Path Navigation

1.2.4 Generating Learning Sequences

A student may want to learn a particular topic without learning every single topic previously mention. A student who wants to summarize a learning object topic may want to learn the minimum set of concepts in order to understand the chosen concept, where as a student whom is not familiar with the topic may want to learn the maximum set of concepts in order to understand the chosen concept in more detail. PaceJena will display a learning sequence for the chosen concept topic.

1.2.5 Assessment Based Learning

The assessment based learning process is a multiple choice challenge quiz. All knowledge based learning objects contain a challenge quiz. The challenge quiz can be taken at any point in the learning process. The challenge quiz will determine and recommend what the student should learn next or the student may need to re-learn the same topic depending on the student's assessment performance outcome. This assessment will determine at what learning level to adjust the students learning role.

1.2.6 Instructor Teaching Experience Representation

In PaceJena Tutoring System both the PaceJena program and the Web Tutorial Ontology provide the instructor's teaching methods. Both components act as a virtual tutoring instructor, presenting the contents in an appropriate sequence, based on the student's knowledge and his/her learning style. This is an interactive process and these components have the ability to assess the knowledge state of the learning process. The student module is used as input to this component, so the pedagogical decisions reflect the differing needs of each student. One of the motivations for building the PaceJena Tutoring System for learning was to create a more effective representational teaching and student strategy. Many teaching strategies are the same for all learners regardless of learning level. This causes an issue for educational institutions because the learning materials are not in sync with the overall learning strategies for different learning styles. This can cause students to be inconsistent with their learning goals. Instructors tend to design learning materials using parameters of their own such as

learning order, learning concepts, and learning assessment. The semantic technology representation and design of the ontology knowledge structure for this research permits the student to decide on how the system should initiate his/her personalized learning from the registration form when creating a user account. Personalization presents the choice of most appropriate learning pattern or resource that will be recommended to the learner. This decision determines what concepts and resources the system is going to present to the learner. In our PaceJena Tutoring System the student is asked to select their level of learning. For this dissertation we decided to keep it simple and just present two levels of learning patterns. One is “Beginner” and the other is “Intermediate”. The PaceJena program and the Web Tutorial Ontology knowledge structure ensure the uniformity among different views in the domain. The order in which the teacher presents the material is up to him/her, the basic knowledge graph structure is not violated.

1.2.7 Studying with Alternative Materials

The traditional way of studying has been from educational system such as attend classes and teachers lectures. This would require buying books, lab materials, and other school accessories like paper, pencils, and much more. But as technology grew and advanced this has changed the dynamics of how we gain knowledge. Today learning is very agile. Agile allows us to move quickly, adapt and reassess our learning consumption. We can now conduct research and access information remotely using the World Wide Web, or the new terminology call cloud computing. The alternative to this type of learning provides use the ability to gain access to other resources and study materials which otherwise would not be available or limited. Cyberlearning gives us the opportunity to

explore this way of learning. In our research we demonstrate how learning objects can be an alternative way to study learning materials.

1.3 Problem Statement

1.3.1 Research Problem

This dissertation paper proposes to explore some of the key challenges that stand in the way of adopting a semantic approach to intelligent and adaptive learning based on web learning objects for traditional colleges and universities. Such systems rely on some of these key challenges that have been mentioned in previous sections, such as;

- 1) **First Major Problem** – Learning objects for online learning resources provide insufficient knowledge structure representation and relations needed to support personalize learning and knowledge space assessment.
- 2) **Second Major Problem** – Learning objects have not been used to guide students to online resources by automated recommendation because they currently don't represent knowledge structure in web technology content system, and only do indexing of resources.
- 3) **Third Major Problem** – Learning objects are not interchangeable, because the objects are not of consistent sizes, nor are they written in consistent languages, they are not really interchangeable. Learning objects are interactive, text, html, flash, Java, and Java-Script. While others learning objects are in audio, multimedia, PowerPoint, and movies. One way to resolve these issues is to link the learning objects. There are problems associated with this strategy, where links

- go down, students don't have the right plugins or drivers, or the learning material supports only one fit's all strategy.
- 4) **Fourth Major Problem** – OWL syntax does not support declaration and accurate semantic inference engines for usage of new customized relations.
 - 5) **Fifth Major Problem** – Ontology does not support any built-in primitives to support part-whole relations.
 - 6) **Sixth Major Problem** - Tree structure graphs are very limited in what it can represent. Tree structures allow no multiple inheritances.
 - 7) **Seventh Major Problem** – The lack of a semantically driven tutoring system to improve the dynamic construction of learning objects to update a student model.
 - 8) **Eighth Major Problem** – The lack of a semantically driven automatic recommender to directly guide students to the next learning path.

1.3.2 Solution Methodology

This research focuses on the following solution methodology to the previous sections, which describe the challenges and the problems with Cyberlearning education. We would like to improve Cyberlearning by implementing the following:

- Developing an ontological knowledge structure graph representation allows me to model complex relations for learning objects and personalized learning knowledge graphs concepts.

- Extending web language ontology to support additional customized knowledge structure relations and to assess and recommend learning order for students at different learning levels and learning direction.
- Allow students to navigate the knowledge graph objects and knowledge space structure using learning object concepts.
- Evaluate and validating this proposed methodology using prototype tutorial system.
- Evaluate and validating the research using an effective comparative analysis of student in a classroom environment.

1.3.3 Main Objective and Contributions

The main objective to this research is to achieve customized and personalized educational knowledge structure and to improve assessment base learning so that students can take advantage of enormous amount of resources to help them navigate in the knowledge space in order to create a more effective learning experience. This dissertation makes contributions in the area of developing, extending and analyzing knowledge learning structure relations and how it effects the commitments that aggregation, generalization and specialization impose on learning web-based objects in a personalized semantic web based environment and the effects of adaptive learning to support predefine learning paths for typical learners to improve learning order and to support adaptive learning through embedded assessments and web-based learning objects. This research also gives contributions to the following:

- Developed and design the knowledge structure representation schema domain for course in “Introductory of Web Technology”, extend this schema and use new customized relations.
- Extending the knowledge structure representation schema domain for web based learning objects in PaceJena Framework to represent new customized knowledge structure and relations.
- Creating the knowledge structure graph representation and learning objects of Web Technology to be retrievable, interpretable, sharable, and re-usable and to have granularity.
- Creating an environment to allowing different learning strategies of a tutoring system environment, using the Web Technology course as a guideline, with different learning paths.
- Students have the freedom to navigate knowledge structure and select the learning objects of their choice.

A prototype tutoring system called PaceJena has been developed to provide students the ability to personalize their learning path according to their abilities during the registration process. PaceJena Tutoring System provides the student with a personalized learning path and students are evaluated after taking the challenge quiz which then the tutoring system will recommend the a new personalized learning order in which student are to learn. Another contribution is that the PaceJena Tutoring System also has enhanced the ease of adaptive navigation at which a student is guided

by the knowledge structure and is able to explore the knowledge space of next level knowledge or previous knowledge.

1.4 Dissertation Roadmap

This research paper is structured as follows, Chapter 2 discusses the literature survey of web development technologies and methods, learning objects, describing the advantages of Java web application technology, and finally comparing and examining the best solution for our tutoring system application. Chapter 3 discusses the objectives of why we need to improve student's online learning experiences, logical design and architecture and knowledge representation prototype design, demo and validation process. This chapter will also discuss predefined learning paths for typical learners, assess the outcomes, and illustrate major use cases and a guideline for adopting this tutoring system for other topics. Chapter 4 discusses the tutoring framework and its implementation such as the Tomcat architecture, how student profiles are managed, details and specifications of the integrate assessment, how student knowledge is captured using generalization and specialization, and finally the process to develop a new subject for the tutoring system. Chapter 5 discusses the validation process analysis, effectiveness and adaptability by introducing the predefined learning path in the intelligent tutoring system to a classroom environment. Chapter 6 discusses the major research contribution, and future works.

Chapter 2 Literature Survey of Knowledge Representation Methods and Web-Based Technologies

2.1 Forms of Knowledge Representation Methods

2.1.1 Semantic Networks

Semantic network is a knowledge representation model, which is in a form of graphical schemes consisting of nodes and links among nodes. Semantic networks of computer executions have been first developed with regard to artificial intelligence and machine interpretation. Nodes in a semantic network can show concepts, objects, features, events, time, and also links indicating the connection among nodes. The links should be labeled and directed. As a result, semantic net refers to a directed diagram. In the graphical perspective, circles or boxes usually represent nodes, and the links are sketched as arrows or connectors among the boxes or circles. The network design indicates its meaning, based on which nodes are related to other nodes. Semantic network as a collection of binary relations with a collection of nodes; the system refers with a predicate logic with binary associations. Semantic systems are simply redundancy-free, because they are not able to allow the duplication from the same node. There are many types of relationships that can be used in semantic networks. The four types that are used are:

The “is-a” relationship between class and superclass (Figure 7);

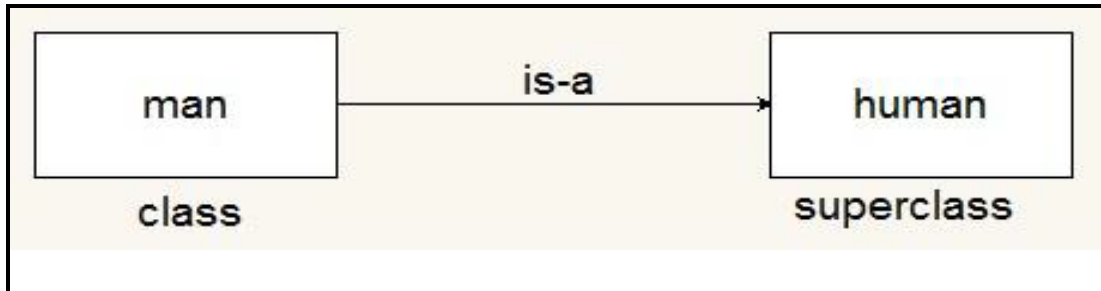


Figure 7 The “is-a” relationship between class and superclass

1. The “is an instance of” relationship between instance and class (Figure 8);

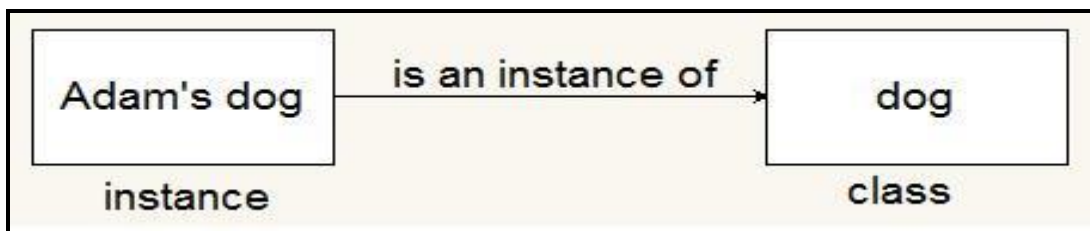


Figure 8 The “is an instance of” relationship between instance and class

2. The “is a part of” relationship between part and whole (Figure 9);

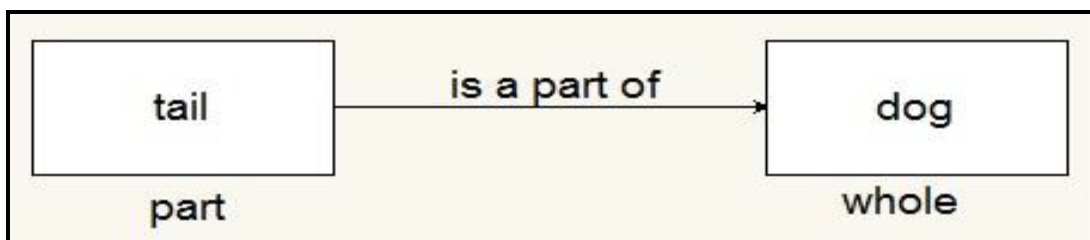


Figure 9 The “is a part of” relationship between part and whole

3. The “has” relationship between object and attribute (Figure 10).

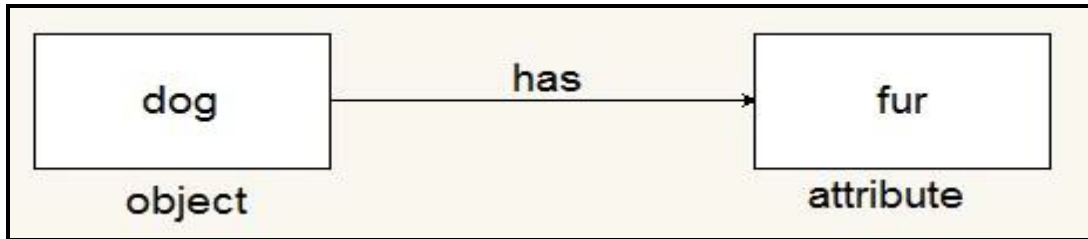


Figure 10 The “has” relationship between object and attribute

The inheritance is the interface of semantic network or is a procedure in which the local knowledge of a node superclass is referred by class node, instance node, and superclass node. In Figure 11 an example about inheritance is given in which a man inherits the attributes of human - name and age.

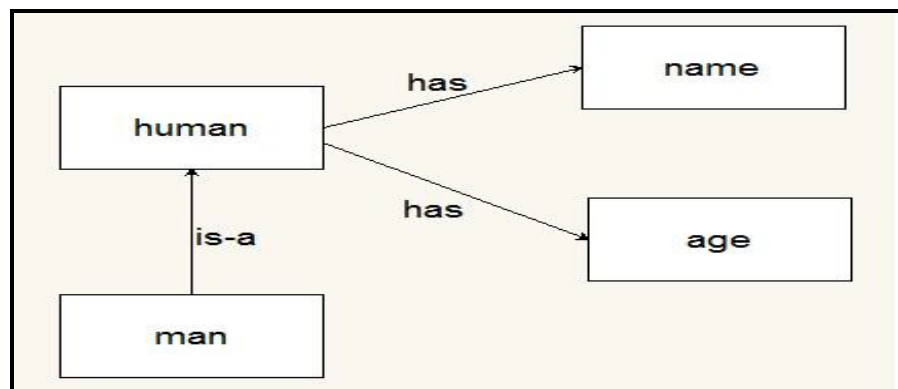


Figure 11 Inheritance of semantic network

We can specify a semantic network by indicating the basic components:

- Lexical component: nodes denoting physical objects; links are relationships between objects; labels denote the specific objects and relationships;
- Structural component: the nodes and links from a directed diagram;

- Semantic component: Definitions are related to the link and label of nodes. The facts will depend on the approval area;
- Procedural part: constructors permit a creation the new links and nodes. The destructors permit the removal of links and nodes.

The semantic network is generally characterized by a superior representation as well as significant power, which explains why many people make up a strong and adaptable approach to represent knowledge. The semantic networks have some advantages as given below:

- 1) Despite the variety of entities, they can be shown in the same semantic network;
- 2) Semantic systems supply a graphic view from the trouble place, and for this reason they may be simple to be implemented and easy to be understood;
- 3) Semantic network can be used as a typical connection application among various fields of knowledge, for instance, among computer science and anthropology;
- 4) Semantic network permits a simple approach to investigate the problem space;
- 5) Semantic network gives an approach to make the branches of related components;
- 6) Semantic network reverberates with the methods the people process data;
- 7) Semantic network is more natural than the logical representation;
- 8) Semantic network is characterized by a greater cognitive adequacy compared to logic based formalism;
- 9) Semantic network permits using of effective inference algorithm (graphical algorithm);

10) Semantic network has a greater expressiveness compared to logic.

Semantic network also provides a number of disadvantages that frequently cause problems. Some disadvantages are given below:

- 1) There is no difference between individuals and classes. The system is restricted by the user's knowledge of the definitions with the links in the semantic network. The links among nodes aren't most similar to functions. It is needed to distinguish the links, which comprise a number of connections, and links, which are structural in nature. The same links can be used to connect three nodes to show the structure of a network (Figure 12). Actually the link "is-a" is used in two different relationships - the first link labeled with "is-a" makes a relation between nodes "Ben" and "dog" that identifies that Ben is a dog, but in the second "is-a" relation the nodes "dog" and "living being" are connected to identify the category. It is necessary to specify more descriptive method name of links differentiating concerning relational and structural types demonstrated in figure 13. In such cases we rewrite the link between nodes "Ben" and "dog" as a "type-of", and the link between the objects "dog" and "living being" as a "subtype-of" link.

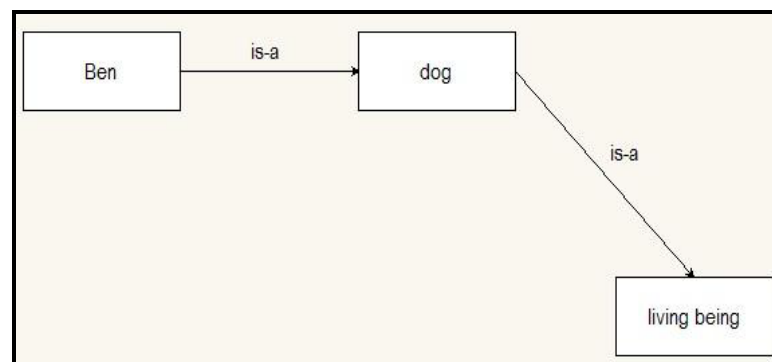


Figure 12 Repeated "is-a" link with different meanings

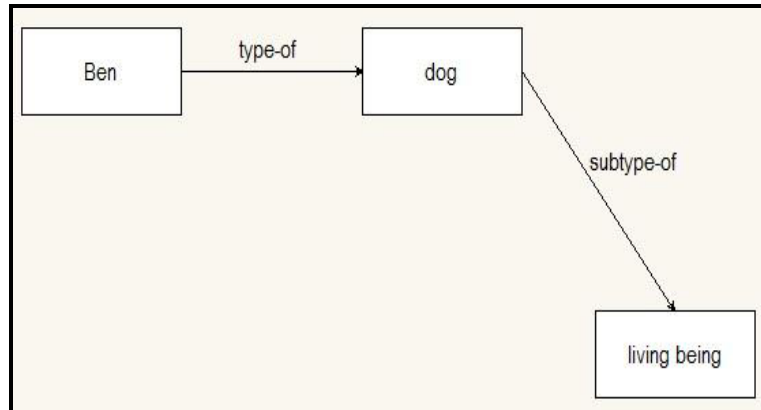


Figure 13 The “type-of” and “subtype-of” links

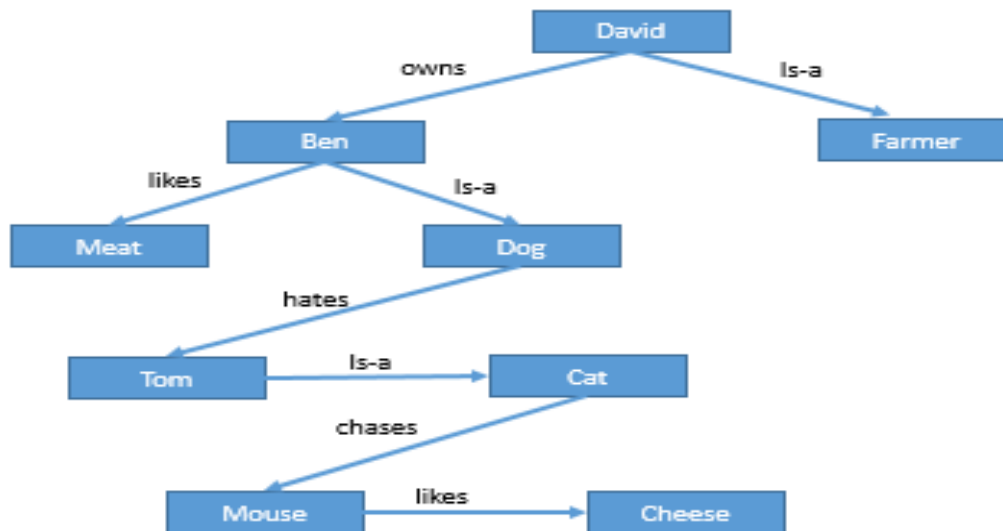
- 2) The difference between features related to a class and features comes from the individuals and from the class that doesn't exist.

- 3) A conventional semantic doesn't really exist; therefore there isn't an agreed-upon idea of what offered representational design indicates. The semantic systems are usually based upon the techniques that change them. An alternative to this problem could be both making use of conceptual diagrams, the formalism with regard to knowledge representation KL-ONE that allows conquering semantic indistinctness in the semantic system. KL-ONE is a popular knowledge representation system in semantic network and frame.

2.1.2 Frame-Based

Frame-based representation is an important knowledge representation formalism permitting us to show the concept of inheritance. The frame technique includes a number of frames or nodes that are related to each other by relationships. Every frame explains

both an instance and a class frame. Marvin Minsky presented the idea of frame in 1975 as the major way to show a range of knowledge. A frame is a group of properties identifying the condition of an object, and this object is related with other frames or objects. Actually a frame is more than only a record or perhaps a data structure that contains data. In artificial intelligence the frame is known as a slot-filler knowledge representation method. Every frame provides a number of slots, which are designated as slot values. This is the way the frame network is created. Instead of simply processing links among frames, every relationship is indicated by away from a value being put into any slot. For instance, the semantic network is represented in the form of frame in the table below. The frame system can be shown in another form called diagrammatic, and it is represented in figure 15. Whenever we point out that “Ben is a dog“, we actually mean that “Ben is an instance with the class of “dog“ or “Ben” can be a member of the class of dogs“. The “is a” connection is important in a frame-based system since it permits to state a membership associated with classes. This connection can be referred as a generalization due to the fact refereeing to the actual class associated with mammals, and is more common in comparison with the class “dog”, and the class “dog” is more common than the class “Ben”.



Frame Name	Slot	Slot Value
David	is-a	Framer
	owns	Ben
	likes	Meat
Ben	is-a	Dog
	hates	Tom
Tom	is-a	Cat
	chases	Mouse
Mouse	likes	Cheese

Figure 14 Representation of Semantic Network Frame

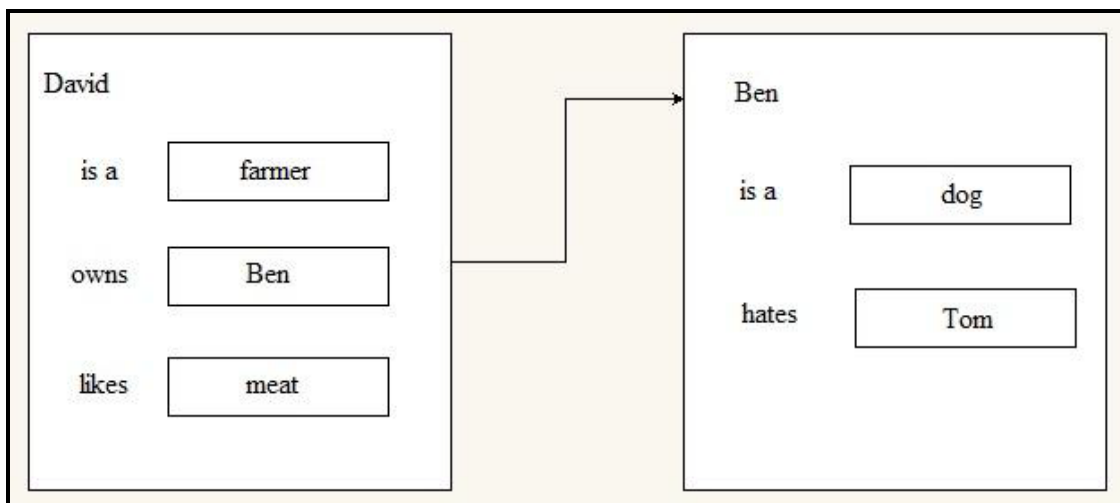


Figure 15 Diagrammatic form of frame-based system

Also it is practical to be able to discuss one object currently being a component of another object. For instance, Ben has a tail, and the tail is one of the parts of Ben. This connection is referred to as aggregation in order Ben can be viewed as an aggregate of parts of dog. Some other relations are generally called association. An instance of such a connection is the “hates” relationship shown in Table 1. This clearly shows that how Ben and Tom are related with each other. This relationship (association) has two direction meanings. The point that Ben hates Tom shows that Tom is hated by Ben, therefore we’re truly indicating two relationships in a single association.

The frame is just like a record construction and related to the fields and values which are generally slots as well as slot fillers. Generally speaking, the frame is a set of fillers and slots, which are identified as stereotypical objects. An individual frame isn’t much beneficial. The frame technique has a set of frames that can be joined together. The attribute value of one of the frames may become another frame. The frame example of

the book “Semantic Web” is represented in Table 2. Table 3 shows a frame example of personal data.

Table 1 Frame Example of the Book “Semantic Web”

Slots	Filters
Title	Semantic Web
Publisher	James and Bartlett
Author	Dr. Lixin Tao
Edition	1st
ISBN	0-76373230-3
Pages	768
Year	2016

Table 2 Frame Example of Personal Data

Slots	Filters
Name	Maria Sette
Job	Teacher
Gender	Female
Height	178 cm
Weight	50 kg
Marital Status	Single
Intelligence	High

A frame may be sometimes referred to a specific object or a group of comparable objects.

To be more specific, we use the actual instance frame while dealing with a specific object

as well as the class-frame while talking about a similar object. For example, in Table 4 the frame example of computer “Dell OptiPlex” is represented.

Table 3 Frame Example of “Dell Optiplex”

Class	Computer
Code	LW-02016
Model	Dell
Processor	Core i7
Hard Disk	500 GB
Memory	16 GB
CD-ROM	DVD-RW
Screen	21”
Mouse	Yes
Keyboard	Yes
Battery	6 Cell
Camera	1.3MP
Wireless	DW1501
Bluetooth	Yes

A class-frame explains a set of objects with typical features. The person, car, and computer are class-frames.

There are some advantages of a frame-based knowledge representation method described below:

- 1) The frame knowledge representation makes the programming simpler by grouping related data:
- 2) Compare to the knowledge representation method described in the form of production rules, the frame is flexible and intuitive in many application areas;
- 3) The frame representation is easily understood and used by people who are neither programmer nor designer of a system;
- 4) It is not hard to add slots for new attributes and relations:
- 5) It is simple to include default data and to discover the missing values.

The frame knowledge representation formalism has some disadvantages described below:

- 1) It is difficult to use the frame system in a program, so the algorithm is required in the process of using the frame in the program:
- 2) The lack of low-priced computer software;
- 3) Inference mechanism is not easily processed in a frame system.

2.1.3 Rules-Based

In a rule-based formalism, knowledge is cast in the form of *if-then* rules: *if X then Y*. For example, a rule for producing questions may be the following: *if the intention is to query the truth of P, then produce a yes/no question about P*. If-then rules, also called *condition-action* pairs or *production* rules, were developed during the seventies as a

model for human problem solving (Newell & Simon 1972). The rules can only produce actual behavior with the help of an *interpreter*, which performs a *cyclical* process where each cycle consists of three phases:

Identification. This phase determines for which rules the condition sides are currently satisfied in working memory.

Selection. It will often happen that more than one rule's condition side will be satisfied. Since it is in general not desirable for all applicable rules to fire, one or more rules are selected using a criterion, e.g. the first rule found, or the most specific rule.

Execution. The actions of the chosen rule are executed. Although such actions can take many forms, the most typical ones involve the addition to or removal from working memory of certain facts. Some of the facts added may represent a solution to the given problem.

Production rule systems allow both *forward chaining* and *backward chaining*. In forward chaining, inference is *data-driven*, i.e. states in working memory activate rules when their left sides match the current state. Execution of these rules may in turn activate other rules or achieve goals. In backward chaining, inference is *goal-driven*, i.e. goals are asserted and conditions of rules achieving the goals are introduced as new goals when they are not present in working memory. In the latter case, the right-hand side of the rules is used to identify applicable rules.

The rule-base acts as a kind of *long-term memory*, whereas the working memory, which describes the conditions that are satisfied at some point during computation, acts as a *short-term memory*. Rule-based architectures have been further developed toward more sophisticated cognitive models like Act* (Anderson 1983) and Soar (Newell, Rosenbloom & Laird 1989). The Act* system has a semantic network as part of its long-term memory. Soar contains special mechanisms for remembering the results of previous computations in long-term memory. [7]

2.1.4 Logic-Based

Logic based Knowledge Representation formalism include Descriptive Logic, Modal Logic, and Non-monotonic Logic. As stated by Franz Baader, “a knowledge representation formalism should allow for the symbolic representation of all the knowledge relevant in a given application domain” [10]. Knowledge representation formalism such as Semantics Network and Frames was motivated by attempts to provide a structured representation of knowledge [10]. Marvin Minsky who developed Frames and defined frames as “a data-structure for representing a stereotyped situation” combined his introduction of the frame idea with a general rejection of logic as a KR formalization [24]. According to Nilsson, “many database systems and expert systems can be said to use declarative knowledge, and the ‘frames’ and ‘semantic networks’ used by several AI programs can be regarded as sets of declarative sentences” [25]. Semantic Network was developed by Quillian for representing the semantics of natural language that represents concepts and objects as nodes in a graph that has two different edges. The

first is a property edge for example assigning properties such as *color* to a concept and the second an “IS_A” edge that introduces hierarchical relations among concepts. Figure 16 depicts such edges [4].

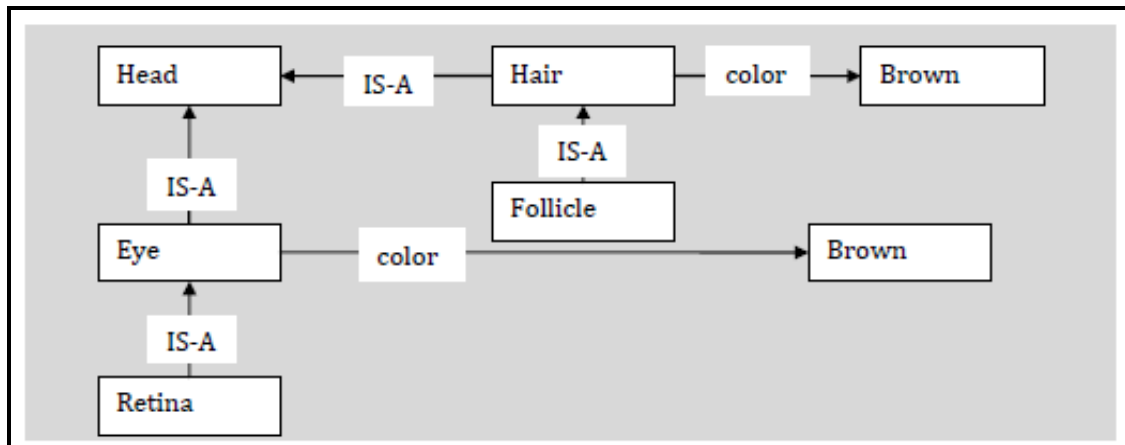


Figure 16 Semantic Network Edges

The primary idea of Descriptive Logic (DL) is to commence with atomic concepts (unary predicates) and roles (binary predicates) to build complex concepts and roles using a small set of adequate constructors [24]. DL has been implemented in an array of application domains such as natural language processing [26], configuration of technical systems [32], software information systems [6], optimizing queries to databases [5], and for support in planning [22]. “Modal logic is, strictly speaking, the study of the deductive behavior of the expressions ‘it is necessary that’ and ‘it is possible that’” [11]. Modal Logic KR extends propositional logic by unary operators, which are called a box and diamond operators [24]. The box operator implies “Necessarily” and the diamond \diamond property implies “Possibly”. “It will snow today” is an example of a “Possibly” modality in that it implies the possibility to snow today. Non-monotonic Logic is KR language

based on classical logics such as the First order predicate in that if a statement Φ can be derived from a knowledge base then Φ can be derived from any larger knowledge base [24]. An advantage of this property is that once an inference is made it should not be revised when more information is received. A disadvantage to this property is that an inconsistency may result when additional information received contradicts the inference rendering the knowledge base useless. To avoid this situation, when plausible conclusions are drawn from available knowledge and the newly acquired knowledge show that some of the plausible conclusions are wrong, the plausible conclusions are withdrawn and do not result in inconsistencies [24]. In the Closed World Assumption (CWA), which by default assumes that the available information is complete, if an assertion cannot be derived using classical inference from the knowledge base, then CWA deduces to negation. Practical application of this assumption is employed in relational databases and in Logic Programming languages with “Negation of Failure” [19].

2.1.5 Ontology-Based

Ontologies were developed in Artificial Intelligence to facilitate knowledge sharing and reuse. The reason ontologies are becoming so popular is due to what they promise: a shared and common understanding of some domain that can be communicated between people and application systems. Also the growth of disciplines such as Artificial Intelligence and machine learning has resulted in the need to structure knowledge with sufficient clarity that people and machines can share common understanding of the terms or concepts in the knowledge domain. The development of ontologies has been driven

by the need to develop a framework for a common vocabulary that would enable people and software agent to share a common understanding of the meaning of the basic concepts and terms in a given domain of knowledge as well as the relationships between them. The distinction between an ontology and a knowledge base, a knowledge base is the addition of individual instances into the ontology. The advantages of an ontology;

- It abstracts the essence of a concept.
- It catalogues and distinguishes various types of objects and their relationships.
- It facilitates communication, sharing and reuse of domain knowledge.
- It enables explicit specifications of domain assumptions.
- It provides a means for separating operational knowledge from domain knowledge.

An ontology typically consists of hierarchical arrangements of the classes which describe the major concepts in the domain and subclasses which are the more specific concepts under a particular class. The properties of the classes and subclasses description of features and attributes are specified as well as the relevant restrictions. There are five kinds of ontology components classes, relations, functions, formal axioms and instances.

- a) Classes represent concepts, which can be considered generic entities.
- b) Relations represent a type of association between concepts of the domain.
- c) Functions are special case of relations.

- d) Formal axioms serve to model sentences that are always true. They are normally coherent description between Concepts / Properties / Relationships is logical expressions.
- e) Instances are used to represent elements or individuals in an ontology.

Web Ontology Language OWL is one of the most used languages for creating ontologies, which is the latest recommendation of W3C. The OWL consists of individuals, properties, and classes. Individuals: are instance of a class. For example, a Mathew is an individual or instance of a Student class or Italy is an individual or instance of a Country class. Figure 17 shows the Representation of Individuals.

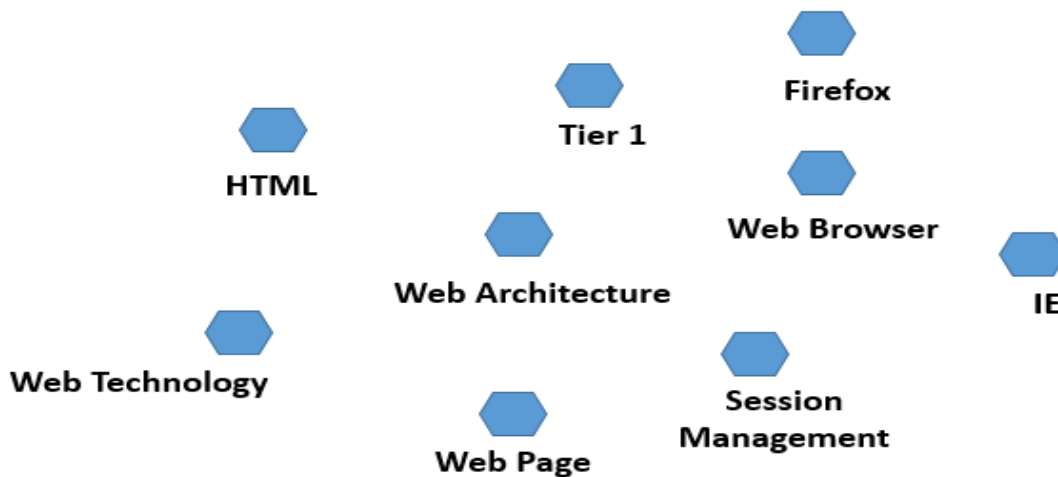


Figure 17 Representation of Individuals

Properties: relate two individuals together. For example, Web Technology Include Web Architecture. Properties have many characteristics such as inverse, reflexive, symmetric, transitive, or asymmetric. Figure 18 shows the Representation of Properties.

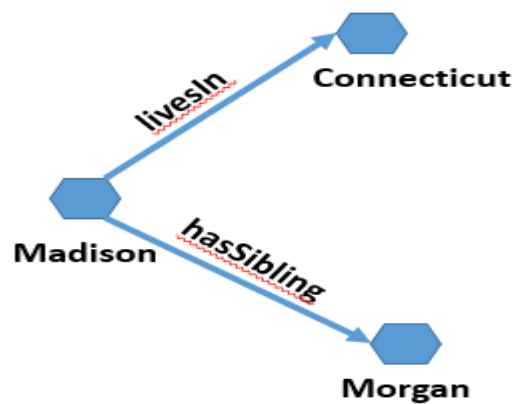


Figure 18 Representation of Properties

Classes: can be described as a set that has individuals as members. Examples of a class are Person, and Pet. Figure 19 shows the Representation of Classes.

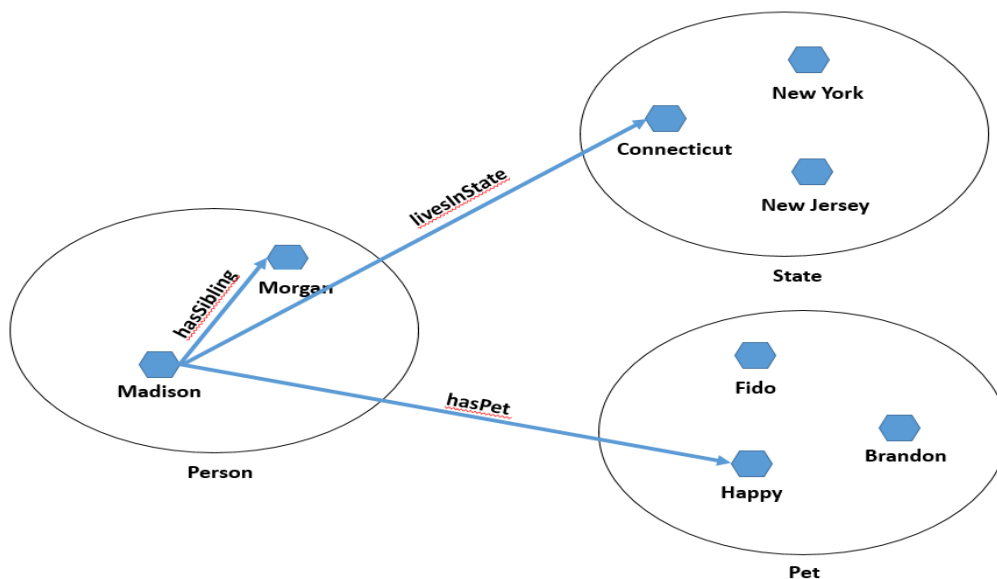


Figure 19 Representation of Classes

OWL uses the Triple to describe the relations between two entities. Figure 20 depicts this relation.

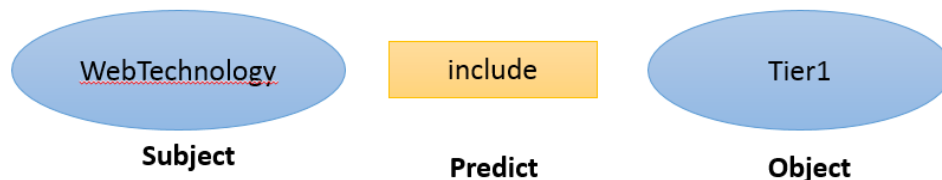


Figure 20 Example of Triple Representation

The most popular tool for creating OWL files is the tool called Protégé. There are many versions of Protégé such as the Stanford University Protégé. Stanford University Protégé: This protégé uses only one relation the “is-a” relation. For example, “Dog is-a Pet”. In this triple, “Dog” is the subject, “is-a” is the predicate, and “Pet” is the object. With “is-a” as the only relation to use between two classes, properties may have to be used to capture a more vivid knowledge of the domain of interest. Figure 21 depicts an ontology created with the “is-a” relation of a class “Pet” and its subclasses.

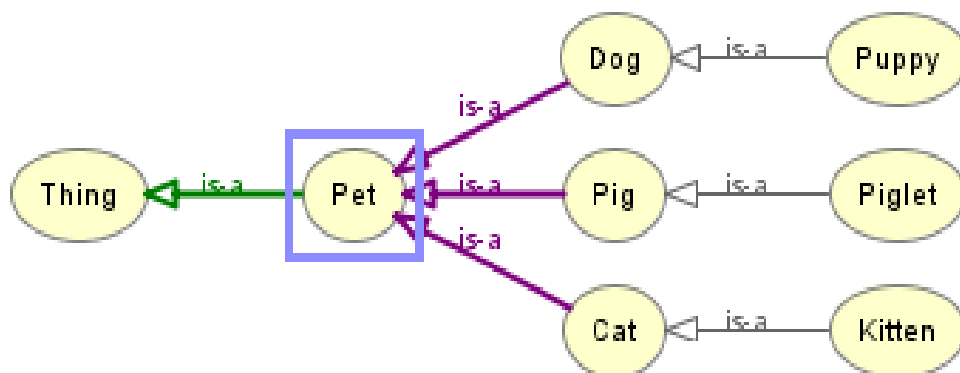


Figure 21 Pet Relations Using “is-a” Predicate

2.1.6 Knowledge Graph

Recent researches have also explored the searching domain by applying knowledge graphs [26]. One attempt was uses relevance feedback to improve graph query that was

based on the implementations of knowledge graphs. The graphical-based approach provides a method of mapping relations between relevant entities. The knowledge graph commonly used to represent conceptual knowledge structures and their link with other artifacts. The knowledge graph expression of *Resource Description Framework* (RDF) language is an example that illustrates the flaws, such as arbitrary positioning and the lack of visual expression for rules, the knowledge graph is used to represent the mechanisms at work in conceptual representation. It is intensively used in works [23] for categorizing systems. There are two types of objects involved in conceptual knowledge navigation: *Conceptual Structures* where nodes are concepts and *Edges Represent Semantic Relations* between the concepts. Based on this awareness, ontology can be visualized as graphs and thus often play the role of conceptual structure for navigation [6]. Hypertexts, where each node is a resource and edges represent hypertext links between documents. What we mean by documents is that small entities that can be addressed by its URL. A web page may contain anchors to several documents or resources were we identify each resource to a learning object. Knowledge structures, where nodes are concepts and edges represent semantic relations between concepts. In adaptive hypermedia, the conceptual representations are generic and thus are different from ontologies.

Many methods that are used for document representation rely on vector space model. The documents are represented as a linear vectors and co-occurrence of the words in text document corpus. Many semantic relations among concepts and significant information

are lost when a vector space model is used. If a document is long, it is very difficult to represent it as a vector model due to large dimensionality. The main benefit of knowledge graph-based method is that it allows keeping the structural information inherent to the source document. The methodology of knowledge graph-based representation contains definitions of graph based, sub graph and graph isomorphism. Knowledge Graph based representation is an initial stage for text mining. It concentrates on how to represent text documents as graph. The graph-based representation relies on the processing of the text level. For example, Google Knowledge Graph Search API enables users to query the Knowledge Graph database for specific information about entities residing in the Knowledge Graph database [29]. Google's Freebase Knowledge Graph Search API has been discontinued since December 2014, and as been taken over by Wikidata. Wikidata is part of Wikipedia, which allowed users to insert data in the knowledge graph database, the new Google Knowledge Graph Search API allows users to only query entities from the Knowledge Graph or Knowledge Vault databases and return results in JSON-LD format [29]. The Pace University Knowledge Graph (KG) provides support to extend and transform OWL with custom relations into a knowledge graph. The Pace University Knowledge Graph introduces minimal syntax extension to OWL so it can benefit from existing tools for OWL. It has the ability to relate two classes using various custom relations. Figure 22 depicts a Hand class with custom created relations. It is evident that in addition to the "is-a" relation, the "partOf" custom relation was used to relate two classes in a meaningful way.

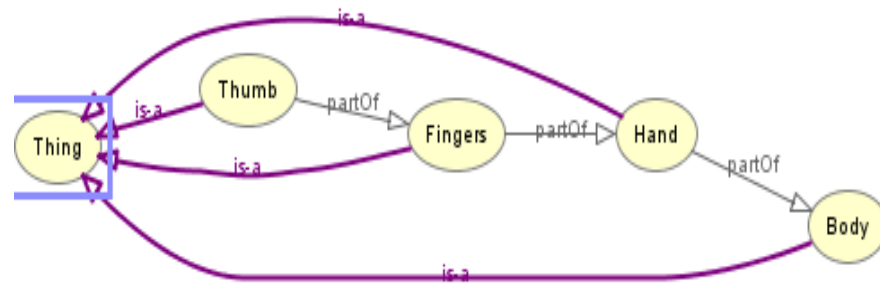


Figure 22 Hand Relations Using “partOf” Relation

While OWL is the current industry standard for knowledge representation, it mainly supports ontologies, which is based on a single “first-class” relation “is-a”, or inheritance. But each knowledge domain has its own rich custom relations among its concepts, and partOf and time dependency are especially important in engineering and algorithm related knowledge representation. This research uses Pace University’s knowledge graph to support a set of custom relations unique to the specific knowledge to be taught. Instructor’s teaching experience, in form of recommended learning order for students with various background, are also encoded in the knowledge graph and used to provide the base line of the personalized learning process of each student. Online assessment at the end of each learning object further supports specific to general or the opposite so the student can freely navigate in the knowledge space and further customize the learning process.

2.2 Learning Objects

Learning objects are things in the world that contain learning material. The learning materials in a learning object can be structured in a meaningful way that can be tied to an educational learning objective [21]. They can vary in size, scope and level of granularity ranging from small chunks of instruction to a series of combined resources to provide a more complex learning experience [13]. Some of the types of Web-based learning object materials can be a web page, web scripts, style sheets, documents, photos, videos, slides, audio, blogs, links, movies, simulations, multimedia, and many other learning object types that can be used, organized and reused for Cyberlearning education. LOs are not interchangeable because the objects are not of consistent sizes, nor are they written in consistent languages. LOs are interactive, text, html, flash, Java, and Java-Script. While others LOs are in audio, multimedia, PowerPoint, and movies. The proposed problem is finding out a way to link the Lobs according to custom relations. Moreover, LOs lack consistent classification schemes. There have been initiatives that focus on the need to develop a classification scheme, to catalogue the objects so that individuals can retrieve and organize them. Another issue is learning levels are not resolved in a consistent way, nor are the points of authorship, copyright, language of object, and many other issues. The current issues with *LO Metadata* (LOM), is that it is an XML-based development, which emphasizes on syntax and format and is not semantically driven by knowledge representation. LOM does have the advantage of data transformation and digital libraries, but lacks the semantic metadata to provide reasoning and inference. XML-based markup language is limited in this approach because it causes compatibility problems with existing data applications. LOs has its advantages and disadvantages; however. The lack

of technical experience may deter some technologically inexperienced staff to create web-based LOs; they would be more comfortable using software, which they are familiar with. It can also be also time consuming to create a high quality web-based LO and subsequently the workload of the author will be increased [25]. Effective pedagogy is lost if the author does not have a clear educational goal when designing a LO. Also, since the definitions and the size of the Los are fairly unclear, it is not certain how much content is contained in one single LO and it is difficult to construct LOs which are independent of each other and with no context in mind [5]. Access to digital materials is now very easy; however, permission must be obtained and correct attribution must be provided if you are using learning materials in your LO which have been created by someone else [5]. The requirement of LOs to be reusable has meant that many authors have had to reformat all their existing learning content before it can be reused. For example, a PDF user manual for a piece of software is required to be deconstructed into several smaller components, and converted into XML before storing it into a database so that it can be reused in a LO system. A high percentage of current digital educational materials cannot be reused because they have not been decomposed and workload is therefore increased.

2.3 Web-Based Technologies

2.3.1 JAVA

The Java programming language [12] is ideal for Web-based applications because of its rich network APIs and its ability to run on diverse operating systems and architectures. Most of the technologies used in this dissertation use Java directly or indirectly, because they are written as Java applications or as Java servlets [1]. Servlets are used to extend

the capabilities of Web servers by enabling programmers to produce interactive, dynamic Web content based on user actions and data content. Servlets run within a container that manages the servlet's interaction with the server's operating system and Web server. Servlet containers are typical components of commercial and open-source Web and application servers, and can also be implemented as stand-alone Web services.

2.3.2 JavaScript

JavaScript is a cross-platform, object-oriented scripting language. It is a small and lightweight language. Inside a host environment for example, a web browser, JavaScript can be connected to the objects of its environment to provide programmatic control over them. JavaScript contains a standard library of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements. Core JavaScript can be extended for a variety of purposes by supplementing it with additional objects, for example:

Client-side JavaScript extends the core language by supplying objects to control a browser and its Document Object Model (DOM). For example, client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation.

Server-side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server. For example, server-side extensions allow an application

to communicate with a database, provide continuity of information from one invocation to another of the application, or perform file manipulations on a server [41].

2.3.3 JavaServer Pages (JSP)

JSP stands for Java Server Pages, which is added to server-side code to an HTML pages. JSP is a Java platform technology for building and supporting applications containing dynamic Web content such as HTML, DHTML or XM, which helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with `<%` and end with `%>`. A JSP page is a text-based document containing static HTML and dynamic actions that describe how to process a response to the client. At development time, JSPs are very different from Servlets. However, they are precompiled into Servlets at runtime and executed by a JSP engine, installed on a Web-enabled application server such as Tomcat 7.0 [20]

Servlets provide platform-independent, 100% Pure Java server-side modules and fit seamlessly into the framework of the application server. Servlets are compiled into Java bytecode that can be downloaded or shipped across the network; they are truly "Write Once, Run Anywhere." Unlike CGI scripts, Servlets involve no platform specific consideration or modifications: they are Java application components that are downloaded, on demand, to the part of the system that needs them [34].

The drawback of this approach is that the creation of the page must be handled in the Java Servlet, which means that if Web page designers want to change the appearance of the page, they would have to edit and recompile the Servlet. With this approach, generating pages with dynamic content still requires some application development experience. Clearly, the Servlet request-handling logic needs to be separated from the page presentation [20].

The solution is to adopt the Model-View-Controller (MVC) paradigm for building user interfaces. With MVC, the back-end system is the tutoring systems logic model, the templates for creating the look and feel of the response is the View, and the code that glues it all together is the Controller. JSPs fit perfectly into this solution as a way of creating a dynamic response or View. Servlets containing logic or managing requests act as the Controller, while the existing business logic rules that implement and realize all the functions of the tutoring system model [20].

2.3.4 AJAX

Asynchronous JavaScript and XML (Ajax) is the most technology often used in Web 2.0 applications. It uses JavaScript to upload and download data from the web server and update parts of a web page without reload the pages. This method allows pages to function more like desktop-based applications rather than as old-fashioned static content pages. It is a technique for creating fast and dynamic webpages [42].

2.3.5 Java Servlets

The Java servlet is a server-side technology that accepts HTTP requests from a Web browser and returns HTTP responses. Servlets, which can be multi-threaded, have performance advantages over CGI for coding presentation logic for a Web client. Servlets operates on the request-response model. Requests come into the servlet engine. The server then executes the appropriate servlet and returns a response to the client. The most commonly used servlet type is the HTTP servlet designed to fill HTTP protocol requests. HTTP servlets provide the following core features:

- HttpRequest objects capture request details from requests submitted via Web page forms, including data availability, protocol types, security levels, and so forth
- HttpSession objects specific to each user handle user session information in the server.
- HttpResponse object capture response details. The servlet developer can output everything that is sent back to the client making the request. The servlet engine handles the rest [8].

Servlet-based applications avoid a lot of processing overhead. Using threads instead of processes means that a Servlet can retain data between requests. Threading and persistence make it easier to develop high performance solutions.

2.3.6 Tomcat

The Tomcat server is an open source, Java-based web application container that was created to run servlet and JavaServer Pages (JSP) web applications. It was created under the Apache-Jakarta subproject; however, due to its popularity, it is now hosted as a separate Apache project, where it is supported and enhanced by a group of volunteers from the open source Java community. Tomcat is very stable and has all of the features of a commercial web application container yet comes under Open Source Apache License. Tomcat also provides additional functionality that makes it a great choice for developing a complete web application solution. Some of the additional features provided by Tomcat other than being open source and free include the Tomcat Manager Application, specialized realm implementations, and Tomcat valves. This research uses the latest Tomcat version 7.0, which is a reference implementation of current Servlet API 3.0, JSP API 2.2 and JDK 1.6 [34].

2.3.6.1 Tomcat Architecture

A Tomcat instance, or server, is the top-level component in Tomcat's container hierarchy. Only one Tomcat instance can live in a single Java Virtual Machine (JVM). This approach makes all other Java applications, running on the same physical machine as Tomcat server, safe in case Tomcat and/or its JVM crashes. Tomcat instance consists of grouping of the application containers, which exist in the well-defined hierarchy. The key component in that hierarchy is the Catalina servlet engine. Catalina is the actual Java servlet container implementation as specified in Java Servlet API. Tomcat 7 implements Servlet API 3.0, the latest specification from Sun [34].

Below is an XML representation of the relationships between the different Tomcat containers.

```

<Server>
  <Service>
    <Connector />
    <Engine>
      <Host>
        <Context> </Context>
      </Host>
    </Engine>
  </Service>
</Server>

```

Figure 23 shows the relationship of the main components of Tomcat architecture that correspond with the described XML code snippet.

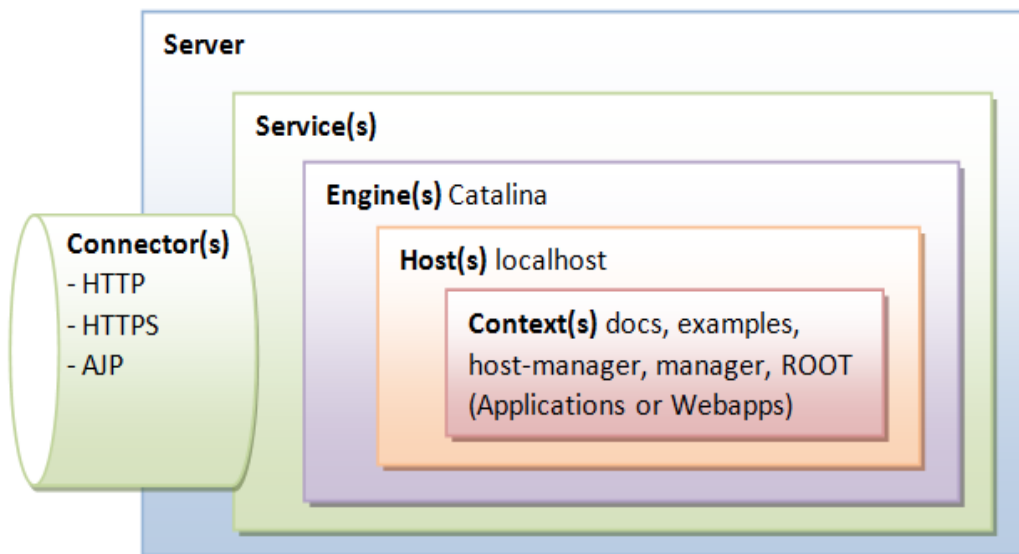


Figure 23 Tomcat Architecture with Main Components [37]

This instance can be broken down into a set of containers including a server, a service, a connector, an engine, a host, and a context. By default, each of these containers is configured using the server.xml file [34].

The Server

The first container element referenced in this snippet is the **<Server>** element. It represents the entire Catalina servlet engine and is used as a top-level element for a single Tomcat instance. The **<Server>** element may contain one or more **<Service>** containers [34].

The Service

The next container element is the **<Service>** element, which holds a collection of one or more **<Connector>** elements that share a single **<Engine>** element. N-number of **<Service>** elements may be nested inside a single **<Server>** element [34].

The Connector

The next type of element is the **<Connector>** element, which defines the class that does the actual handling requests and responses to and from a calling client application [34].

The Engine

The third container element is the **<Engine>** element. Each defined **<Service>** can have only one **<Engine>** element, and this single **<Engine>** component handles all requests received by all of the defined **<Connector>** components defined by a parent service [34].

The Host

The **<Host>** element defines the virtual hosts that are contained in each instance of a Catalina **<Engine>**. Each **<Host>** can be a parent to one or more web applications, with each being represented by a **<Context>** component [34].

The Context

The **<Context>** element is the most commonly used container in a Tomcat instance. Each **<Context>** element represents an individual web application that is running within a defined **<Host>**. There is no limit to the number of contexts that can be defined within a **<Host>** [34].

2.3.6.2 The Default Tomcat Directory Structure

The Tomcat installation directory is referred to as CATALINA_HOME. Table 4 describes the directories that compose a Tomcat installation. It is assumed that each of these directories is contained within the CATALINA_HOME directory [34].

Table 4 Tomcat Default Directory Structure

Directory	Contents
/bin	Contains the startup and shutdown scripts for both Windows and Linux. Jar files with classes required for tomcat to start are also stored here.
/config	Contains the main configuration files for Tomcat. The two most important are server.xml and the global web.xml.

/lib	Contains the Tomcat Java Archive (jar) files, shared across all Tomcat components. All web applications deployed to Tomcat can access the libraries stored here. This includes the Servlet API and JSP API libraries.
/logs	Contains Tomcat's log files.
/temp	Temporary file system storage.
/webapps	The directory where all web applications are deployed, and where you place your WAR file when it is ready for deployment.
/work	Tomcat's working directory where Tomcat places all servlets that are generated from JSPs. If you want to see exactly how a particular JSP is interpreted, look in this directory.

2.3.6.3 Tomcat Servlet

Tomcat Servlet is an interface defined in `javax.servlet` package. It declares three essential methods for the life cycle of a servlet - `init()`, `service()`, and `destroy()`. They are implemented by every servlet defined in SDK or self-defined and are invoked at specific times by the server.

1. The `init()` method is invoked during initialization stage of the servlet life cycle. It is passed an object implementing the `javax.servlet.ServletConfig` interface, which allows the servlet to access initialization parameters from the web application.
2. The `service()` method is invoked upon each request after its initialization. Each request is serviced in its own separate thread. The web container calls the `service()` method of the servlet for every request. The `service()` method determines the kind of request being made and dispatches it to an appropriate method to handle the request.

3. The `destroy()` method is invoked when the servlet object should be destroyed. It releases the resources being held.

The life cycle of a servlet object, is that the servlet classes are loaded to container by the class loader dynamically. Each request is in its own thread, and a servlet object can serve multiple threads at the same time. Threads are not safe. When it is no longer being used, it should be garbage collected by JVM (Java Virtual Machine).

Using servlets allows the JVM to handle each request within a separate Java thread, this is one of the key advantages of Servlet container. Each servlet is a Java class with special elements responding to HTTP requests. The main function of Servlet container is to forward a requests to the correct servlet for processing, and return the dynamically generated results to the correct location after the JVM has processed them. In most cases servlet container runs in a single JVM, but there are solutions where containers need multiple JVMs.

2.3.6.4 Tomcat Servlet Container Process

The Tomcat parsing process of the servlet container must have an instance of an object that calls the start method for Tomcat. The startup logic for Tomcat is based on the observer design pattern, all containers will inherit the lifecycle interface, which manages the containers modifications and state and will go by the notification of the registered observers (Listeners). The initialization of the web application is implemented in the `configureStart` method in `ContextConfig`. The application initializes the parsing `web.xml` files, this file describes the key information of a web application. The `web.xml` file for

each configuration item will be parsed into the corresponding attribute and is stored in a WebXML object. The properties of the WebXML object will be set to the Context container, this includes creating the servlet object, filter, listener and much more. The Context container is really running the Servlet container. The web application is corresponding to a Context container, the container configuration attributes is specify by the application of the web.xml.

2.3.6.5 Tomcat Servlet in Action

The Tomcat servlet request is first initiated by a user from the web browser interface to the server, which will contain the following information: `http://hostname:port/contextpath/servletpath`, the hostname and port is establish by a TCP connection, the URL is used to select the server to the sub container service from a user request. The Tomcat URL and servlet container will do the complete mapping by the `org.apache.tomcat.util.http.mapper`. The mapping of the hostname, the contextpath host, and the context container is provided to the `mappingData` property from the Request. All requests are mapped to a servlet and then it implements the service method, this method is the MVC framework. When the servlet is removed from the servlet container the lifecycle is over and the servlet destroy method will be invoked. The listener implementation class can be configured in the web.xml as `<listener>` label and we can add additional listeners. The `SerlvetContextListener` in container startup monitors the events. The web.xml filter is another common configuration item which uses the `<filter>` and `<filter-mapping>` label. The filters can do the same things as the servlet, and even more. It is flexible because it provides request and respond objects and also the

FilterChain object, which makes the controls more flexible for the transfer request. This provides the mechanism to allow for intercepts, which allows for the increase or decrease of items requested. The web.xml <servlet-mapping>, the <filter-mapping> and the <url-pattern> configuration items roles are to match a request which will execute the servlet or filter. The servlet request is completed through the org.apache.tomcat.util.http.Mapper class, this class will match the configuration in each Servlet according to the request to the URL. The filter url-pattern matching is performed in an ApplicationFilterChain object. It will match all the definition of the filter url-pattern with the current URL. The matching filter is stored in the filters array in the ApplicationFilterChain, and then the FilterChain in turn is called. The web.xml loading will first check the <url-pattern> configuration with the rules. This process will examine the validateURLPattern method in StandardContext. If it is not successful, the Context container restart will fail and will report to the java.lang.IllegalArgumentException:Invalid <url-pattern> /a/*.htm in the servlet mapping error.

2.3.7 Spring MVC

The Spring Web MVC framework provides model-view-controller architecture and ready components that can be used to develop flexible and loosely coupled web applications. The MVC pattern results in separating the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements [14].

- The **Model** encapsulates the application data and in general they will consist of POJO.
- The **View** is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.
- The **Controller** is responsible for processing user requests and building appropriate model and passes it to the view for rendering.

Spring's MVC module is based on front controller design pattern shown in Figure 24 followed by MVC design pattern shown in Figure 25. All the incoming requests are handled by the single servlet named DispatcherServlet, which acts as the *front controller* in Springs MVC module shown in Figure 26. The DispatcherServlet then refers to the HandlerMapping to find a controller object, which can handle the request. DispatcherServlet then dispatches the request to the controller object so that it can actually perform the business logic to fulfill the user request. (Controller may delegate the responsibility to further application objects known as service objects). The controller returns an encapsulated object containing the model object and the object view (or a logical name of the view). In Springs MVC, this encapsulated object is represented by class ModelAndView. In case ModelAndView contains the logical name of the view, the DispatcherServlet refers the ViewResolver to find the actual View object based on the logical name. DispatcherServlet then passes the model object to the object view, which is then rendered to the end user [14].

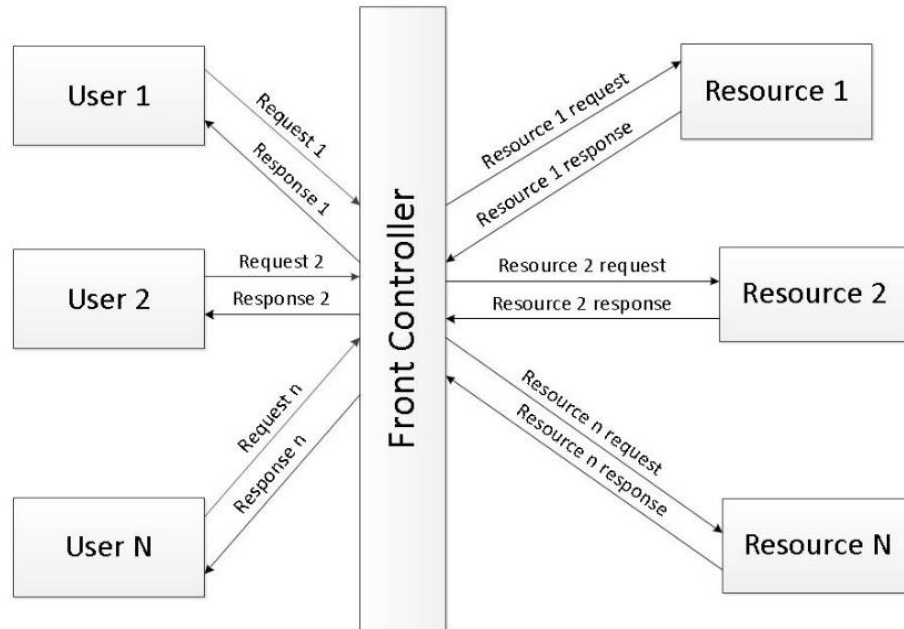


Figure 24 Front Controller Design Pattern [14]

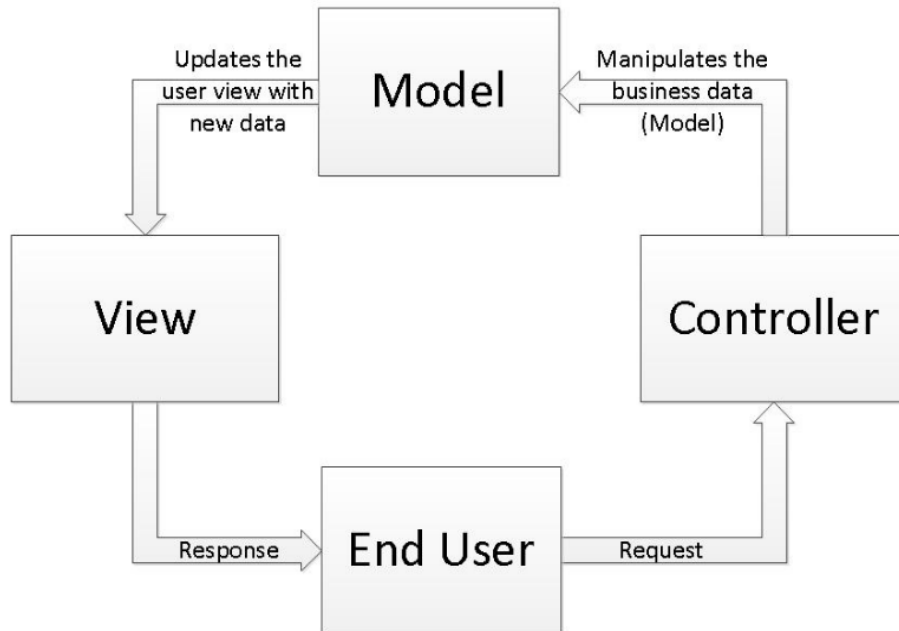


Figure 25 Spring MVC Design Pattern [14]

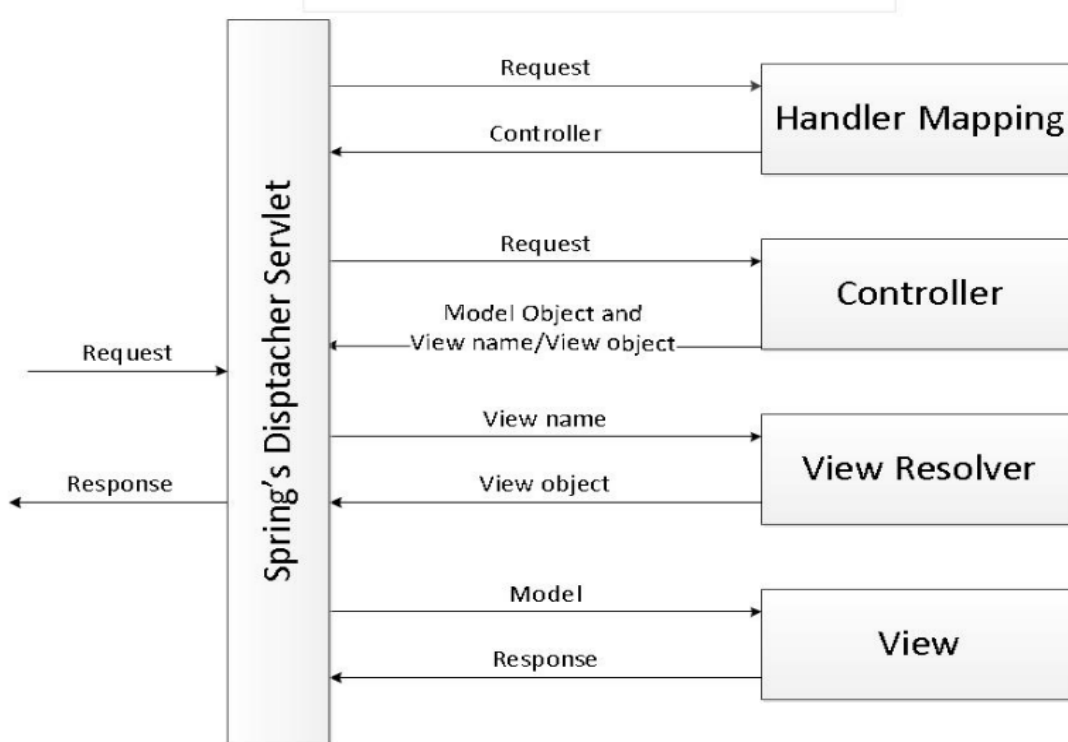


Figure 26 Spring MVC Architecture [14]

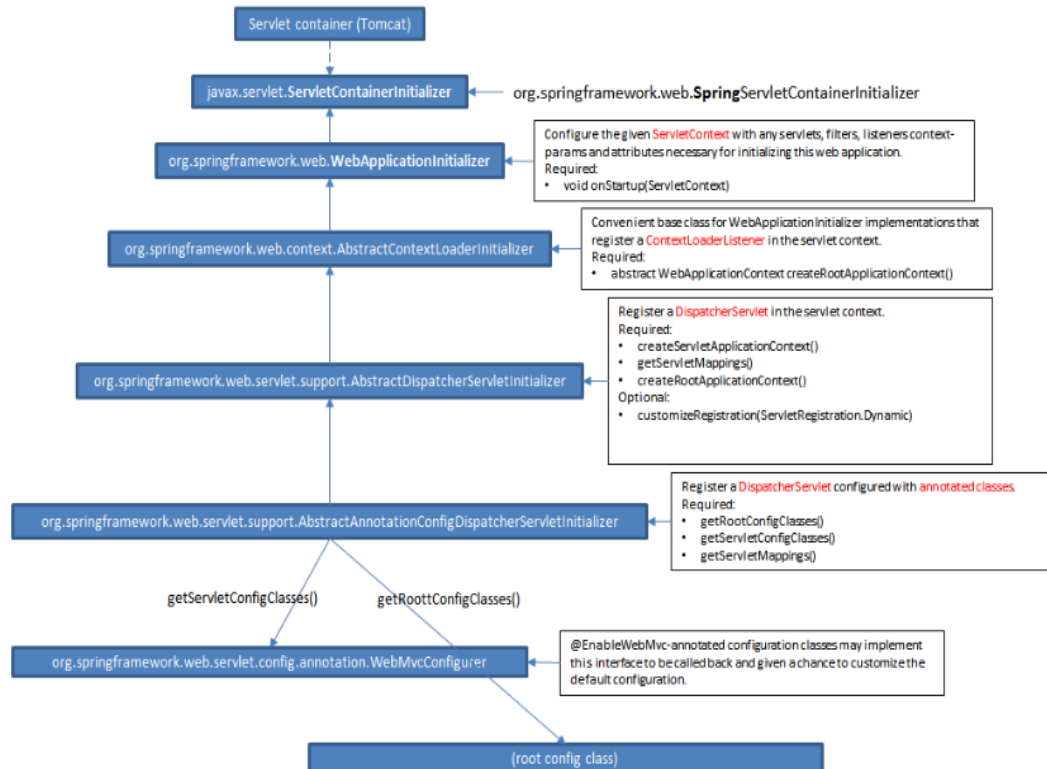


Figure 27 Spring Web Configuration Hierarchy

In Figure 27 is a spring web configuration hierarchy explaining the how each component interacts with each other.

Spring MVC Request Life Cycle

This section describes the Spring MVC request life cycle. Figure 28 is based on these components and their purposes are as follows:

- Filter: The filter applies to every request. Several commonly used filters and their purposes are described in the next section.
- Dispatcher servlet: The servlet analyzes the requests and dispatches them to the appropriate controller for processing.
- Common services: The common services will apply to every request to provide supports including i18n, theme, file upload, and so on. Their configuration is defined in the DispatcherServlet's `WebApplicationContext`.
- Handler mapping: This maps the request to the handler (a method within a Spring MVC controller class). Since Spring 2.5, in most situations the configuration is not required because Spring MVC will automatically register the `org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping` class that maps handlers based on HTTP paths expressed through the `@RequestMapping` annotation at the type or method level within controller classes.
- Handler interceptor: In Spring MVC, you can register interceptors for the handlers for implementing common checking or logic. For example, a handler interceptor can check and ensure that only the handlers can be invoked during office hours.
- Handler exception resolver: In Spring MVC, the `HandlerExceptionResolver` interface

(under the package `org.springframework.web.servlet`) is designed to deal with unexpected exceptions thrown during request processing by handlers. By default, the `DispatcherServlet` registers the `DefaultHandlerExceptionResolver` class (under the package `org.springframework.web.servlet.mvc.support`). This resolver handles certain standard Spring MVC exceptions by setting a specific response status code. You can also implement your own exception handler by annotating a controller method with the `@ExceptionHandler` annotation and passing in the exception type as the attribute.

- **View Resolver:** Spring MVC's `ViewResolver` interface (under the package `org.springframework.web.servlet`) supports view resolution based on a logical name returned by the controller. There are many implementation classes to support various view resolving mechanisms. For example, the `UrlBasedViewResolver` class supports direct resolution of logical names to URLs. The `ContentNegotiatingViewResolver` class supports dynamic resolving of views depending on the media type supported by the client (such as XML, PDF, JSON, and so on). There also exists a number of implementations to integrate with different view technologies.

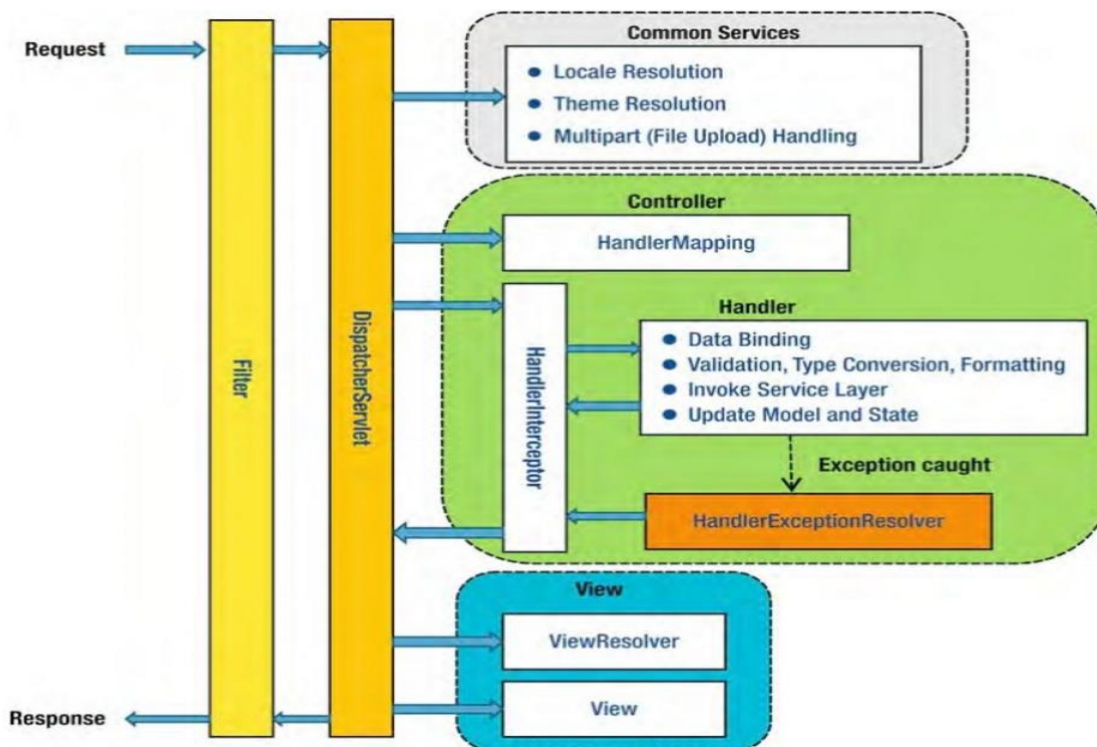


Figure 28 Spring MVC Request Life Cycle

2.3.8 Apache Jena

Apache Jena is an open source Semantic Web framework for Java. It provides an API to extract data from and write to RDF graphs. The graphs are represented as an abstract "model". A model can be sourced with data from files, databases, URLs or a combination of these. A Model can also be queried through SPARQL 1.1. Jena is similar to Sesame; though, unlike Sesame, Jena provides support for OWL (Web Ontology Language). The framework has various internal reasoners and the Pellet reasoner (an open source Java OWL-DL reasoner) can be set up to work in Jena [16].

2.3.9 XML Parsers

XML parsers simply read XML documents and provide the application with any information it needs. It reads through the characters in the document and makes a determination on what is a markup character and what is a data [19]. There are two main parsers, which are included in android. Each parser has its pros and cons. They are called SAX and DOM Parser: DOM stands for Document Object Model. DOM builds an in-memory tree representation of the XML document. Generally it is an application for validating XML Documents. [33] It defines the objects and properties of all elements in xml and the methods to access them, where the elements and attributes are represented as nodes in a hierarchical tree structure as shown in Figure 29. Each node stores the element's name and attributes along with pointer information indicating the parent-child-sibling relationship and the node value. Although it is memory inefficient, this memory structure provides DOM with its basic feature, random access. Unlike SAX sequential parsing, DOM provides fast random access to any and every node, due to its tree hierarchical structure [15].

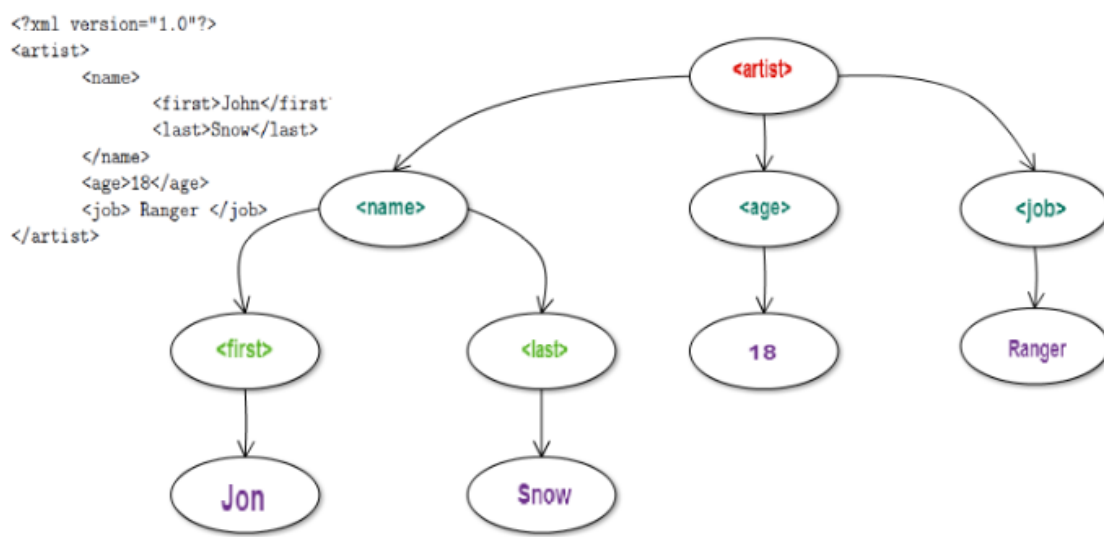


Figure 29 DOM Parser Workflow

However, DOM has one major drawback: the data in the tree can only be accessed when the tree is finished like the parser is done with parsing the whole XML document. Thus, complex and large XML documents will not be available any time before the needed memory and space are allocated and the parsing is complete [35]. Sax stands for Simple API for XML. Sax is an event based XML parser, as it parses XML documents step by step. It is considered a light-weight and fast parser with low memory consumption when compared to a DOM parser due to the fact that, instead of creating hierarchical structure representation of all elements and maintaining pointers to indicate the parent-child-sibling relationship between them, it resets when it encounters opening tag, element or attribute and works accordingly. When a certain event is triggered, it calls the corresponding function to handle the event as shown in Figure 30.

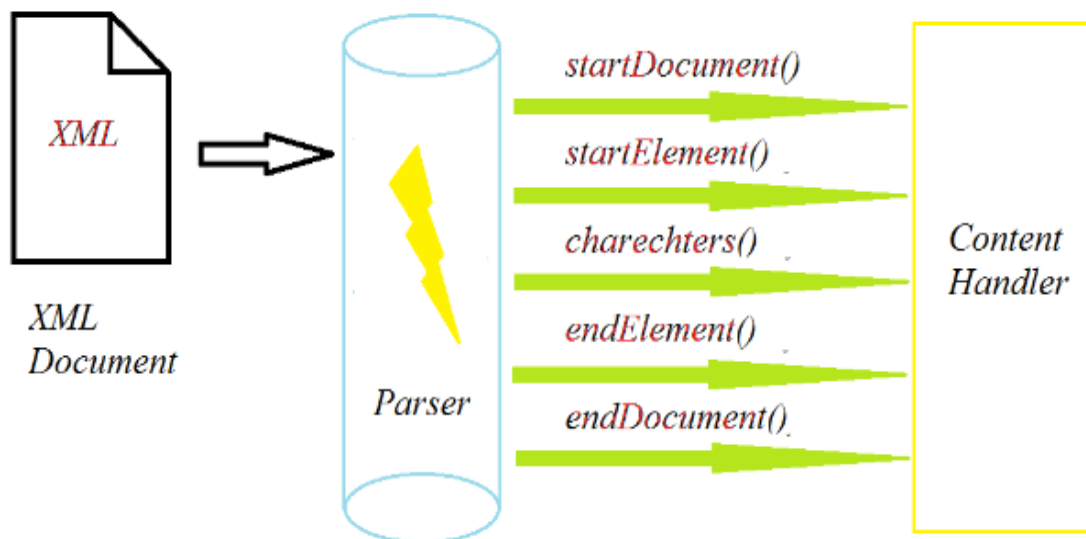


Figure 30 SAX Parser Workflow

A SAX parser is recommended with large XML files that do not require frequent modification, because SAX doesn't require loading the entire XML file in Java if the application needs to parse specific parts in the XML document. Also, the memory

consumption is not affected by XML document size since the objects associated with the triggered events could be destroyed on a regular basis. Moreover, it supports partial data access before the XML document parsing is done [42]. However, DOM is more powerful. It is more suitable for complex and frequent updates despite its high memory consumption. Since a DOM parser uses a hierarchical tree structure for the data representation of the XML document, it is much easier to add, modify or delete nodes in the DOM tree. This could be achieved by pointer maneuvers between the tree nodes for fast editing, insertion and deletion. On the other hand, SAX is more useful with applications with limited memory that require simple or rare modifications [15].

Chapter 3 An Intelligent Tutoring System Framework Empowered with Knowledge Representation

3.1 Cyberlearning Challenges

Historically in the past, e-learning systems technologies were developed using HTML and CGI, which resulted in low interactivity and poor content renewal. The development of new emerging web and cloud technology development supports dynamic generation of Web documents like Java Server Pages (JSP), Active Server Pages (ASP), and Hypertext Preprocessor (PHP), which led to the development of improved interactive systems. This brings us to the future of Cyberlearning, which is rapidly changing as new technologies emerge. This has led to a shift of how this effect's Cyberlearning content delivery to meet student's needs. If the education system can make the transition to a knowledge graph driven learning environment, which can be personalized to adapt to student needs instead of delivering a fixed content structure, then the system can start to be far more intelligent than it currently is and this can improve students learning outcome. For example, if student A spends her summers at camp and playing tennis, then this is a different skill set than student B who spends his summer working in a coffee shop. But when school starts in the fall both student A and B who attend the same Cyberlearning class will get the

same learning content regardless of skill and knowledge which is a fixed content learning environment.

In chapter one we discuss some of the proposed challenges for this research to improve Cyberlearning tutoring system to guide and personalize student's needs.

3.1.1 Overwhelming Study Resources And Lack Of Personalized Guidance

During any given semester students are presented with online learning materials for their selected courses work. These online learning materials can be overwhelming due to the lack of personalization and no direct guidance from students' instructor. These shortcomings of traditional e-learning systems lack the necessary intelligence to improve how to guide learners according to their learning level and learning order. In other words, today's e-learning systems lack the necessary semantic knowledge representation and technology architecture, which makes learners, feel unconfident and suspicious. Some of the challenges of this research is how to represent personalizing same student course content for different learning levels and what should be the learning order for these learning levels and resources? Also, supporting and improving learning object concepts for interoperability, accessibility and reusability. Some use case studies have been generated to demonstrate the need to facilitate learning object reuse thru the process of student personalize guidance.

3.1.2 Lack Of Instructor Experience

In a traditional classroom setting an instructor is presented to the student to teach a course. The face-to-face interaction allows student to engage with the instructor in order to gain his/her experience and feedback on the course that is being taught. But with an online learning system the student lacks the experience and feedback of instructor. This can result in disorientation on what to learn first, where to find other resources, and waiting to get feedback from the instructor. The instructor's experience is a very important concept to capture in a tutoring system. Intelligent tutoring systems are needed to improve the Cyberlearning quality. One of the major difficulties is knowledge representation. The current industry standard is to use Web Ontology Language (OWL) for representing knowledge structure. But OWL only supports one "first-class" relation, "is-a", between the concepts, and different knowledge areas usually need different custom relations to describe the relations among the concepts. For example "part-of" and time dependency are important relations to represent most engineering knowledge bodies. OWL has to use object properties to emulate such custom relations, leading to awkward knowledge representation hard for domain experts to code, validate and use such knowledge bases. This research uses Pace University's extension to OWL, named Knowledge Graph, to support knowledge representation with custom relations. The instructors can use Pace University extended Protégé IDE to declare and apply custom relations in a single document. The instructors teaching experience is also coded in the Knowledge Graph to better support custom learning order by students with different backgrounds. The instructor teaching experience is the center of intelligent tutoring system, used in organizing, managing and implementing. This dissertation provides a

framework which empowers the knowledge representation that will adapt to students learning level and learning order in order to analysis the current state of the students, and give teaching strategies in order to implement the most effective teachings methods, so that it can supervise and evaluate the results of learning, and achieve individualized guidance.

3.1.3 Lack Of Assessment Based Study

Most mainstream learning systems have relied on predetermined and often fixed-path delivery of content. Such systems lack agility in adapting to learners' mastery states and are thereby limited in their ability to tailor learning experiences to individual learners. An adaptive, "intelligent" learning system needs an accurate model of the learner, a model of the knowledge domain, and a capability that can evaluate the differences between the two models and identify or devise (on-demand and in real time) instructional strategies that will achieve desired instructional outcomes. In this dissertation a prototype of a knowledge-driven tutoring system was designed and implemented to illustrate how the Knowledge Graph supports integrated assessments, using assessment results to custom student learning order or material, and let the students freely navigate in the knowledge space from general to specific or the opposite, and following various custom relations. A web technology tutorial is used to validate the design and effectiveness of this approach.

3.2 Intelligent Tutoring System Solution Framework Overview

3.2.1 Concepts and Proposed Solution Framework

In this section, we present the main concepts used in our proposed model and explain the mechanism via an organized statement. We define our main research problem in this section. The problem definition is a Linking Learning Objects Problem: Inputs include a group of non-interchangeable LOs that are not in consistent sizes, are not written in consistent languages. The output is a link LOs. The proposed problem is finding out the method of linking all learning objects by using ontology-based semantic solution. LOs are not interchangeable because the objects are not of consistent sizes, nor are they written in consistent languages. LOs are interactive, text, html, flash, Java, and JavaScript. While others LOs are in audio, multimedia, PowerPoint, and movies. The proposed problem is finding out a way to link the Lobs according to custom relations. Moreover, LOs lack consistent classification schemes. There have been initiatives that focus on the need to develop a classification scheme, to catalogue the objects so that individuals can retrieve and organize them. Another issue is learning levels are not resolved in a consistent way, nor are the points of authorship, copyright, language of object, and many other issues. The current issues with LO Metadata (LOM), is that it is an XML-based development, which emphasizes on syntax and format and is not semantically driven by knowledge representation. LOM does have the advantage of data transformation and digital libraries, but lacks the semantic metadata to provide reasoning and inference. XML-based markup language is limited in this approach because it causes compatibility problems with existing data applications. LOs has its advantages and disadvantages; however. The lack of technical experience may deter some

technologically inexperienced staff to create web-based LOs; they would be more comfortable using software, which they are familiar with. It can also be time consuming to create a high quality web-based LO and subsequently the workload of the author will be increased [5]. Effective pedagogy is lost if the author does not have a clear educational goal when designing a LO. Also, since the definitions and the size of the Los are fairly unclear, it is not certain how much content is contained in one single LO and it is difficult to construct LOs which are independent of each other and with no context in mind [5]. Access to digital materials is now very easy; however, permission must be obtained and correct attribution must be provided if you are using learning materials in your LO which have been created by someone else [5]. The requirement of LOs to be reusable has meant that many authors have had to reformat all their existing learning content before it can be reused. For example, a PDF user manual for a piece of software is required to be deconstructed into several smaller components, and converted into XML before storing it into a database so that it can be reused in a LO system. A high percentage of current digital educational materials cannot be reused because they have not been decomposed and workload is therefore increased.

3.2.2 Knowledge Representation

This section aims to provide a knowledge representation framework solution to personalized learning. The knowledge representation framework is based on the idea that adaptiveness is matched with the users profile knowledge learning level and learning objects of the course concept learning order. Therefore we need to model the learner and

the course concepts. The adaptive learning system applies a mechanism to select the appropriate content type resource to the learner.

Internet-related technology such as cloud-based solutions has dramatically revolutionized the way people communicate and it has become a changing life style [38][37][39][43]. Educational systems around the world are trying to adapt to this change. This change gives researchers the opportunity to explore how online digital content is enhanced, designed and implemented to support personalized learning. Web based LOs are designed in a contextualized manner to promote reusability in various diverse learning contexts [28]. However, educators are required to deconstruct learning web based objects into components in order to rebuild the materials according to their individual learning needs. Difficulties and the associated financial expenses arise when decomposing the learning web based objects because it is difficult to index extremely decontextualized materials for human discovery and use and also computers are unable to make meaning of these materials [30]. An adaptive student navigation support system is needed to guide students and generate available learning resource objects, and assist in assessment base learning knowledge to track and compute student's performance level and lead students to a recommended learning path [3]. In e-Learning applications metadata is also fundamental for describing online learning materials and among other information, it can capture the subjects or knowledge domains associated with each document [18]. This information is needed for customizing learning sessions, but it is not sufficient. For locating relevant content, the system needs to know how the subjects are interrelated in an abstract knowledge space and follow the associations to suggest links to the student. Moreover,

knowledge is the collection of raw facts and rules, which is the information about objects, attributes and relations between objects, situations, events, states and time, causes and effects [36]. Representing knowledge for different learning levels in a web-based tutoring system such as beginner, intermediate and expert are often disaggregated into modules, sections, or chapters. Therefore, knowledge representation provides the way to represent all the above-defined things. The main characteristic of knowledge is that it is hard to characterize such that it is difficult for us to represent it properly. We have to represent the knowledge in such way that we can infer new knowledge also. Recently new concept of knowledge graph is introduced by structuring knowledge into a graph [7]. There are various techniques available to represent the knowledge. Part-whole are important relations and plays a key role in many application domains, but receive less attention than “IS-A” subclass / subsumption relations [4], [24]. The part-whole relations learning model offers a helpful framework for developing the semantic relations and the compositional knowledge structure to help improve learner’s navigational learning experience. This conceptualization defines the knowledge graph representation and the learner’s activities. In Figure 31, the PaceJena Tutoring System, learners are given the option to choose their initial orientation by selecting the type of learner he/she is when creating their user accounts. Learners are given the option of “Beginner” or “Intermediate”. The role that the learner will encounter is the presentation of learning objects in the form of its whole to its parts [25]. Learners are given the option to navigate the knowledge graph from generalized broad concepts to a specific narrow concepts and these concepts can be also bidirectional, such that a specific narrow concepts can return to a generalized broader concept. In some cases the learner may not comprehend the

concept of the whole concept without an understanding and proficiency of the individual parts concepts, so the return to the whole concepts allows the learner a second change to arrive at a better understanding of the domains learning concepts for its whole. Some learners are not “Beginners”. Some are “Intermediate” learners. Intermediate learners are more advanced than a “Beginner”, but not yet an expert. These learners have a different starting point in the whole concepts to its parts concepts. The learner’s level of learning does matter because from a personalization perspective they have different learning starting point but both learning level roles will achieve the same goal [4]. This becomes a problem when trying to represent the part-whole relations when using OWL because it does not provide any built-in primitives for part-whole relations as it does for the subclass relation, but contains sufficient expressive power to capture most but not all of the common cases.

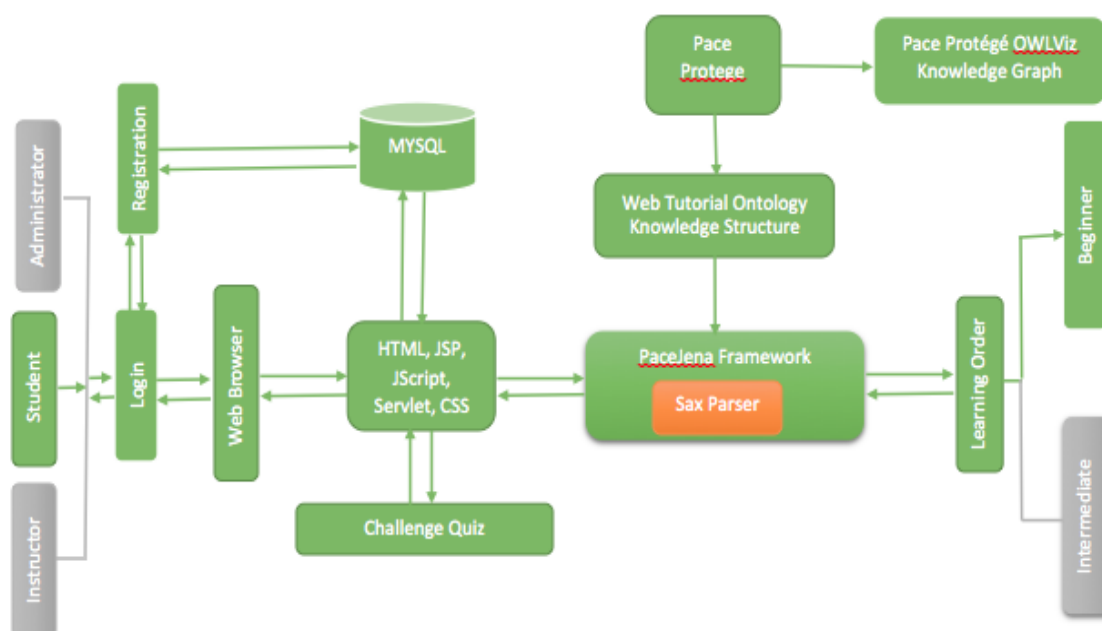


Figure 31 Solution Framework of PaceJena Tutoring System

3.2.3 Instructor Experience Representation

The instructor is a person who organizes the subject to be taught in several lessons and each lesson includes several topics. Instructor's expertise includes domain specific knowledge like topics in a lesson, examples, exercises and domain independent knowledge which as general pedagogical rules, adaptation to contextual constraints such as managing the last lesson.

In this dissertation we use learning objects to represent our web tutorial domain learning concepts. The learning objects include the instructor's expertise, for representing these domain concepts by introducing adaptive personalized learning levels, adaptive personalized learning orders, dynamic links with clear concept definitions, graphical displays as well as navigational interactions. Another part of the teaching expertise is the prototype model that is being proposed to describe the domain, the criteria used to dynamically build the learning path for each user, knowledge graph navigation and the adaptive learning assessment process. A learning order for each learning level is presented to the user in a menu-driven knowledge graph navigational interface. The instructor can easily enrich the given domain learning object content in the tutorial system environment.

The explicit knowledge representation is the main feature of our tutoring system. The instructor expert module containing the explicit knowledge provides the domain intelligence for the system. In the domain model it is necessary to decide not only what

knowledge to include in the instructor expert model but also how it can be articulated, codified, accessed and verbalized. The PaceJena Tutoring System is representing and using a knowledge graph domain, which extends OWL and represents new custom relations.

3.2.4 Knowledge Graph Navigation

Knowledge navigation in the PaceJena Tutoring System is used to describe learner navigation among digital learning resources, guided by a knowledge structure graph of the “Web Tutorial” domain concepts. The motivation of the PaceJena Tutoring System is to provide learners with an improved navigational process for personalized learning using learning objects. The conceptualization knowledge graph means that an explicit representation of concepts and relations are involved to support the navigational activities, such as navigating to reading a document, viewing a video, or listening to audio, and much more. Learners are navigating the knowledge structure graph from top level to next level and from next level to previous level. This behavior allows the user to move from one resource to another. This research uses software engineering methods and tools to support knowledge graph engineering activities that are concerned with the knowledge graph development process, the knowledge graph life cycle, as well as the methodologies. The knowledge graph development for this research was done using specification and conceptualization. The specification phase is to acquire informal knowledge about the domain and the conceptualization phase is to organize and structure this knowledge using external representations that are independent of the implementation languages and environments. This section shows how we have adapted different

knowledge graph development methodologies to define the specification and conceptualization phases. This also shows how different software engineering techniques were used to define different representations during these phases.

3.2.5 Integrated Assessment

The problem of assessing a student's knowledge could be regarded as a process of estimating what the student actually knows about the subject being taught. It is usually performed through a set of quizzes generated by a tutor and solved by students. An assessment provides very detailed information about a student's competence at all points during instruction. Assessments are used to guide the selection of the next instructional actions.

This proposed integrated assessment solution uses multiple-choice quizzes to assess each concept in our knowledge graph. During the assessment process students are to answer 10 questions. Upon completing the assessment process the system will evaluate the student's answers and recommend to the student the next steps in his / her learning process. The conceptualization knowledge graph and navigation supports the recommended outcome of student's next learning order.

3.2.6 Web Tutorial Ontology

The current e-learning standards are for learning management systems and not for

integration with semantic web-based application systems. The most recent standard for IEEE LOM is expressed in Resource Description Framework (RDF). RDF alone does not provide common schema that helps to describe the resource classes and represent the types of relationships between resources. A specification with more facilities than those found in RDF to express semantics flexibility is needed. A semantic web-based intelligent tutoring system solution can help solve these problems.

PaceJena was used to demonstrate how the Web Tutorial Knowledge Graph and Learning Object could be the right extension to OWL and RDF based development, to emphasize semantics, knowledge reasoning and inference functions. These functions are necessary for computer interpretable descriptions, which are critical in the area of dynamic course decomposition, learning object mining, learning object reusability, and automatic course generation. This research will also identify important custom relations such as include, partOf, ref, implement, and implementedBy for effective knowledge representation to support adaptive learning based on web learning objects along with using a prototype to validate feasibility of run-time knowledge representation. PaceJena Tutoring System supports easy and flexible adaptive navigation access to related learning order resources, preventing users from getting lost and providing consistent navigation operation. The goal is to support adaptive knowledge navigation to help students find their paths in knowledge space by adapting the knowledge structure presentation and functionality to individual user's needs. Using the semantic solution aims to providing learners with a feasible on-demand tutoring systems, by which the study plan can be generated. Moreover, the adaptive methodology is to assess the student's knowledge by having them

take a challenge quiz at the end of learning a LO. The challenge quiz will determine whether the student was competent in achieving a pass grade of 70 or better. If so, then the system will adapt and recommend the student to continue to the next learning level. If the student was not competent in achieving a pass grade of 70, the system will adapt and navigate to the student's need by recommending the student to return to an alternative LO at the same learning level. In addition, the PaceJena tutoring system is a prototype to validate research. This type of application is needed to help students personalize their learning needs. Semantic web languages and technologies can help in achieving this goal. These technologies can help fulfill the lack of knowledge representation and functional requirements about specific education subjects. Most of these requirements should be flexible enough to accommodate new requirements. Most of the requirements are about supporting predefined learning paths for typical learners. This application should also be interoperable with other subject topics. The predefined learning paths for this system will be the beginner's level and the intermediate level. This can be expanded to many other types of learning level. As shown in Fig. 1 in Section I, we designed PaceJena tutoring system with four tiers. Detailed statements of these four tiers are given as follows:

1) Client Tier: This tier consists of JSP/Servlets and web services. The PaceJena framework is also a part of this architecture and is also built in Java, which is a more mature framework that supports the semantic web technologies. The client tier interacts with the educational tier and sends its requests to this tier.

2) Educational Tier: The educational tier consists of educational specific components, in which all clients interact with it. Education components are responsible for getting the requests from the clients and forward them to the semantic tier. Then the educational tier

forwards the results coming from the semantic tier to the clients.

3) Semantic Tier: The semantic tier consists of PaceJena, which is an alternative implementation of Jena for research. It provides an API to extract data from and write to RDF graphs and provides an extensive Java library for helping developers develop code and includes a rule-based inference engine. All the educational components can interact with this model through the semantic web tier.

4) Data Tier: Data tier consists of RDF/XML graph/model. JENA's RDF model can be build either in memory or in a file or in a database.

In summary, our tutoring system aims to assist learners to gain knowledge in a manner of knowledge graph based knowledge representation. The diverse concept levels are linked to learners' knowledge levels, which can enable learners to accomplish learning activities from a lower level to a higher level.

3.2.7 Evaluation

The evaluation process will consist of validating the systems effectiveness through the prototype tutoring system.

3.2.8 Documentation

The documentation process and guidelines in developing the ontology was to use the Methontology Framework.

3.2.9 Conclusion

The purpose for executing this educational ontology Methontology Framework was to help improve the process of developing and evaluating this ontology and to follow some guidelines to document the ontology process. This proposed process called Methontology framework describes each phase and the techniques in developing a prototype for this research.

3.3 Knowledge Representation of Intelligent Tutoring System Solution

3.3.1 Knowledge Representation

3.3.1.1 Knowledge Representation Overview

This section aims to provide a knowledge representation framework solution to personalize learning. The knowledge representation framework is based on the idea that the adaptiveness is matched with the users profile knowledge learning level and learning objects of the course concept learning order. Therefore we need to model the learner and the course concepts. The adaptive learning system applies a mechanism to select the appropriate content type resource to the learner. We use ontologies to describe learning objects and the learner state, and define pedagogical recommendation axioms that specify which learning objects are best suited for a particular learner in a specific situation. The web tutorial ontology represents individual learners, which we have optimized by means of guidance through learning pathways of a particular order in which learning objects have to be studied. We use OWL to model our learning pathways as structured sequences. Ontologies provide a uniform model for the different aspects of a learning

process that can be conceptualized as the navigation through a network of available LOs, thus requiring techniques for dynamic and adaptive sequencing of LOs. In e-Learning, Learning Objects are organized into sequences that describe an optimal navigational path towards a learning goal. But at present, there is no support for defining sequences in OWL as would be needed for reasoning over learning pathways. Another limitation is OWL 2 (Web Ontology Language) is that only binary relations between classes can be represented.

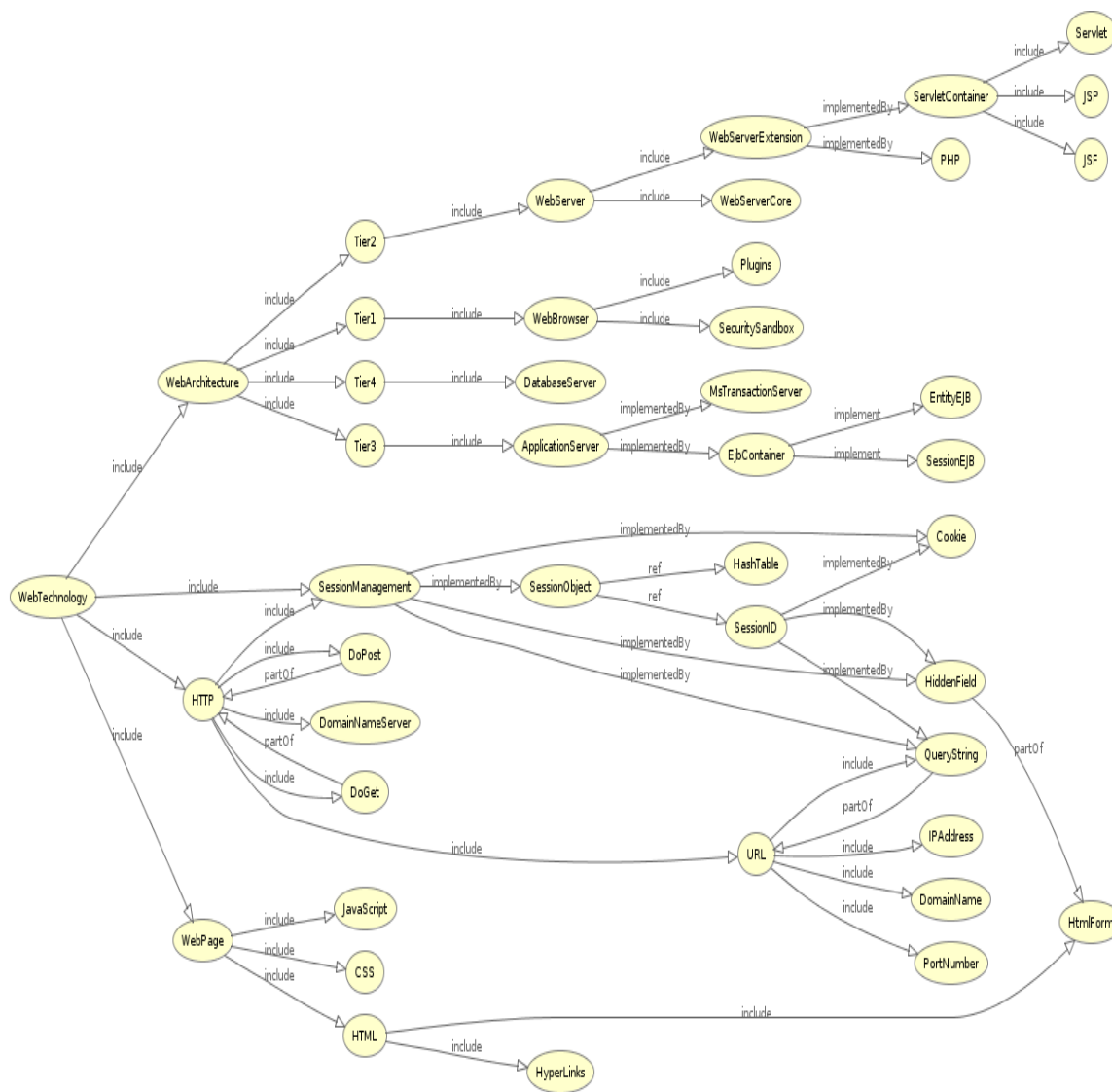


Figure 32 Knowledge Graph Representation of Extended and Custom Relations

We introduce a knowledge graph to support our knowledge structure. There are various techniques available to represent our knowledge structure. Part-whole relations which has been mentioned in a previous chapter plays an important role for modeling and developing semantic relations for compositional knowledge structure to support and improve learner's navigational learning experience. This conceptualization defines the knowledge graph representation and the learner's activities. In our domain ontology we have encoded the learning level and order for representing initial personalize learning knowledge state. The learning concept's for each topic contains a quiz, which will assess and recommend the learners next knowledge state.

3.3.1.2 Linking Learning Objects

In this dissertation we research and examine to find a method of supporting and improving the linkage of learning objects by using knowledge graph solution. A link is a basic relationship among objects and shows that the objects are somehow related. Learning objects are interactive learning resources such as text, html, flash, java, and java script, while others are audio, multimedia, PowerPoint, and movies. The current issue with learning objects is that it is not semantically driven by knowledge representation because it is represented as an XML-based language. XML-based markup language is limited, because it lacks the semantic metadata to provide reasoning and inference. Learning objects also lack consistent classification schemes and learning levels are not resolved in a consistent way. Another issue with learning object is the lack of reusability due to the lack of decomposition.

3.3.1.3 Ontology Representation

Ontologies represent and support relationships among concepts providing them meaning. Three types of relationship are considered in this work: generalization/Specialization, which are powerful abstractions for sharing similarities among classes while preserving their differences; composition also known as aggregation (part-whole and part-of relationships), in which classes representing the components are associated to the class representing the entire assembly; and binary relationships, representing any other type of relationship that connects two concepts. We also extend OWL and introduce new custom relations such as the “include”, “implement” and “implementBy”. The URLs are then mapped into the ontology concepts according to the tomcat web server services with the Spring Web MVC structure, along with the knowledge graph and the MYSQL database structures. The MVC structure is a well-known pattern suggests the separation of domain models from user interface and control logic. The separation of non-visual parts from visual components makes it potentially easier to reuse and share domain models for other applications and target platforms. The WebTutorial.owl ontology for this dissertation has been constructed to contain both the domain ontology with topics to be taught in a course along with the decoupled encoded learning level and order for students. The most straight forward way in OWL to separate the classes that represent the domain model from the instances that represent the topics being taught in a particular course is to use the subClassOf property to model the relationships between the classes in the abstract domain model and the instanceOf property to connect the concrete course topics to the classes in the abstract domain model. Structuring domain models using these properties provides only

generalization and specialization relationships in general taxonomy and type information for the topic instances of the course. The domain model is only representing the “is-a” relationship. Domain models should fully represent all of the topics in the domain so that they can be reused between different courses that teach the domain they represent. What is needed is a richer representation of the domain concepts and its relationship between each concept. We go beyond the “is-a” and inheritance relationship and extend OWL to support the “part-of” relations. This dissertation work also support recommendation for learning order, which goes beyond ontology.

3.3.1.4 Part-Whole Relations

Part-whole representation of educational materials is a very important and challenging issue among people who want to use ontology. Currently OWL does not provide any built-in primitives to support part-whole relationships. Part-whole plays a key role in many application domains and is a central structuring principle in artifact design. For example, it is used to describe ships and cars or when used in chemistry and medicine to describe structure of a substance or structure of anatomy. Traditionally, part-whole receives much less attention than subclass / subsumption relation. We can introduce new relations to support this type of hierarchy. An important and common feature of part-whole relationship is to support transitivity. If A is part of B and B is part of C, then A is part of C. If we define a property as transitive property then the reasoned can conclude that A is part of C. OWL also supports inverse relations, which we can define `hasPart` as the inverse of `partOf`. For example, for any two individuals A and B, if A is part of B, then B `hasPart` A. Unfortunately, reasoning cannot support both `partOf` and `hasPart`, and

as a result it is prefer to use `partOf`, since most queries are performed using this property. It is important to structure and represent part-whole relationships as a hierarchy in which objects at one level are composed of inter-related objects at the next level down. But also, using this knowledge structure and representation for linking learning objects for reusability and classification.

3.3.2 Capturing Instructor Experience in Learning Orders

The knowledge representation in our domain ontology and also in PaceJena captures the instructor experience to support the learning order and also to personalize learning experience for this research. The domain ontology for this research is called Web Tutorial Ontology and the course topic is on “Introduction to Web Technology”. In figure 33 and 34 is a snapshot of both learning orders for “Beginner and Intermediate”, sequentially. The learning order for “Beginner” starts at a different starting point then the intermediate. The “Beginner” concepts will start with “WebPage”, “HTTP”, “SessionManagement”, and WebArchitecture”.

3.3.2.1 Instructors Representation of Learning Order

The Web Tutorial Ontology captures the instructor’s representation of how the learning order needs to be presented to the different types of learner. In this section we will dissect some of the knowledge representational features to describe how we can capture this learning order. The class `owl:NamedIndividual` is new to OWL 2, it is used globally in a domain ontology. Individuals are facts about their class membership and their property

values. The declaration `owl:NamedIndividual` is extended using declaration called `&pace;LearningOrder`. In figure 33 and 34 we declare two sets of `&pace;LearningOrder`. One is for a “Beginner” and the other for “Intermediate” each learning order as a new custom relations called `level` which is associated with a number, such that `<pace:level>0</pace:level>` is learning level for a “Beginner” and `<pace:level>1</pace:level>` is learning level for “Intermediate”. The namespace “pace” along with the relations “ref” creates facts about class membership and their relations according to the students learning order. This code is reusable and for any learning order and learning level and can be extended to add additional learning orders, along with any ontology learning concept. The most important part of the web tutorial ontology structure is the semantics between parent and child concepts and how these relations are extended and inferred from PaceJena and how to relate this information into the ontology learning order for each learning level. In order to enable rule-like knowledge representation and inferencing we extend OWL with PaceJena rules, functions and methods.

```

550
551 <!-- http://csis.pace.edu/semweb/webTutorial.owl#Beginner -->
552
553 <owl:NamedIndividual rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Beginner">
554   <rdf:type rdf:resource="&pace;LearningOrder"/>
555   <pace:level>0</pace:level>
556   <pace:name>Beginner</pace:name>
557   <pace:ref rdf:resource="http://csis.pace.edu/semweb/WebPage"/>
558   <pace:ref rdf:resource="http://csis.pace.edu/semweb/HTTP"/>
559   <pace:ref rdf:resource="http://csis.pace.edu/semweb/SessionManagement"/>
560   <pace:ref rdf:resource="http://csis.pace.edu/semweb/WebArchitecture"/>
561 </owl:NamedIndividual>
562

```

Figure 33 Learning Order for Beginner

```

564
565 <!-- http://csis.pace.edu/semweb/webTutorial.owl#Intermediate -->
566
567 <owl:NamedIndividual rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Intermediate">
568   <rdf:type rdf:resource="&pace;LearningOrder"/>
569   <pace:level>1</pace:level>
570   <pace:name>Intermediate</pace:name>
571   <pace:ref rdf:resource="http://csis.pace.edu/semweb/HTTP"/>
572   <pace:ref rdf:resource="http://csis.pace.edu/semweb/SessionManagement"/>
573   <pace:ref rdf:resource="http://csis.pace.edu/semweb/WebArchitecture"/>
574   <pace:ref rdf:resource="http://csis.pace.edu/semweb/WebPage"/>
575 </owl:NamedIndividual>
576

```

Figure 34 Learning Order for Intermediate

The student whom is a “Beginner” must start with the easier concepts first, students have the option to choose which concepts to start with first, and at any point in time a student may take the “Beginner” quiz. If student receives a passing score of 70 or above then student can proceed to the next learning concept. If students receive a failing score of 70 or less then student will be directed back to the previous learning concept.

3.3.3 Capturing Instructor’s Adaptive Learning Topics

IMS Learning Design specification (IMS LD) and IEEE Learning Object Metadata standard (LOM) enables one to specify learning designs and learning objects (LOs) targeted for different learning situations, based on different pedagogical theories, comprising different learning activities where students and teachers can play many roles, and carried out in diverse learning environments. However, they cannot enable more advanced learning processes, such as dynamic adaptation of content in accordance with the students’ objectives, preferences, learning styles, and knowledge levels. Further, if adaptation is to happen automatically information must be codified in an unambiguous

manner. Ontologies help increase the consistency and interoperability of metadata.

The Semantic Web community has already developed a number of different kinds of ontologies that can be integrated in an ontological framework in order to enable adaptive use of LOs inside a learning design.

We have investigated various tutoring principles used by tutors and educational software. As described earlier this work focuses on several aspects of teaching and learning using LOs in a web-based environment. The approach taken in this dissertation proposed a framework that will improve the learning process support needed to provide the learner with recommended learning content to learn from and the recommended learning path. The topics are represented in a hierarchical and sequential order for each learning level. This is represented once the student is logged into the tutoring system. These topics are all composed of top level generalized topics and are decomposed to a more specific and specialized topics. Each topic contains its own quiz to assess the learners understanding at his/her current knowledge state. The instructor's knowledge graph representation helps support and guide students through the navigational knowledge space learning process. The instructor's integrated assessment strategy and techniques helps adjust student's learning strategy on what topics he/she should learn next. It is useful for students and tutor to ensure what is being learned and assessed is relevant to the topic. It is important to ensure that the terms being brought up during the learning process is relevant to the topic. This iterative process and knowledge graph representation helps the student's learning by introducing terms from topics that are more specific to the general topic. This also helps

the instructor understand the students learning progress and how the knowledge representational design can be improved by introducing other LOs for this framework.

3.3.4 Capturing Instructor's Integrated Assessment

This section captures the instructors integrated assessment feature's, which interacts with the knowledge graph to determine student's next recommended learning order. Quizzes will represent 10 multiple-choice random questions from the quiz database. In order to measure a student's learning performance students will have the option to select the challenge quiz. During the challenge quiz, students must answer all 10 questions. At the last question the system will present the submit button for students to submit their quiz for evaluation and feedback. The system will evaluate the students results according to an IF, THEN, ELSE statement. If quiz results are less than 70 then the system will recommend and direct student to a more specific or narrow concept from the previous concept. If greater then 70, then the student will continue to the next concept at the 1st concept level moving along the knowledge graph in a horizontal direction. If not then students will be directed to follow the previously mention less than 70 pattern. Quizzes will be recorded in the students profile for the instructor to evaluate student's performance and learning outcomes. Administrators and instructors have the option from their web-based application interface to add, save, edit, and delete quiz questions and answers from the quiz database.

3.3.5 Conceptualizaion of Web Tutorial Ontology

In this section we describe the most important terms which where elaborated according to the Methontology method. The top-down strategy approach was used to identify the core basic terms first which are the most generalized terms then to specify terms that are more specific at each level. The key terms are specified at the top level. The key terms where categorized according to the concept levels. In this research, the concept levels were defined by two relations, which were “include” and “partOf”. For example, in the statements “A includes B” and “B is part of A”, the object A is a more generalized concept than the object B. Then with these concepts as reference, the key term list was defined and graph to illustration the scope of knowledge representation shown in Figure 35 and Table 5 showed an example of partial Web Tutorial Ontology in representing the Introduction to Web Technology taxonomy. In the given example, we exhibited a variety of relations, such as “include”, “partOf”, “ref”, “implementedBy”, and “implement”. OWL was used to define these customized relations.

Table 5 Web Tutorial Ontology Key Terms

Web Technology	Text Document
Web Architecture	Tag Names
HTML	Links
Web Pages	Elements
Session Data Management	Attributes
HTTP Protocol	Cookies
URL	Hidden Fields
Web Browser	Query String
Application Server	Server Side Session
Computer Program	DNS
Web Server	HTTP Request
Database Server	HTTP Response
TCP IP	Ports
Markup Language	Domain Names
Resources	Web Resources

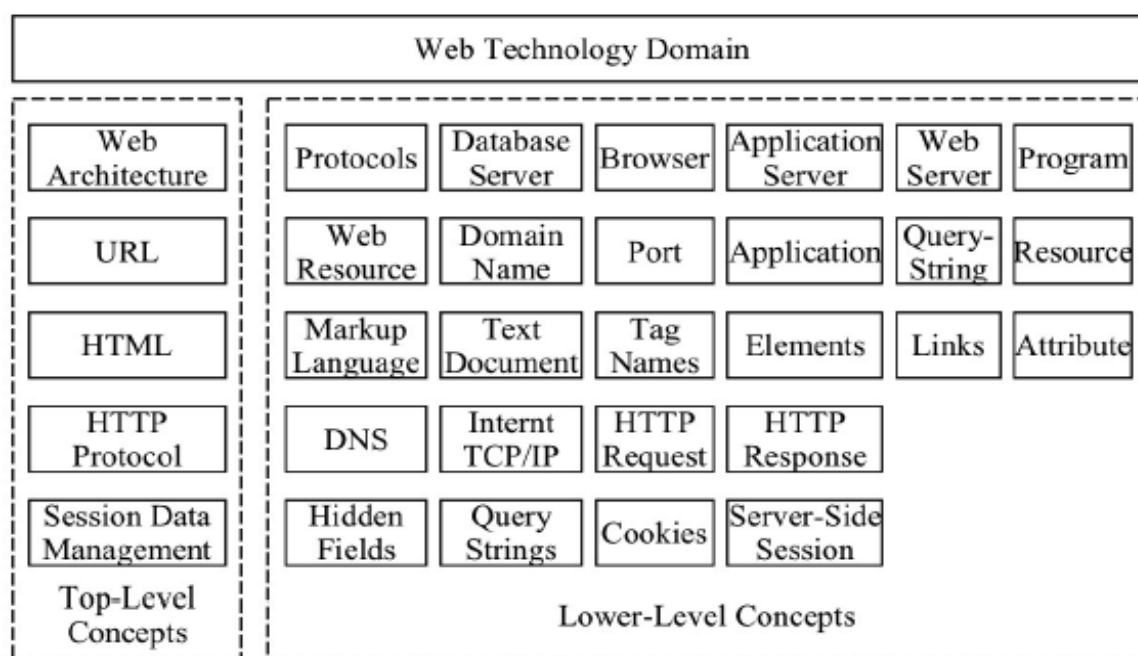


Figure 35 Web Tutorial Ontology Scope of Knowledge Representations

3.3.5.1 Tools for Representing Web Tutorial Ontology

The tools used to implement the Web Tutorial Ontology were Pace University's Pace Protégé version 5.0. Figure 36 and Figure 37 illustrate some of the features presented by Pace Protégé. Pace University's Pace Protégé version 5.0 was developed by Pace University's PhD computer science students in an effort to help students establish a way to identify their custom relations and also be able to use OWLViz to visualize their knowledge structure graph while building ontologies that use these custom relations. Pace Protégé has a relations tab which makes it easier to create custom relations. Currently Protégé does not provide an easy way to create custom relations nor does its add-on plugin, OWLViz provide a way to visually recognize these custom relations.

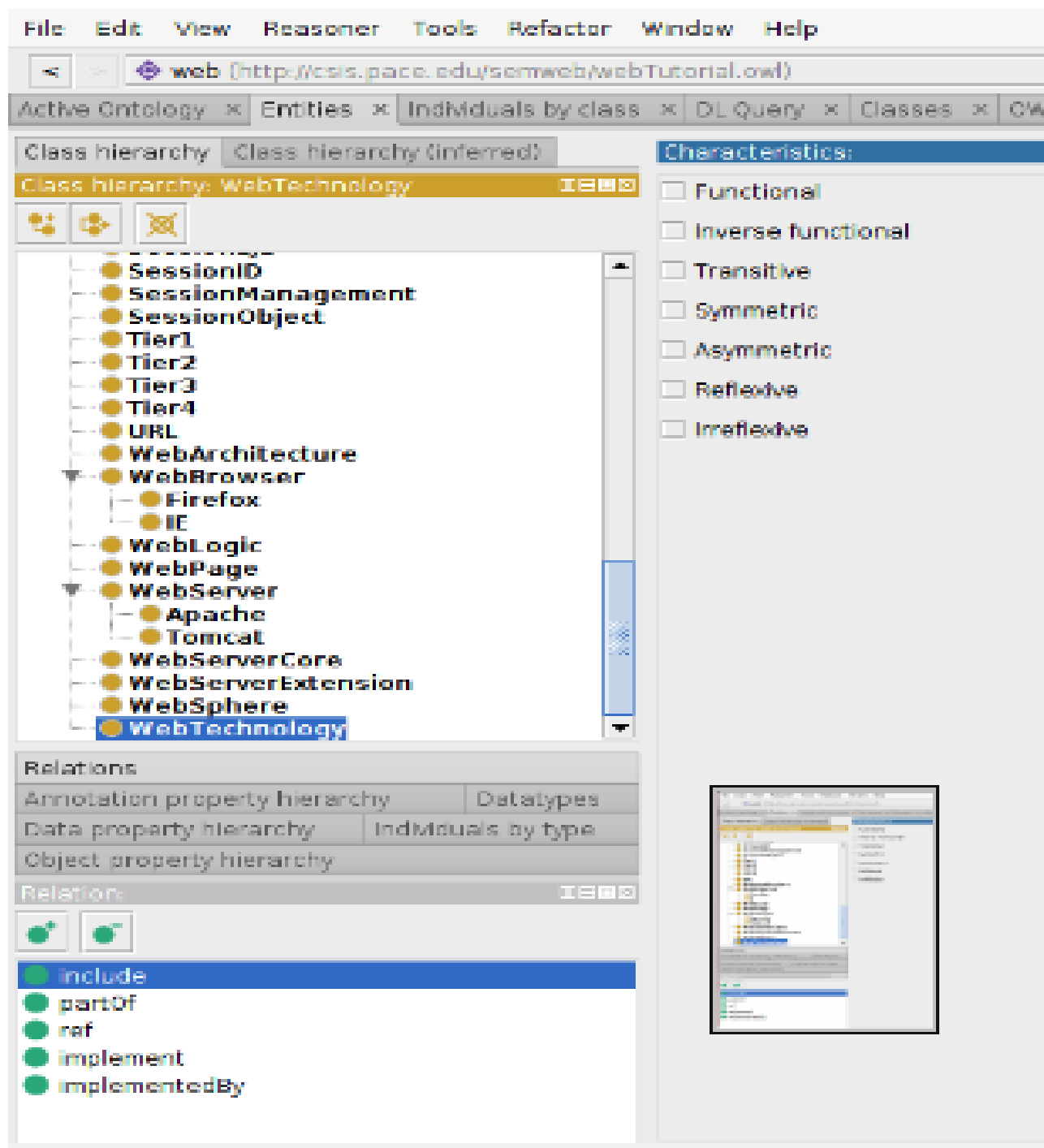


Figure 36 Pace Protege Version 5.0 - Adding Relation

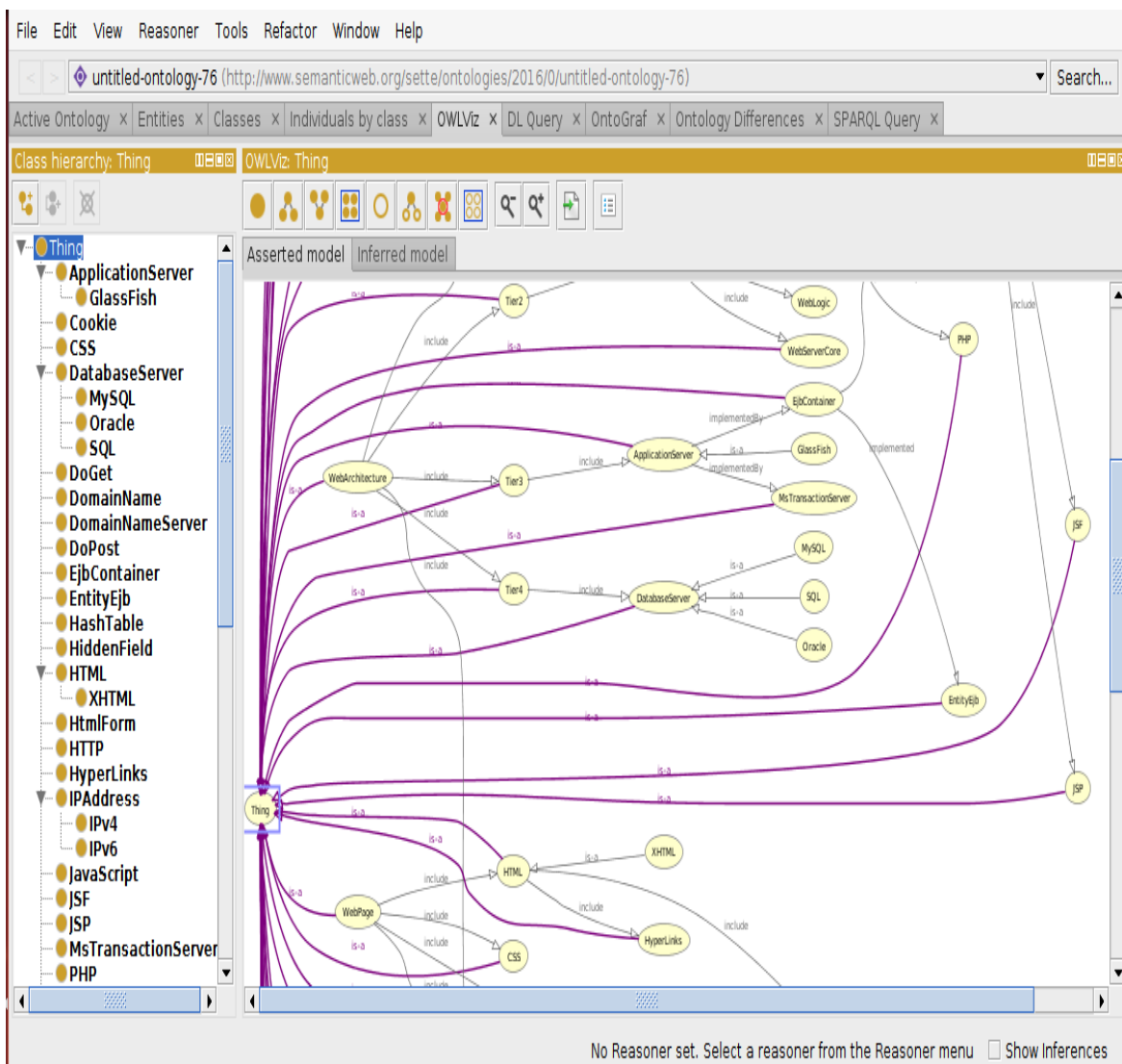


Figure 37 Pace Protege OWLVIZ Knowledge Graph with Custom Relations

After completing the ontology develop in Pace Protégé it is saved as a RDF/XML file. This file is then used in our PaceJena program to read the ontology and will be parsed by the SAX Parser, which is an event driven algorithm for parsing such files as RDF/XML. When the parser has retrieved the necessary information it is then delivered to the frontend application called PaceJena Tutoring System.

3.4 Tutoring System Life Cycle Based on Knowledge Representation

This dissertation proposes to deliver a tutoring system life cycle framework, which will provide direct personal feedback to students, without the intervention of human being. The definition of an intelligent tutoring system is an expert system, which can provide artificial intelligence without the intervention of a human being. The tutoring system that is presented in this dissertation contains four models or modules: the student module, the expert module, the assessment module, and the Instructor and Administrator module.

- The student module is where the student interacts with the interface, which provides the means for students to interact with the tutoring system, usually through a graphical user interface with a rich simulation of the domain ontology content the student is learning. The student module also contains the student knowledge of learning level and learning order.
- The expert module references an expert or domain model containing a description of the knowledge or functionalities and methods that represent expertise in the subject-matter domain the tutoring system is teaching.
- The assessment module references the quiz module, which contains questions and answers on the domain model containing the knowledge information. The assessment model will evaluate the student's knowledge level and will adaptively respond with feedback on the student's performance.
- The instructor and administrator module is where instructor has access to evaluate student's performance and also implement new course topics and quizzes. The administrator module is where the administrator can manage user accounts and maintain system.

An intelligent tutoring system with a semantic web environment and adaptive characteristics should utilize the opportunities that are provided by all the modules mentioned previously. Adaptive teaching and testing is essential to a student's learning environment. The adaptive characteristics provide the students with personal content delivery and assessment evaluation. The agent is equipped to assess the students profile and performance in order to adjust content delivery. The domain knowledge, expert module and student model is reusable. This is achieved by using and extending the OWL language and using Web Services to publish this knowledge information.

3.5 Tutoring System Architecture

3.5.1 Functional and System Design Requirements

PaceJena tutoring system is a prototype system design for this dissertation. This type of system application is needed to help students personalize their learning needs. Semantic web languages and technologies can help in achieving this goal. These technologies can help fulfill the lack of knowledge representation and functional requirements about our computer subject topic on "Introduction to Web Technology" ontology. Most of these requirements should be flexible enough to accommodate new requirements. Most of the requirements are about supporting predefined learning paths for typical learners. This application should also be interoperable with other subject topics. The predefined learning paths for this system will be students at the beginner's level and students at the intermediate level. This can be expanded to many other types of learning level but for this dissertation we will limit it to just two learning levels.

3.5.2 System Architecture

This four-tier architecture is implemented using semantic web technologies. Figure 38 illustrates a high-level architecture diagram of our system.

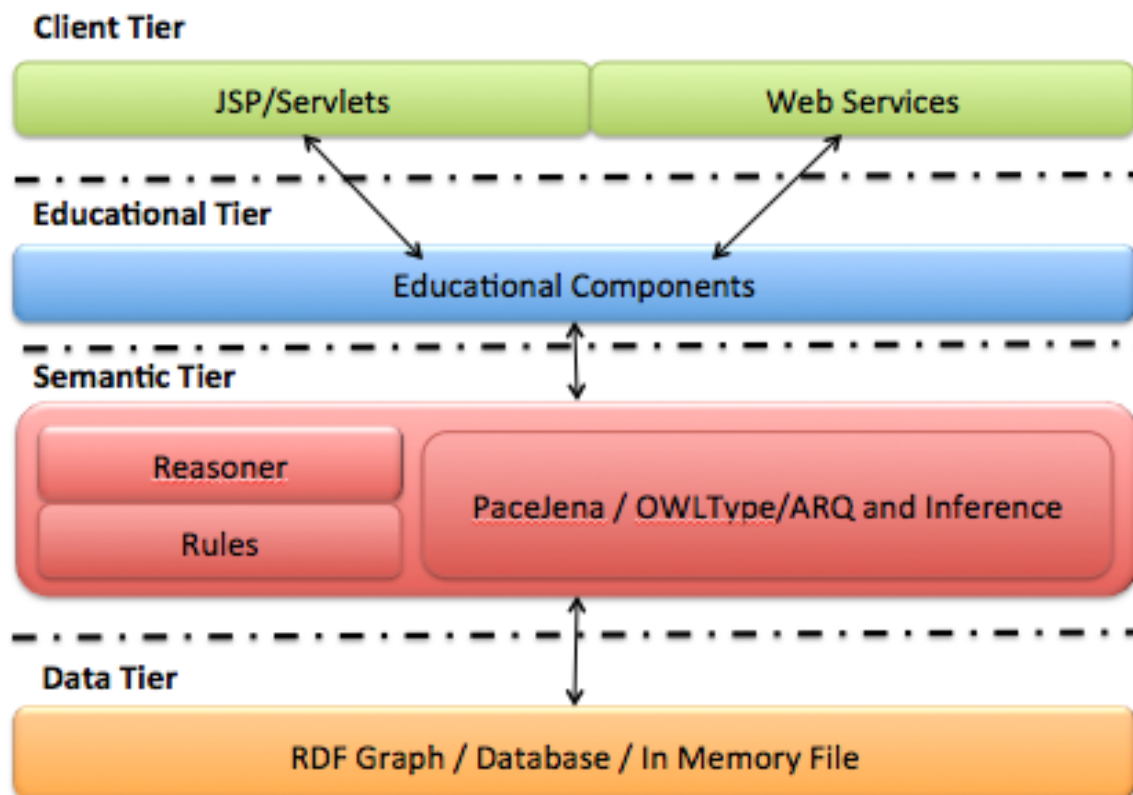


Figure 38 PaceJena Tutoring System Architecture

3.5.2.1 Client Tier

Client tier consists of JSP/Servlets and web services. JSP/Servlets based client is more java specific and there is strong support for this technology. The PaceJena framework is also a part of this architecture and is also built in Java, which is a more mature framework that supports the semantic web technologies. The client layer will interact with the educational layer and send its request to this layer.

3.5.2.2 Educational Tier

The educational tier consists of educational specific components like handlers. All clients will interact with this layer. Education components will be responsible to get the request from the clients and forward them to semantic tier. Then educational tier forwards the results coming from semantic tier to the clients.

3.5.2.3 Semantic Tier

The semantic tier will consist of PaceJena, which is an extension of Jena. The name Pace comes from Pace University, in Pleasantville, NY, which is the origin of this dissertation work. Jena was developed by HP Labs and is an Apache open source Semantic Web Framework for Java and is built on Semantic Web applications. It provides an API to extract data from and write to RDF graphs and provides an extensive Java library for helping developers develop code that handles RDF, RDFS, OWL, and SPARQL and includes a rule-based inference engine which is in line with W3C recommendations. Jena is the most stable framework that supports semantic web technologies. This is why we choose to extend this framework. All the educational components can interact with this model through the semantic web tier. It is important to understand this framework and how it plays a role in our knowledge representation for our ontology.

3.5.2.4 Data Tier

Data tier consists of RDF/XML knowledge graph / model. JENA's RDF model can be build either in memory or in a file or in a database.

- In Memory Model is a model in RAM. It is not possible to build a large in memory model due to limitations of RAM size.
- File Model is physical storage like hard disk.
- Database Model is a model built into a database. PaceJena just like Jena supports all major databases like Oracle, Microsoft SQL Server, MYSQL, PostgreSQL, etc... A database can store more data in a more efficient way.
- The database server is central and uses persistent storage of all data.

In summary, our tutoring system aims to assist learners to gain knowledge in a manner of ontology-based knowledge representation. The diverse concept levels are linked to learners' knowledge levels, which can enable learners to accomplish learning activities from a lower level to a higher level. A use case is represented in the following section.

3.6 Tutorial System Use Cases

In this section, we present the tutorial system main use case study that illustrates the application of the methodology, described in the previous section that will support the implementation of the proposed approach. The aim is to demonstrate that the proposed methodology can be realized for the representation of a semantic approach to an intelligent personal tutoring system in order to be used within the context of an educational system.

3.6.1 Main Use Case

In this section we will also discuss the instructional teaching use case, student use cases such as creating student accounts, how students learn a lesson, student assessment process, and how students navigate the knowledge graph. Figure 40 shows the screen capture of the PaceJena Tutoring System. A list of the most important terms was elaborated according to the 101 METHOD guide. A Top-down strategy was used. With this strategy, the core of basic terms is identified first and then they are specified and generalized if necessary. The list shown in Table 6 is a list of key terms, which does not include partial or total overlapping of concepts, synonyms, properties, relations and attributes. Next, we categorized these key terms according to the concept levels. In this study, the concept levels were defined by the following relations, which were “include”, “partOf”, “ref”, “implement”, and “implementedBy”. For example, in the statements “Web Technology” includes “Web Page”, “HTTP”, “Web Architecture”, and “Session Management”; and “Web Page”, “HTTP”, “Web Architecture”, and “Session Management” are part of “Web Technology”.

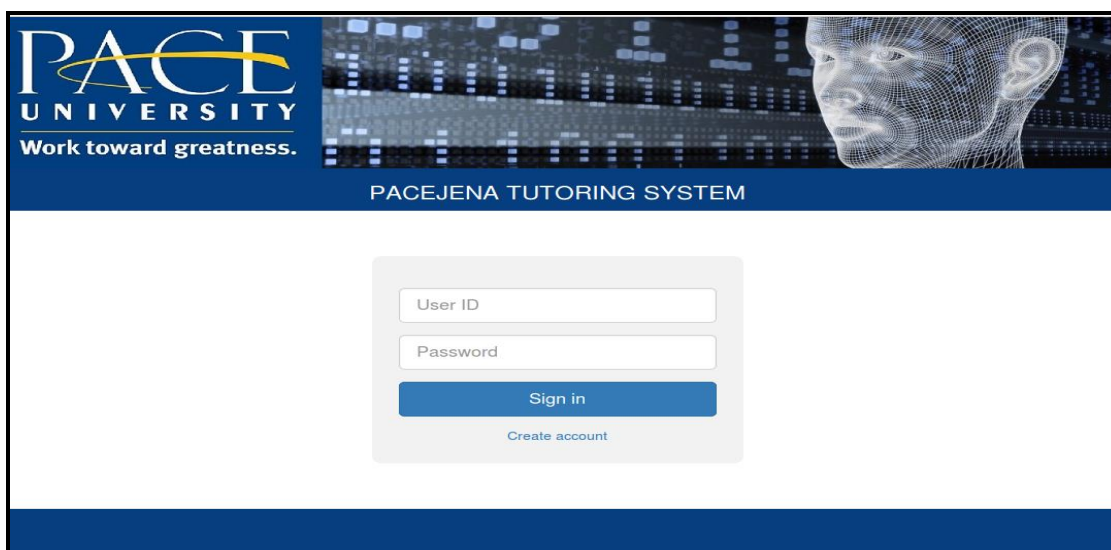


Figure 40 Screen Capture of the PaceJena Tutoring System

Table 6 Example of Key Terms

Web Technology	Text Document
Web Architecture	Tag Names
HTML	Links
Web Page	Elements
HTTP Protocol	Attributes
Session Data Management	Cookies
Web Browser	Hidden Fields
Computer Program	Query String
Web Server	Server Side Session
Application Server	DNS
Database Server	HTTP Request
Internet TCP/IP	HTTP Response
Markup Language	Ports
Application	Domain Names
Resources	Web Resources

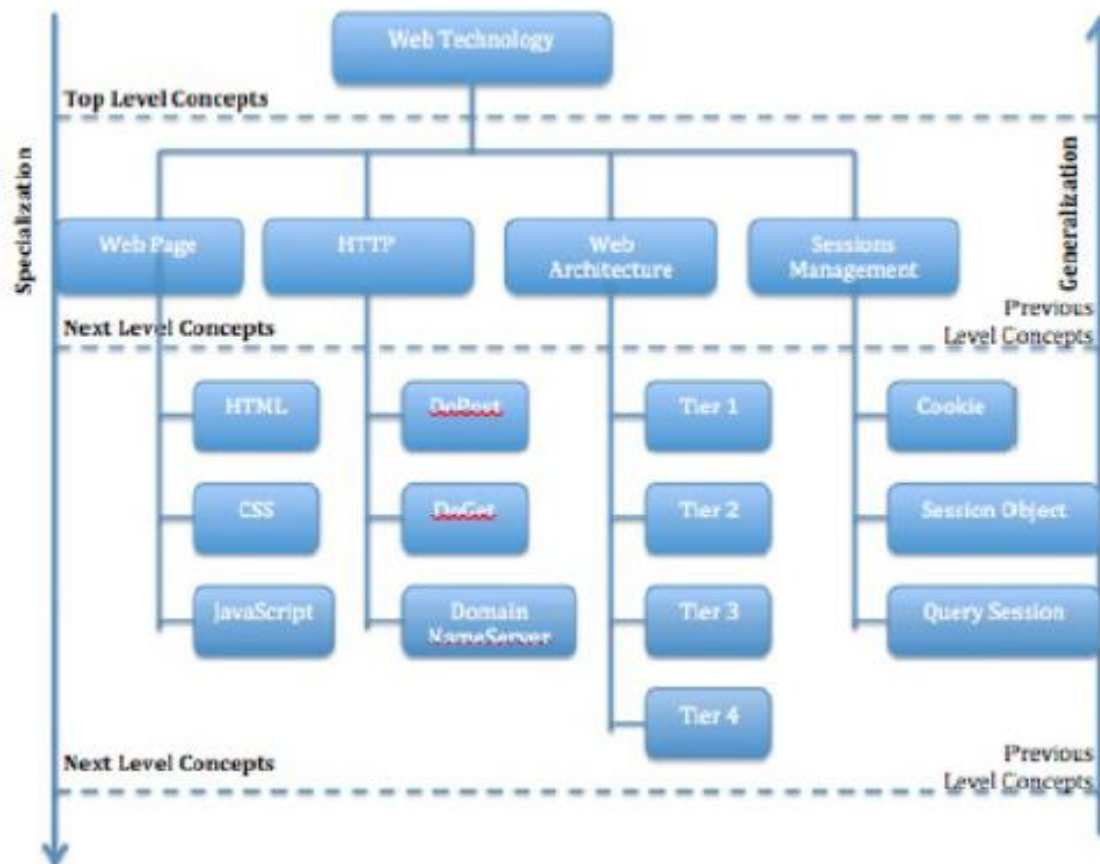


Figure 41 Example of Knowledge Graph Based on Custom Relation "partOf"

Figure 41 showed the knowledge graph related to “Web Technology”. In the given example, we exhibited a variety of custom relations, such as “include”, “implementedBy”, and “implement”. By using this technique to describe classes, an entire semantic knowledge system structure was eventually developed. The linked knowledge ontology had been assessed by our use case studies, which proved that our system was effective for personalized learning.

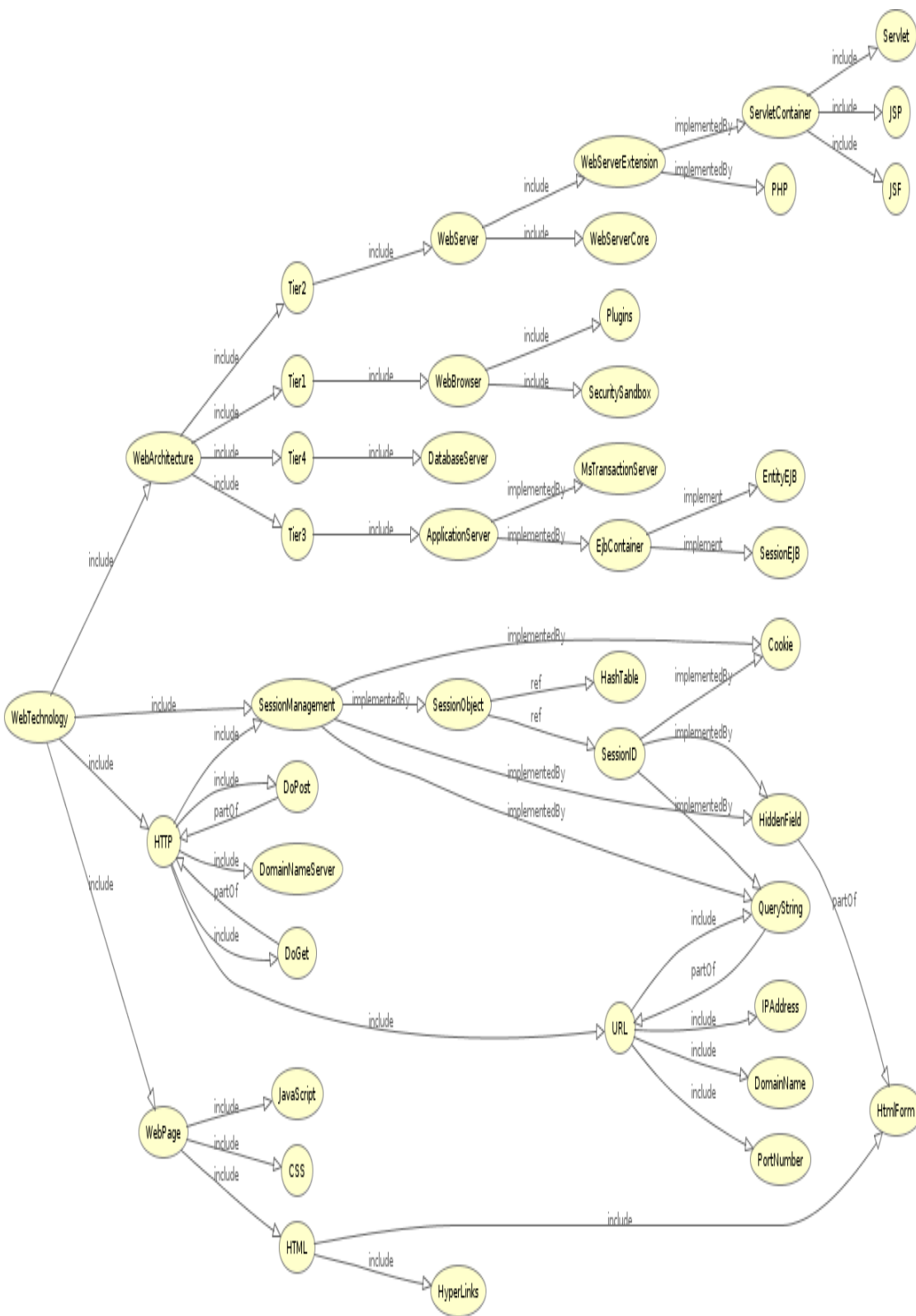


Figure 42 Sample Showing Ontology Taxonomy

3.6.2 Instructors Managing Course Content Use Case

Use Case Name	Instructors Managing Course Content Use Case
Primary Actor(s)	Instructor
Description	The Instructor / Teacher use case for editing and saving the course content.
Precondition	Instructor / Teacher must have a user account and admin permission level.
Post condition	
Basic Course of Action	
<p>Instructor / Teacher must login using Administrator user name and password. Instructor / Teacher can navigate to the concept of choice. Instructor / Teacher can click on the edit button to edit the concept content. Instructor / Teacher as completed the changes to the content then he/she can click on the close button to save the edited information in the database. End of use case.</p>	
Alternative Course of Action	



Figure 43 Instructor / Teacher Login Use Case

The screenshot displays the PACEJENA Tutoring System interface. At the top left is the PACE UNIVERSITY logo with the tagline "Work toward greatness." The main header area features a blue background with a wireframe head and data points. Below the header, the text "PACEJENA TUTORING SYSTEM" is centered. A navigation bar includes "Home", "Profile", and "Reports" on the left, and "Welcome, admin (Beginner)" with a "Logout" button on the right. The main content area is divided into two sections. On the left, a "Learning Order" sidebar lists "Tier4", "Tier3", "Tier2", "Tier1", "Introduction" (highlighted with a yellow background and a double arrow icon), and "Challenge Quiz" (in a green box). On the right, a text editor shows the text: "Website architecture is an approach to the design and planning of websites that involves technical, aesthetic and functional criteria. As in traditional architecture, the focus is on the user and on user requirements. This requires particular attention to web content, the business plan, usability, interaction design, information architecture and web design. It also includes deciding technology stack, site map and navigation system of website." Below this text is a blue "Edit" button.

Figure 44 Instructor / Teacher Editing Course Content Use Case

This screenshot shows the same PACEJENA Tutoring System interface as Figure 44, but in the saving phase. The layout is identical, including the PACE UNIVERSITY logo, header, navigation bar, and sidebar. The main content area now features a rich text editor. At the top of the editor is a toolbar with icons for Bold (B), Italic (I), Underline (U), Strikethrough (ABC), Text Color (S), Font Size (14), Bulleted List, Numbered List, Indent, Text Color (T), Background Color (X), Source Code (</>), and Print (M). The text content is the same as in Figure 44. Below the text is a blue "Close" button.

Figure 45 Instructor / Teacher Saving Course Content Use Case

The instructional teaching use case improves the support of adopting different domain topics and the reusability of learning object resources and also supports reusability of student assessment process. This is further explained in the section called “Guidelines for Adopting a New Topic”. The instructional teacher is task with creating the conceptual vocabulary structure for the course by using the Pace Protégé tool in Figure 46.

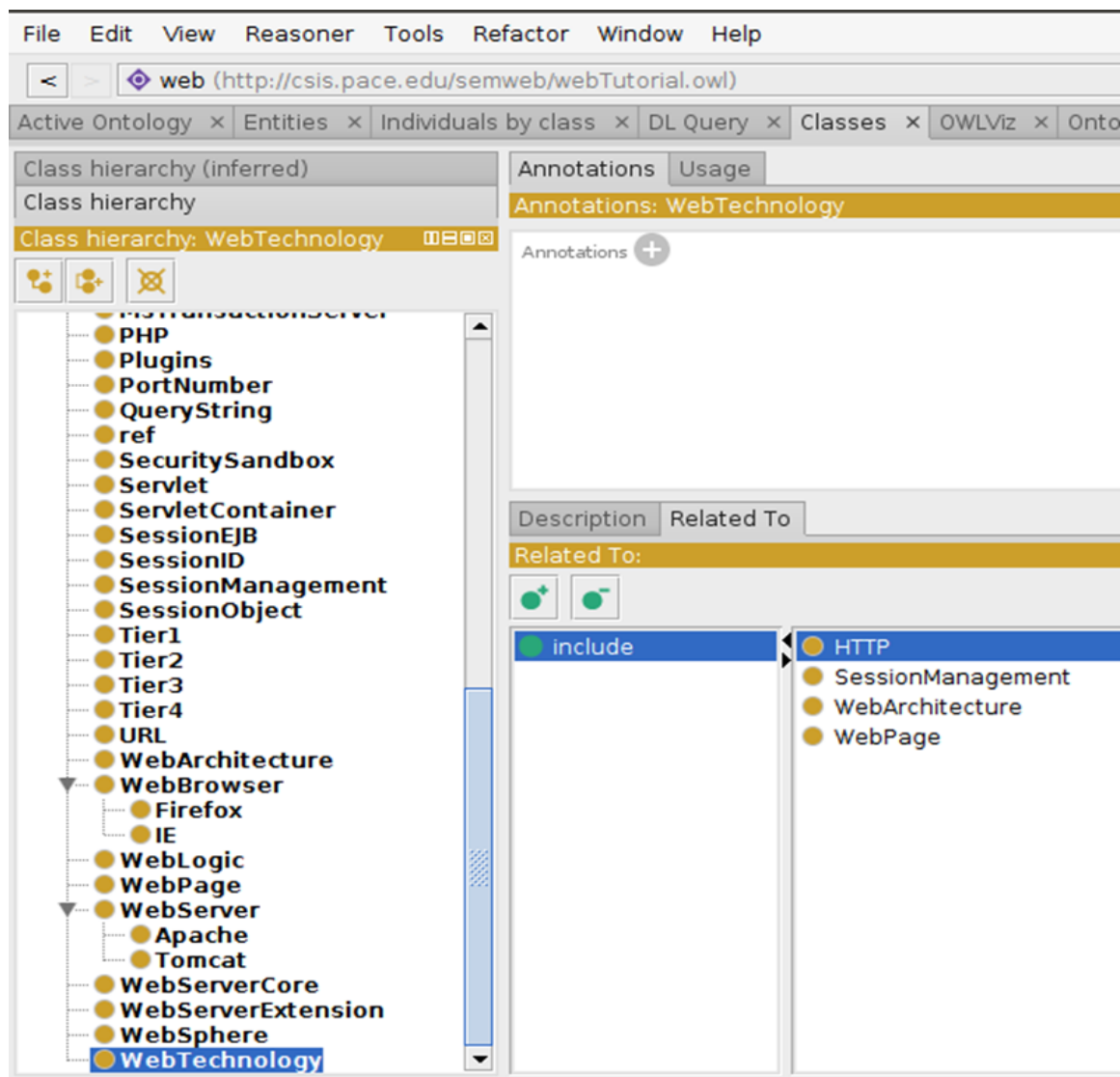


Figure 46 A Snapshot of Proposed Web Tutorial Ontology In Pace Protégé

A template has been constructed for the instructional teacher, which can be used, retrieved and altered for other domain subjects for reusability and adaptability. This template is located within the tomcat architecture file structure, which will be discussed in the next chapter. This course module for this research is in the subject area of Introduction to Web Technology and the domain ontology name for this course is called Web Tutorial Ontology. Pace Protégé is used along with Pace Universities OWLViz which are important tools for the instructional teacher to be able to visualize the knowledge graph domain constructs along with visualization of developing and extending OWL and new custom relations.

3.6.3 Student Login Process Use Case

Use Case Name	Student Use Case
Primary Actor(s)	Student
Description	Student's personalized learning
Precondition	Must create a user account and select a learning level
Post condition	Student must login and logout when done.
Basic Course of Action	
<p>The typical use-case is:</p> <ol style="list-style-type: none"> 1. A user logs in, via a web browser (HTTP client), issues a URL request to an HTTP server to start a webapp. 2. A client-side program (such as an HTML/JSP form) is loaded into client's browser. 3. User is presented with his/her learning level and learning order in the navigational pane. 4. User selects the learning object concept from the navigation pane 	

5. The client-side program sends the query parameters to a server-side program.
6. The server-side program receives the query parameters, queries PaceJena, first by reading the knowledge graph, secondly by parsing the request in the Sax Parser and finally retrieves the URI from the database and returns the query result to the client.
7. The client-side program displays the query result on the browser.
8. The process repeats.

Alternative Course of Action

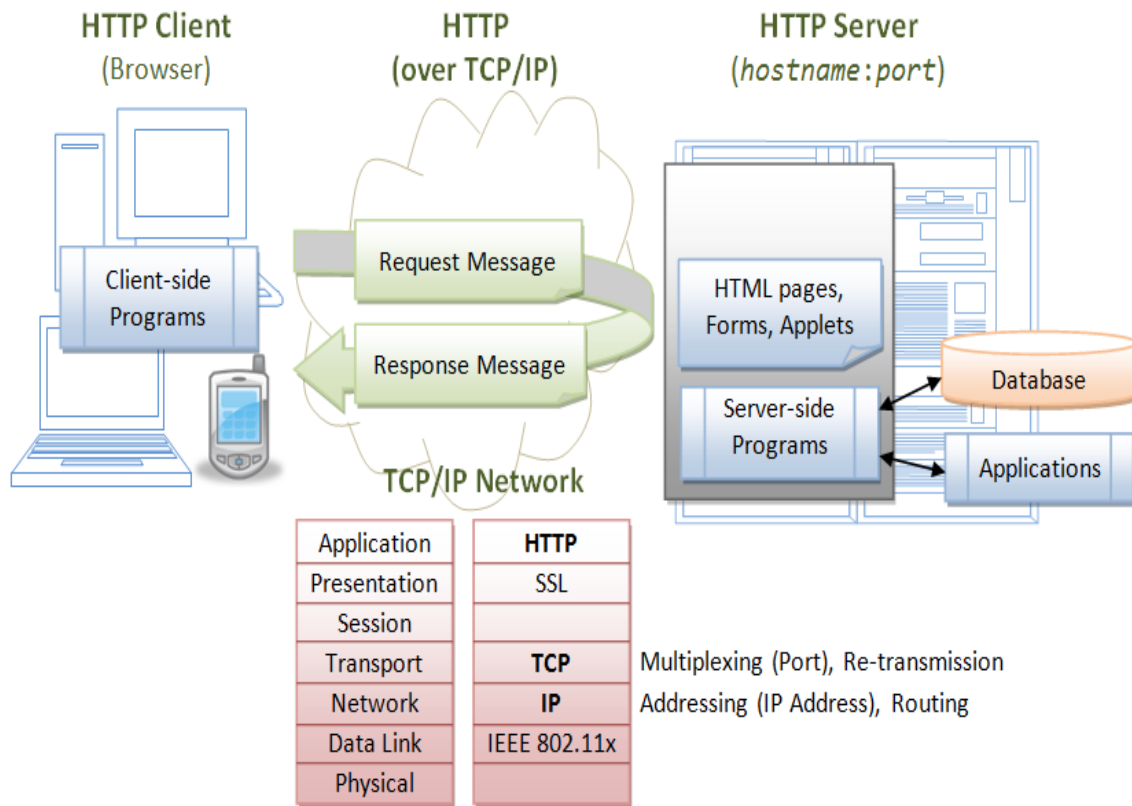


Figure 47 Student Login Process Use Case

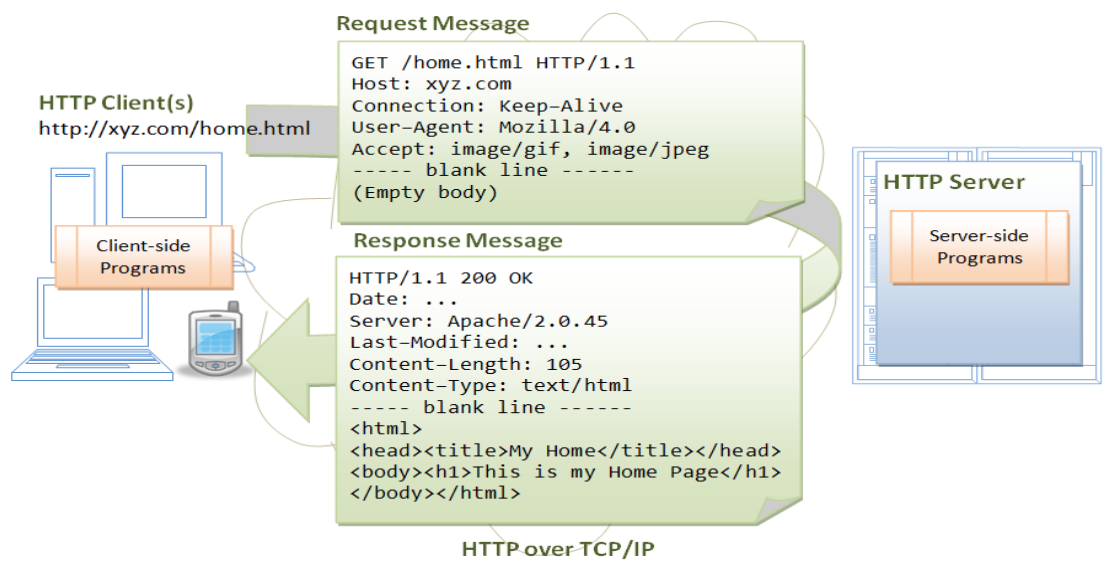


Figure 48 Student Login Process Use Case

3.6.4 Login Use Case

Use Case Name	Login
Primary Actor(s)	User, Administrator, and Instructor /Teacher
Description	Login to system. This use case describes how users gain access to the PaceJena Tutoring System through the login process.
Precondition	All authorized users must have system account into order to login with username and password.
Post condition	The system accepts username and password
Basic Course of Action	
<ol style="list-style-type: none"> 1.) All authorized users must log into the system. 2.) Users must have username and password. 3.) User enters their username and password. 4.) The system checks the username and password. 5.) The system accepts username and password. 6.) The login use case ends. 	
Alternative Course of Action	
Alternative Course of Action A	
<p>A1. If the users have no username and password.</p> <p>A2. Users create their own new account by registration process.</p> <p>A3. Go to step 3.</p>	
Alternative Course of Action B	
<p>B1. If the users enters incorrect username and password.</p> <p>B2. Try with another username and password.</p> <p>B3. Go to step 4.</p>	

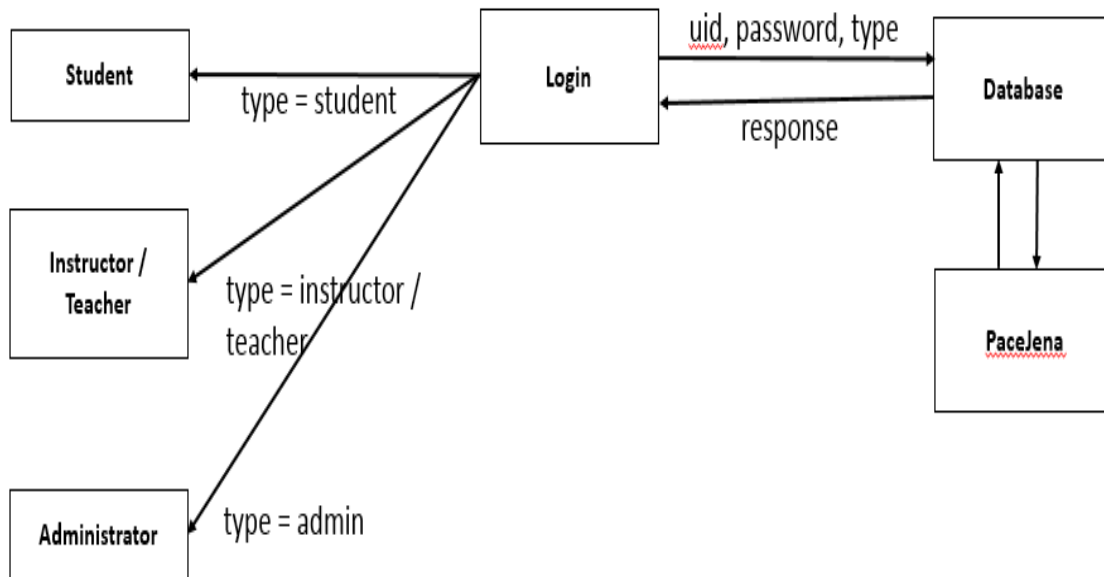


Figure 49 Interaction Diagram for Login Use Case

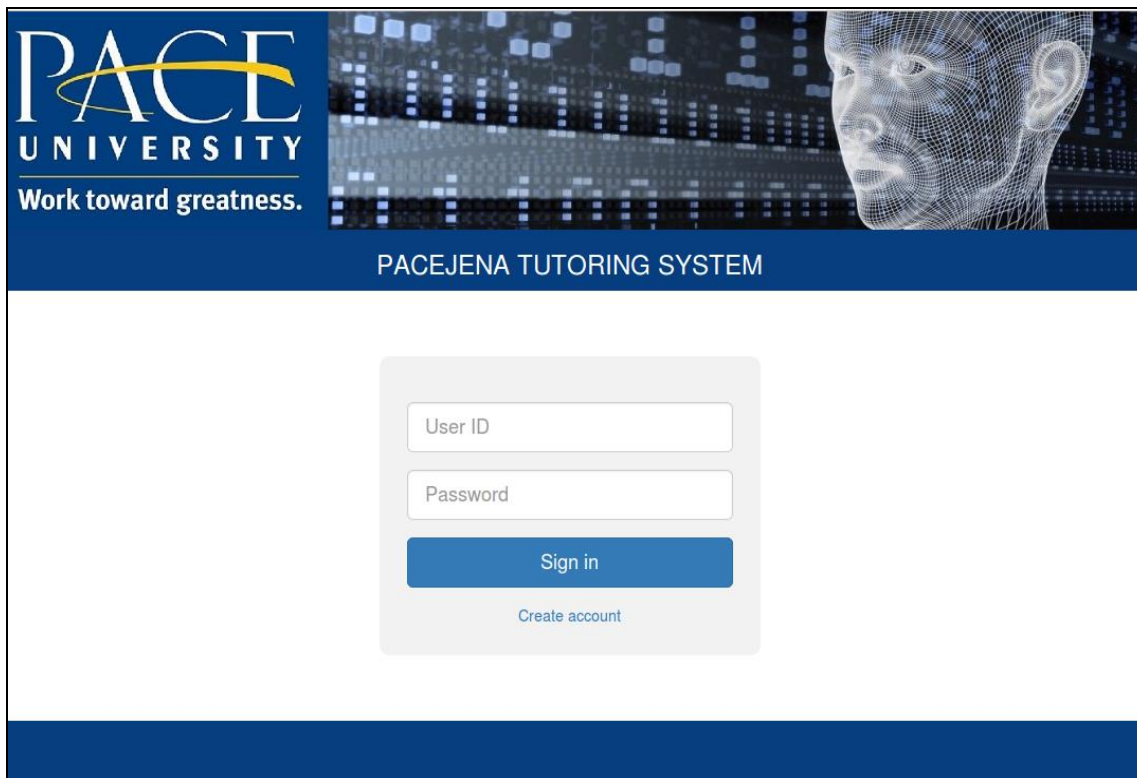


Figure 50 Login Use Case

3.6.5 Registration Use Case

Use Case Name	Registration Use Case
Primary Actor(s)	User and Tutoring System
Description	User registration is required to gain access into system This use case describes how users gain access to the PaceJena Tutoring System through the registration process.
Precondition	User must input all required information into the registration form.
Post condition	User is registered
Basic Course of Action	
<ol style="list-style-type: none"> 1.) The user browses to the "Login" page. 2.) The new user chooses the "Create Account" link on the "Login" page. 3.) The new user is at the Registration form screen. 4.) The new user enters the following details in the Account Information Screen. <ol style="list-style-type: none"> a. User ID b. Email c. First Name d. Last Name e. Password f. Confirm Password g. Select One Learning Level <ol style="list-style-type: none"> i. Beginner ii. Intermediate 5.) The user clicks on Register 6.) The system validates the information entered 7.) The system will return the new user back to the Login screen 8.) New user enters the new Username and Password 9.) The system accepts username and password. The use case ends. 	
Alternative Course of Action	
<p>Alternative Course of Action A</p> <p>A1. If the users have no username and password.</p> <p>A2. Users create their own new account.</p> <p>A3. Go to step 2.</p> <p>Alternative Course of Action B</p> <p>A1. If the users enters incorrect username and password.</p> <p>A2. Try with another username and password.</p> <p>A3. Go to step 4.</p>	
Special Requirements	
<ul style="list-style-type: none"> • After three consecutive unsuccessful login attempts, the user's account will be locked and must be reset by a system administrator. 	

- Users may not login from multiple different computers simultaneously. If this condition is detected, the user will be notified with appropriate warning/error messages.

Frequency of Occurrence

Users must log in to access their account information, to get tutorial, and, to take a quiz. The system administrator must log in to administer the system.

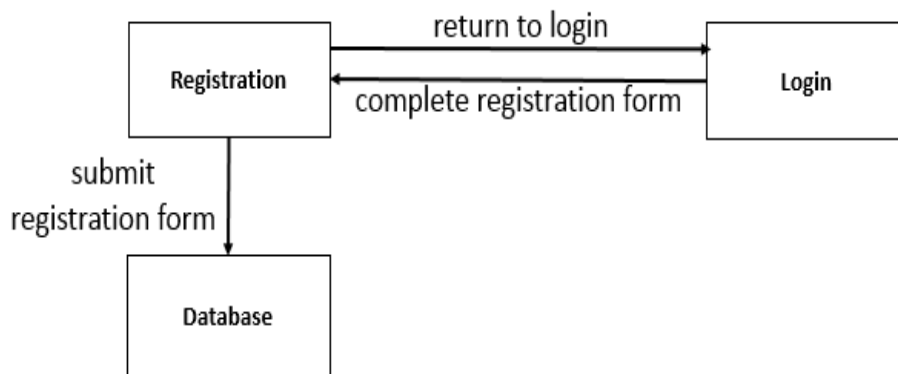


Figure 50 Interaction Diagram for Registration Process

The screenshot shows the registration interface for the PACEJENA Tutoring System. At the top left is the PACE UNIVERSITY logo with the tagline 'Work toward greatness.'. The header area includes the text 'PACEJENA TUTORING SYSTEM'. The registration form contains the following fields:

- User ID: text input field
- Email: text input field
- First Name: text input field
- Last Name: text input field
- Password: text input field
- Confirm Password: text input field
- Learning Level: dropdown menu with 'Beginner' selected

 A blue 'Register' button is located at the bottom of the form.

Figure 52 Registration Use Case

3.6.6 Login Learning Order Use Case

Use Case Name	Login Learning Order Use Case
Primary Actor(s)	User
Description	User is required to select learning level during new account registration. The learning level will determine the learning order.
Precondition	Select learning level during new account registration.
Post condition	System will retrieve learning order for learning level that was selected during new account registration for user each user.
Basic Course of Action	
<ol style="list-style-type: none"> 1.) User is at the new account registration form. 2.) User must select a learning level from pull down menu. 3.) User submits form. 4.) User information is stored in database. 5.) User login. 6.) System will retrieve the last user learning level and learning order state stored in system. 7.) Learning order will be displayed at the left side pane for user to navigate learning structure. 8.) End of login learning order. 	
Alternative Course of Action	
<ol style="list-style-type: none"> 1.) Learning order will change after student takes a quiz. This is explained in the quiz use cases. 	

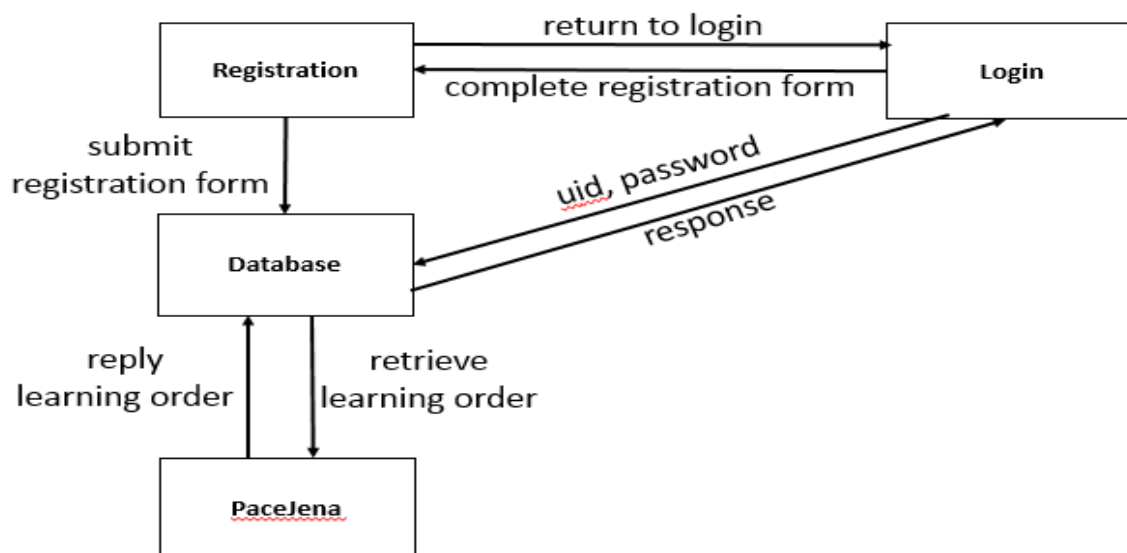


Figure 53 Interaction Diagram for Login Learning Order Process

PACE UNIVERSITY
Work toward greatness.

PACEJENA TUTORING SYSTEM

Home Profile Reports Welcome, Maria (Intermediate) Logout

Learning Order

- XHTML
- HyperLinks
- HtmlForm
- WebPage
- Challenge Quiz

HyperText Markup Language, commonly abbreviated as **HTML**, is the standard **markup language** used to create **web pages**. Along with **CSS**, and **JavaScript**, HTML is a cornerstone technology used to create web pages,^[1] as well as to create user interfaces for mobile and **web applications**. **Web browsers** can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website **semantically** and, before the advent of **Cascading Style Sheets (CSS)**, included cues for the presentation or appearance of the document (web page), making it a **markup language**, rather than a **programming language**.

HTML elements form the building blocks of HTML pages. HTML allows **images** and other objects to be embedded and it can be used to create **interactive forms**. It provides a means to create **structured documents** by denoting structural **semantics** for text such as headings, paragraphs, lists, **links**, quotes and other items. HTML elements are delineated by **tags**, written using **angle brackets**. Tags such as `` and `<input />` introduce content into the page directly. Others such as `<p>...</p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed **scripts** written in languages such as **JavaScript** which affect the behavior of HTML web pages. HTML markup can also refer the browser to **Cascading Style Sheets (CSS)** to define the look and layout of text and other material. The **World Wide Web Consortium (W3C)**, maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.^[2]

Figure 54 Login Learning Order Use Case

3.6.7 Knowledge Space Exploration Use Case

Use Case Name	Knowledge Space Exploration Use Case
Primary Actor(s)	User and Navigation
Description	This use case describes how the User of the system can navigate through the PaceJena Tutorial System.
Precondition	<p>‘Learning Order’ page should be loaded for either “Beginner” or Intermediate”.</p> <p>Success Guarantee (Post Conditions):</p> <ol style="list-style-type: none"> User is able to navigate through his/her learning order for Introduction to Web Technology learning resources successfully. User can navigate from a general concept to a narrow concept, and from narrow concepts to general concepts.
Post condition	User takes quiz or exit system
Basic Course of Action	

1. User logs into system with username and password.
2. The system will retrieve and display the navigation learning order from either initial registration or from quiz result and present it on the left side navigational pane as navigational buttons. The center pane will display the “Web Technology” introduction page.
3. User clicks on ‘Web Architecture’ button link from left side navigational pane.
4. System displays the next level of learning order for ‘Web Architecture’, which will be “Tier 1”, “Tier 2”, “Tier 3”, “Tier 4”, and ‘Web Architecture’.
5. User can take a quiz or log off system.

Alternative Course of Action

1. System will display “Tier 1”, “Tier 2”, “Tier 3”, “Tier 4”, and ‘Web Architecture’.
2. User clicks on ‘Web Architecture’ button link in the Screen.
3. System displays the previous level of learning order, which is “Web Page”, “HTML”, “Session Management”, and “Web Architecture”.



Figure 55 Knowledge Space Exploration Use Case

3.6.8 Quiz Module Overview

This quiz module represents an interaction with our PaceJena Tutoring System, which will evaluate and personalize students learning environment. The students may take this quiz at any point and time during their learning material studies. The block diagram below is a representation of how students and instructor / teachers interact with the quiz module. Students whom are “Beginners” will be quizzed on a “Beginner” quiz. Students whom are “Intermediate” will be quizzed on “Intermediate” quiz. Students whom receive a grade of greater than 70 will be allow to proceed to the next level. Students whom receive a grade less than 70 will not be allowed to proceed to the next level and must return back and study the learning material again.

3.6.8.1 Instructor / Teacher Editing Quiz Assessment Use Case

Use Case Name	Teacher Editing Quiz Assessment Use Case
Primary Actor(s)	Instructor / Teacher
Description	Instructor / Teacher wants to evaluate students.
Precondition	Teacher is identified and authenticated.
Post condition	Quiz assessment is stored and saved in database.
Basic Course of Action	
<ol style="list-style-type: none"> 1.) Instructor / Teacher logins into system using the Administrator user name and password. 2.) Instructor / Teacher clicks quiz for any concept. 3.) Instructor can edit and save a quiz questions and answers. 4.) Quiz questions and answers are saved to the database. 5.) End use case. 	
Alternative Course of Action	
None	

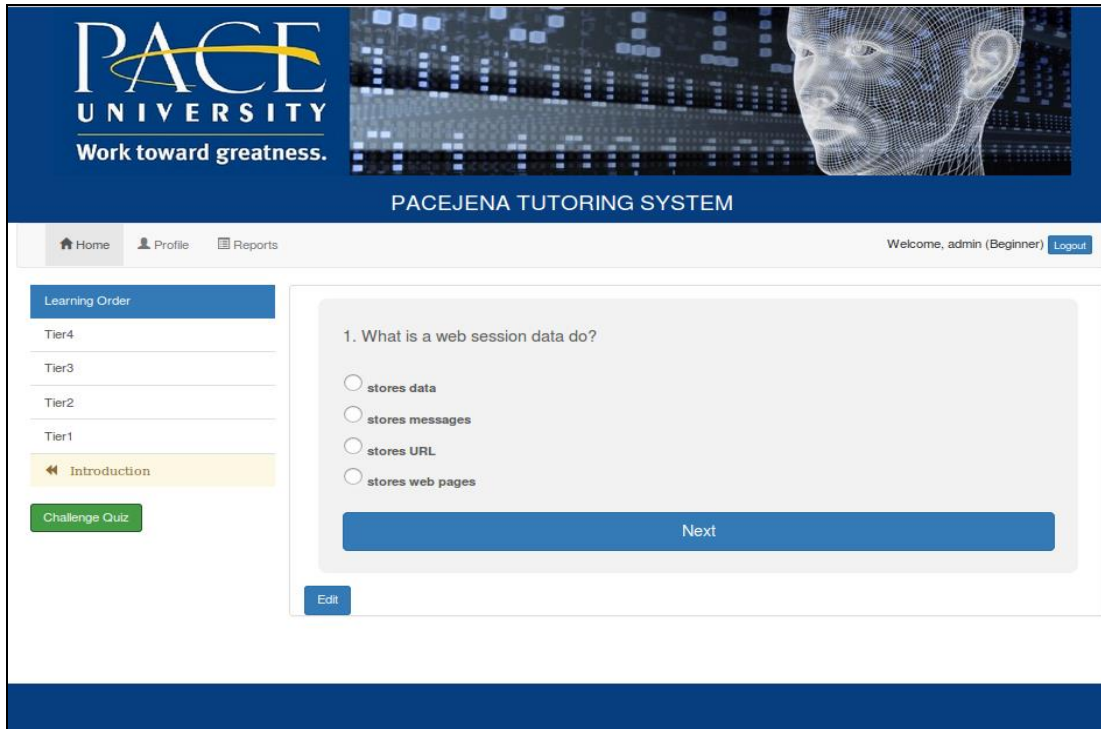


Figure 56 Instructor / Teacher Editing Quiz Assessment Use Case

3.6.8.2 Student Quiz Assessment Use Case

Use Case Name	Student Quiz Assessment Use Case
Primary Actor(s)	Student
Description	Student wants to be evaluated and get feedback from taking the intelligent quiz.
Precondition	Student is identified and authenticated.
Post condition	All quiz results are stored and saved in database.
Basic Course of Action	
<ol style="list-style-type: none"> 1.) Student logs into system. 2.) Student clicks on challenge quiz button for his/her level. 3.) If student is beginner then a beginner quiz will appear. 4.) If student is intermediate then an intermediate quiz will appear. 5.) Student will be presented 10 random and dynamic questions one at a time. 6.) System will display first random question. 7.) Student will view and answer each question, then click on submit button. 8.) System will display the next question. 9.) Student must answer each question before proceeding to the next question. 10.) Student submits the last question the system will assess all answers. 11.) System will display the quiz results and give feedback on student's next step. 	

- 12.) If student received a >70 result, then student can proceed to intermediate concept topic.
- 13.) If student received a <70 , result then student will be instructed to go back and review course materials.
- 14.) Student will follow a link to next step
- 15.) End use case.

Alternative Course of Action

None

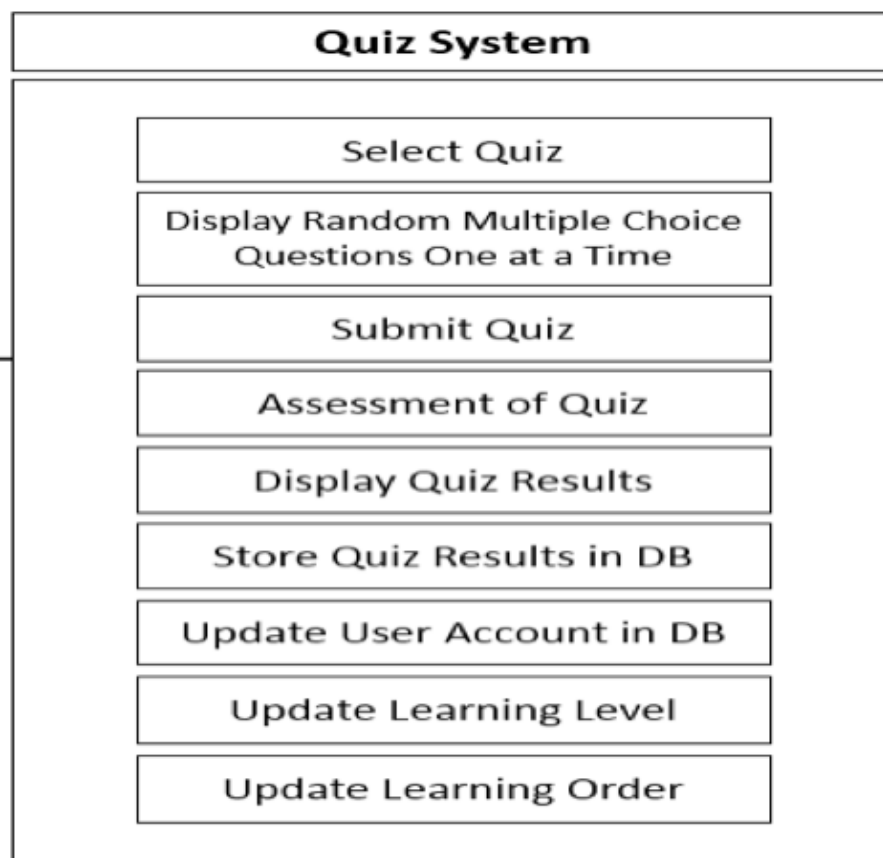


Figure 57 Quiz Diagram

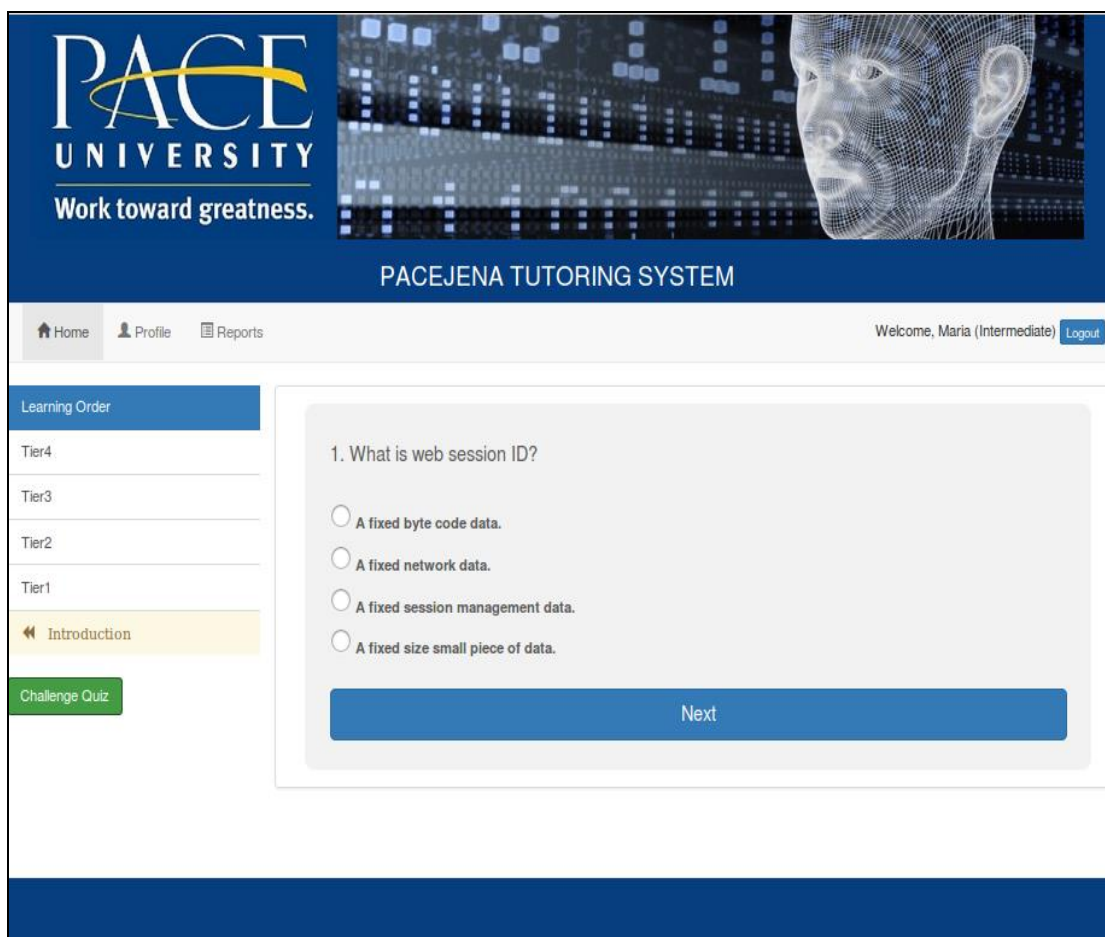


Figure 58 Student Quiz Assessment Use Case

3.6.8.3 Retrieve Quiz Result Learning Order Use Case

Use Case Name	Retrieve Quiz Result Learning Order Use Case
Primary Actor(s)	Quiz Module
Description	Quiz module will evaluate all 10 questions and determine the outcome based on whether the student can achieve a greater than 70 result to deliver the next learning order for the next learning level. If not then students will be returned to same learning order and learning level until student can achieve the greater than 70 requirement.
Precondition	Student must complete all 10 questions and submit to the quiz module in order to evaluate and return results.
Post condition	Depending on the results student will be either directed to go to next learning order and learning level or student maybe directed to go back and study the learning materials again until the student can achieve a greater than 70 score.

Basic Course of Action
1.) Login 2.) Retrieve learning order 3.) Take quiz 4.) Quiz contains 10 questions 5.) Submit quiz 6.) System evaluates quiz 7.) System returns quiz results 8.) If greater than 70 direct to go to next learning level and order. 9.) If less than 70 direct to go back and study learning materials again for current learning level and order. 10.) End use case.
Alternative Course of Action
None

3.6.9 Conclusion

This section discusses the different use cases for the PaceJena Tutoring System prototype.

3.7 Pace Protégé Extension for Knowledge Representation

Protégé is an open-source tool developed at Stanford University Medical Informatics. It is an open source tool, and as an ontology editor, it provides a suite of tools to construct the domain model using various formats. Also, using plug-ins for adding further functions makes it flexible. These plug-ins such as importing and exporting ontology language specifications like XML, RDF, RDFS, OWL and different types of reasoner are available. Pace Protégé is an extension of Protégé. Figure 59 shows the startup of the Pace Protégé from the command line.


```

Framework: Apache Software Foundation (1.7)
OS: linux (3.19.0-25-generic)
Processor: x86-64
Plugin: OWLAPI RDF Library (1.0.2)
Plugin: Guava: Google Core Libraries for Java (18.0.0)
Plugin: Cajun Visualization Library (1.0.2)
Plugin: Protege Editor OWL (5.0.0.beta-18-SNAPSHOT)
Plugin: OWLViz (4.1.6.SNAPSHOT)
Plugin: OWL Difference Engine (2.0.0)
Plugin: DL Query (2.0.2)
Plugin: OWLAPI OSGi Distribution (3.5.1)
Plugin: Protege Hermit Integration (1.0.0)
Plugin: OWL Code Generation Plug-in (1.0.2)
Plugin: Protege SPARQL Plugin (1.0.0)
Plugin: OWL Difference (5.0.0)
Plugin: OntoGraf (1.0.3)
Plugin: Explanation Workbench (2.0.0)
      OWLAPI RDF Library Plugin has no plugin.xml resource
      Guava: Google Core Libraries for Java Plugin has no plugin.xml resource
      Cajun Visualization Library Plugin has no plugin.xml resource
      OWL Difference Engine Plugin has no plugin.xml resource
      OWLAPI OSGi Distribution Plugin has no plugin.xml resource
Using OWL API version 3.5.1

```

Figure 59 Launching Pace Protege from Command Line

Web Ontology Language (OWL) is used by domain experts to encode knowledge. OWL primarily only supports the subClassOf (is-a or inheritance) relation. Various other relations, such as partOf relations, are essential for representing information in various fields including all engineering disciplines. The current syntax of OWL does not support the declaration and usage of new custom relations. Representing part-whole relationship is a very common issue among people who want to develop ontologies for semantic web. The “partOf” relationship is one of the basic fundamental primitives of the universe. Many applications are required to show this kind of relationships in the real world. RDF schema and OWL does not provide any built-in primitives to support part-whole relationship. In Workarounds to emulate custom relations do exist, but they add syntax burden to knowledge modelers and don’t support accurate semantics for inference engines. The Pace extended version of Protégé and OWLViz allows us to declare new

custom relations with special attributes, and apply the knowledge representation to be visualized in a knowledge graph. In figure 60, we are able to extend the “partOf” relations and custom relations such as “include”, “ref”, “implemented”, and “implementedBy”.

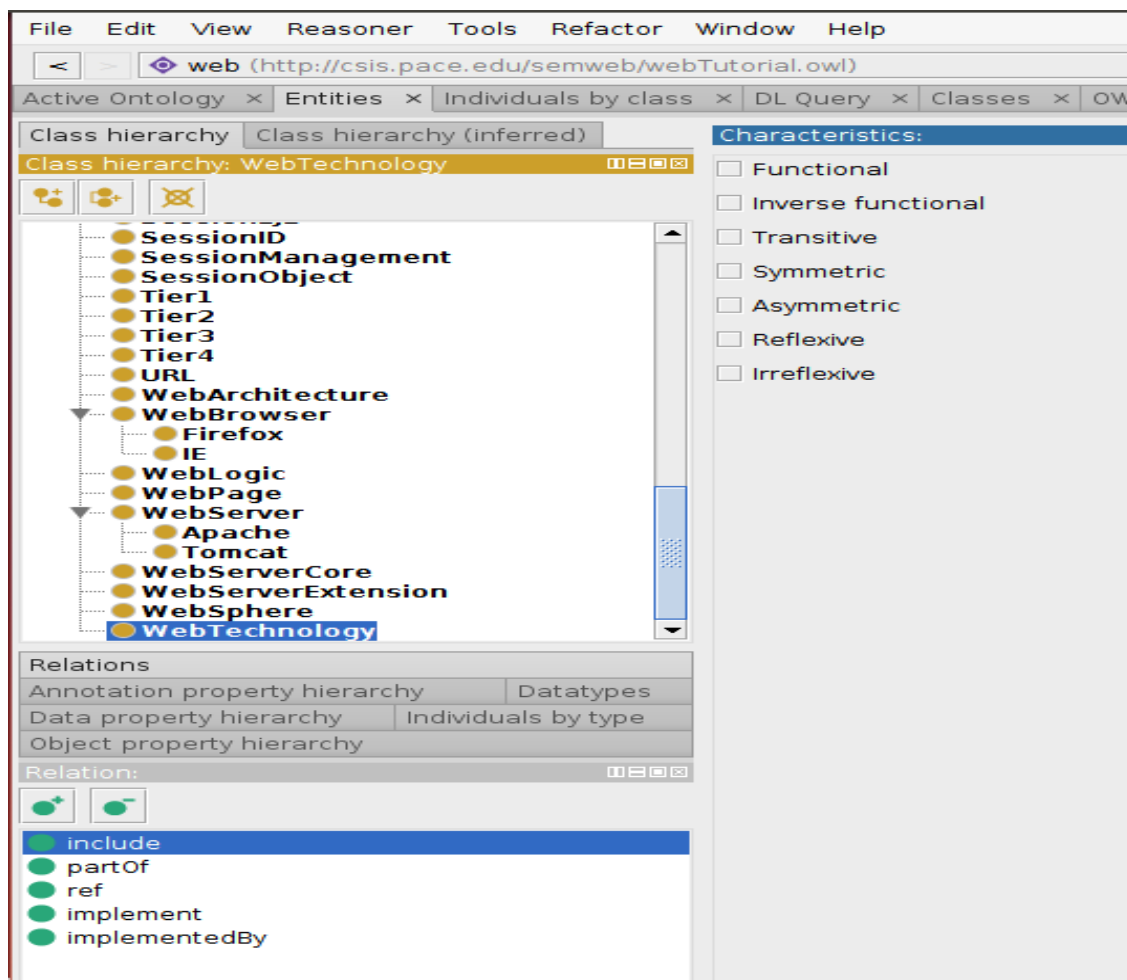


Figure 60 Pace Protege GUI Interface with Custom Relations Features

The entities tab allows us to view the class hierarchy for the Web Tutorial domain ontology. In this pane we can add new classes, sub classes and remove classes. The relation’s pane allows us to add our custom relations and remove custom relations. In Figure 61, Pace Protégé GUI interfaces with custom relations and related to feature show

how to relate classes to the “include” relations for the class “WebTechnology”. The relations are used to relate these relations to each class in the domain. Once this has been completed we can then use Pace’s extended customized version of OWLViz to visualize our knowledge graph with these new custom relations.

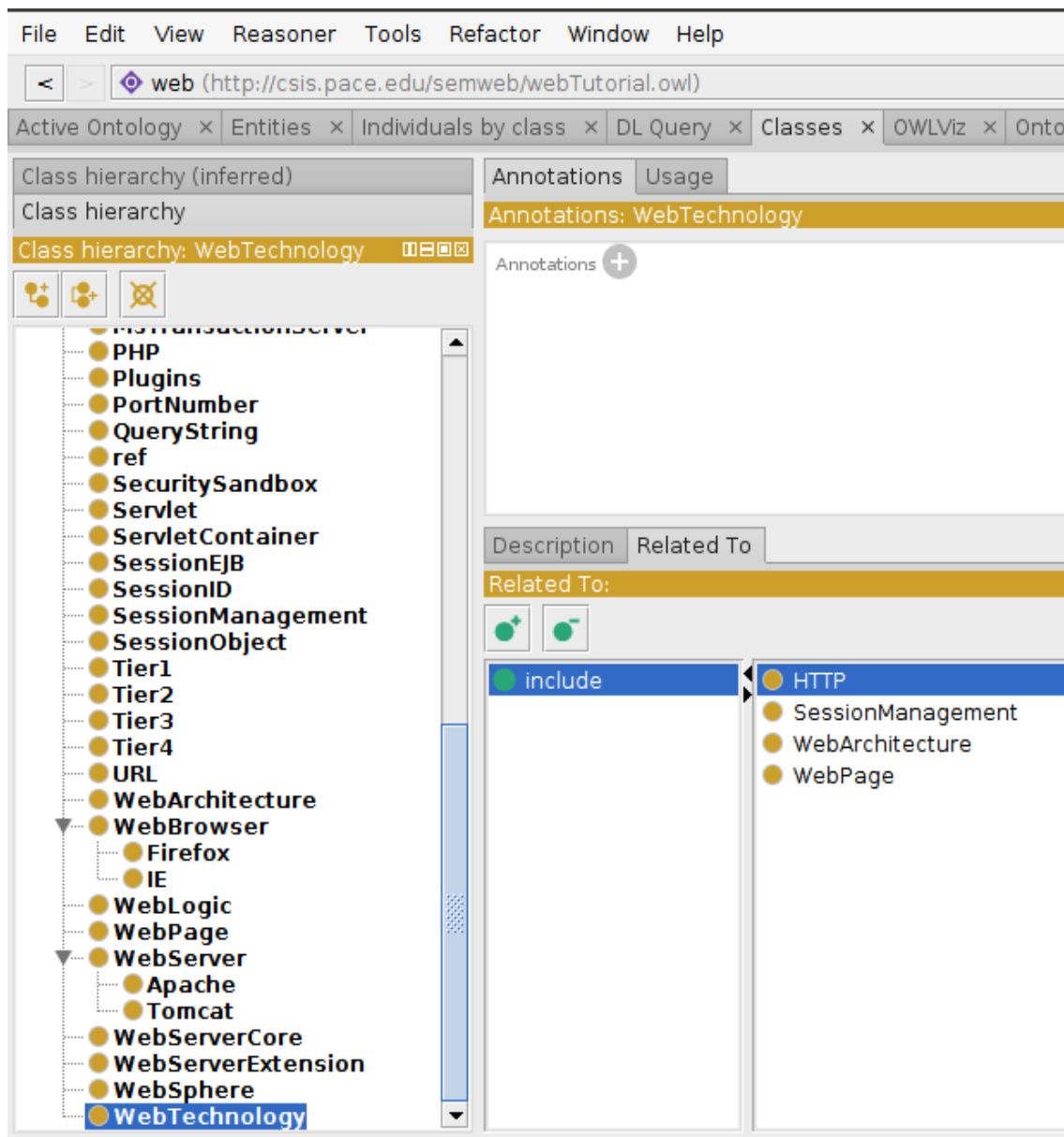


Figure 61 Pace Protege GUI Interface with Custom Relations and Related To Feature

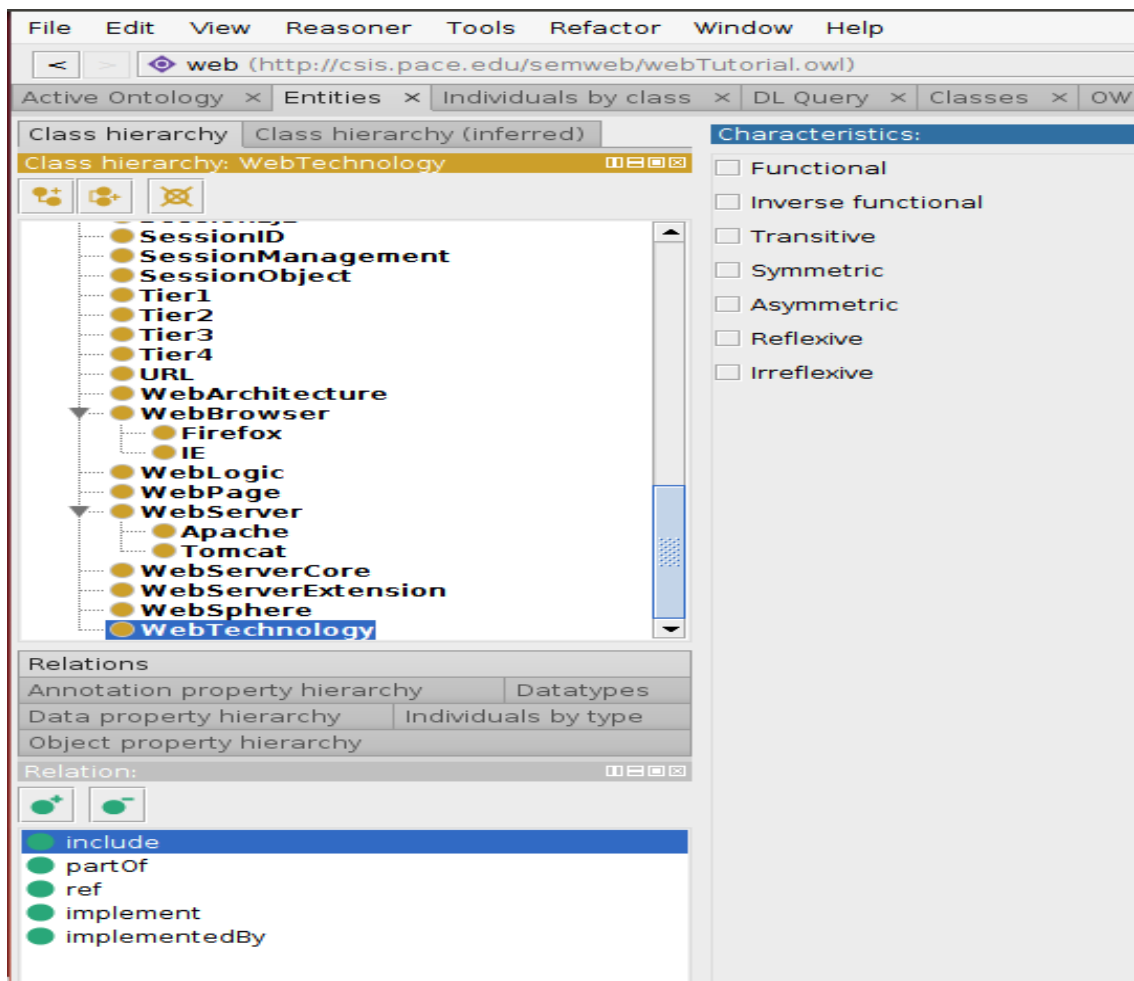


Figure 62 Pace Protege GUI Interface with Customized OWLViz Knowledge Graph Features

OWLViz was developed as part of the CO-ODE project by Mathew Horridge at the University of Manchester. It is licensed under the Lesser GNU Public License. OWLViz uses the layout algorithms provided by the GraphViz software by AT&T, and also the Balik libraries by the Apache Software Foundation. OWLViz is design to be used in Protégé as a plugin. It enables the class hierarchies in OWL Ontology to be viewed and incrementally navigated, allowing comparison of asserted class hierarchy and inferred

class hierarchy. The finally step is to save this knowledge graph as an rdf/xml file. This file will be used in our PaceJena API.

3.8 PaceJena for Knowledge Navigation

Knowledge navigation in the PaceJena Tutoring System is used to describe learner navigation among digital learning resources, guided by a knowledge structure graph of the “Web Tutorial” domain concepts. The motivation of the PaceJena Tutoring System is to provide learners with an improved navigational process for personalized learning using learning objects. The conceptualization knowledge graph means that an explicit representation of concepts and relations are involved to support the navigational activities such as navigating to reading a document, viewing a video, or listening to audio, and much more. Learners are navigating the knowledge structure graph from top level to next level and from next level to previous level. These actions allow the user to move from one resource to another.

3.9 Knowledge Graph Representation for Navigation

The knowledge graph representation for navigation plays a major role in the PaceJena Tutoring System for personalizing and adapting the users learning level, learning order and quiz knowledge assessment. But in order for this to happen in our PaceJena Tutoring System the user must create a user account. The user account is created from the registration form. The registration form contains input information that must be entered

in order for the system to accept your registration. This information contains the following information: User must enter a User ID, Email, First Name, Last Name, Password, Confirm Password, and Learning Level. This information then gets logged into the MYSQL database. The student can now access the PaceJena Tutoring system. Student is presented with the login page where he/she must enter her username and password. Once the user is authenticated into the system a HTTP request is sent to the Tomcat server-side where the request accesses the knowledge graph and is parse by the Sax Parser and the server-side sends the responses back to the client-side via HTTP. The response's is displayed on the browser where the student will see their learning order in the navigational pane. The function of the navigational pane allows the user to navigate the knowledge graph structure from narrow concepts to board concepts and from board concepts to narrow concepts. If a student wishes to take the quiz to assess his / her learning understanding of the domain then he or she will need to answer 10 questions. After completing the quiz assessment the system will evaluate the student's performance and will recommend the student to either proceed to the next topic or return to the previous topic. Depending on the student's performance the system will either navigate from a narrow to a boarder concept or a boarder to narrow concept or proceed forward to the next concept on the knowledge graph.

3.10 Summary

In the chapter we discuss an intelligent tutoring framework, which is empowered by knowledge representation. We examine the importance of the different ITS modules such

as the domain knowledge representation, the user knowledge representation the instructors knowledge representation and the expert knowledge representation and the technology and mechanisms that are used to produce a tutoring system web-based application for learning.

Chapter 4 Intelligent Tutoring System Implementation Highlights and Adaptation to Other Subjects

4.1 Introduction to Tutorial Web-Based System

This web based intelligent tutoring system framework was developed to help support and improve instructors tutoring tasks of organizing and personalizing student's Cyberlearning content. This framework can be used for any tutoring course content. The central part of this tutoring system takes place during the new user account registration form process. The main question during the new user registration process that a student needs to answer is whether they are a "Beginner" or "Intermediate" level for the web tutorial course content. This will determine the initial state of user's personalized learning order. The users profile information is then stored in a MYSQL database and is updated according to the user's current state. This initial system state process tells the tutoring system where the student needs to start at login. The tutoring system will then retrieve this information and respond back with the recommended learning order that the instructor as provided for the system. During the learning process student will be required to take a quiz for each learning concept. The quiz results will evaluate the student's response and recommend the next learning concepts and the order in which these concepts are to be learned. The main purpose of this dissertation is to develop a

Cyberlearning tutorial system framework using open source web based architecture and semantic web technology to address student's needs for personalized learning.

4.2 Overview of Tutorial Web-Based Architecture

The tutoring system web architecture design prototype is implemented using the Tomcat with Spring MVC (Model, View and Controller) patterns. The JSPs (Java Server Pages) are used for the presentation logic. Servlets control the execution of the business rules and presentation logic. The personal data, linked data and quiz data is stored and retrieved from the MYSQL database using URLs and XML.

In this tutorial web system application, the servlets control where and what technology is used to retrieve the personal data, linked data and quiz data information. The servlet's are processed during the user's tutorial training by user's selection of learning concepts or challenge quiz which then invokes the corresponding methods which interact with the dispatcher servlets and containers. Figure 63 illustrates the tutoring system architecture process.

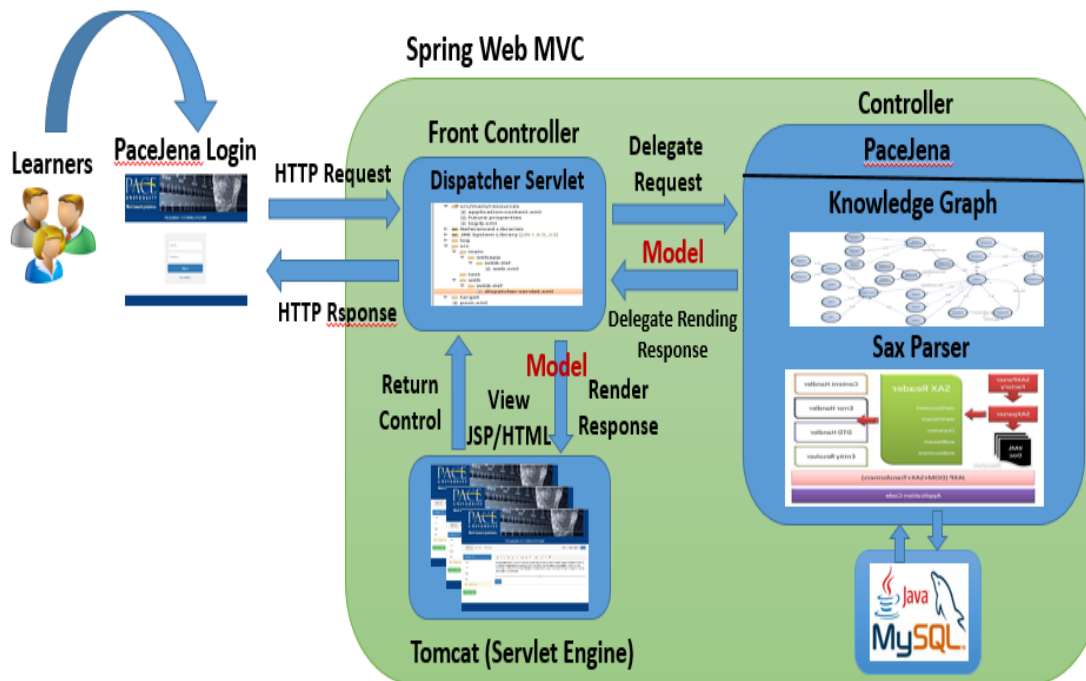


Figure 63 Tutoring System Architecture

4.3 Open Source Web-Based Technologies

The tutoring system was developed using open source web technology architecture. Table 7 below describes the technology name, the version and a brief description.

Table 7 Open Source Web Technology

Open Source Web Technology	Description
Oracle VirtualBox 5.1.6 r110634	<p>VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. See "About VirtualBox" for an introduction.</p> <p>Presently, VirtualBox runs on Windows, Linux,</p>

	<p>Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.</p>
<p>Ubuntu 16.04.1</p>	<p>Ubuntu is published by Canonical Ltd, who offer commercial support.^[12] It is based on free software and named after the Southern African philosophy of <i>ubuntu</i> (literally, 'human-ness'), which Canonical Ltd. suggests can be loosely translated as "humanity to others" or "I am what I am because of who we all are".^[13] It uses Unity as its default user interface for the desktop.</p> <p>Ubuntu is the most popular operating system running in hosted environments, so-called "clouds",^[14] as it is the most popular server Linux distribution.</p> <p>Development of Ubuntu is led by UK-based Canonical Ltd., a company of South African entrepreneur Mark Shuttleworth. Canonical generates revenue through the sale of technical support and other services related to Ubuntu. The Ubuntu project is publicly committed to the principles of open-source software development; people are encouraged to use free software, study how it works, improve upon it, and distribute it.</p>
<p>Apache-Tomcat 7.0.69</p>	<p>Apache Tomcat, often referred to as Tomcat Server, is an open-source Java Servlet Container developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment in which Java code can run.</p> <p>Tomcat is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation, released under the Apache License 2.0 license, and is open-source software.</p>
<p>Eclipse Neon 4.6.0</p>	<p>Eclipse provides IDEs and platforms for nearly</p>

	<p>every language and architecture. They are famous for their Java IDE, C/C++, JavaScript and PHP IDEs built on extensible platforms for creating desktop, Web and cloud IDEs. These platforms deliver the most extensive collection of add-on tools available for software developers.</p>
<p>MYSQL Workbench 6.3 Community</p>	<p>MySQL Workbench is a unified visual tool for database architects, developers, and DBAs. MySQL Workbench provides data modeling, SQL development, and comprehensive administration tools for server configuration, user administration, backup, and much more. MySQL Workbench is available on Windows, Linux and Mac OS X.</p>
<p>Spring Web MVC and Spring JDBC Framework 3.2.9 Release</p>	<p>The Spring Web model-view-controller (MVC) framework is designed around a DispatcherServlet that dispatches requests to handlers, with configurable handler mappings, view resolution, locale, time zone and theme resolution as well as support for uploading files. The default handler is based on the @Controller and @RequestMapping annotations, offering a wide range of flexible handling methods. With the introduction of Spring 3.0, the @Controller mechanism also allows you to create RESTful Web sites and applications, through the @PathVariable annotation and other features.</p>
<p>Ajax</p>	<p>Modified in accordance with Spring Security 3.2</p> <p>Changed the sample code for CSRF measures (method to create <meta> tag for CSRF measures).</p> <p>Modified in accordance with Jackson 2.4</p> <p>Changed the sample code and description to use components for Jackson 2.4.</p>
<p>Spring Security 3.2.9 Release</p>	<p>Spring Security is a framework that focuses on providing both authentication and authorization to Java applications. Like all Spring projects, the real power of Spring Security is found in how easily it can be extended to meet custom requirements. [45]</p>

jQuery 2.1.1	jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.
Bootstrap 3.3.7	Bootstrap makes front-end web development faster and easier. It's made for folks of all skill levels, devices of all shapes, and projects of all sizes. With Bootstrap, you get extensive and beautiful documentation for common HTML elements, dozens of custom HTML and CSS components, and awesome jQuery plugins.
Apache Maven	Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.
Pace Protégé 1.0 Release	The Pace Protégé tool is an extension of Protégé. The Pace Protégé was extended to create knowledge graphs with other relations besides the is-a relation. Protégé is a free, open source ontology editor and a knowledge management system. Protégé provides a graphic user interface to define ontologies. It also includes deductive classifiers to validate that models are consistent and to infer new information based on the analysis of an ontology. Like Eclipse, Protégé is a framework for which various other projects suggest plugins. This application is written in Java and heavily uses Swing to create the user interface. Protégé is being developed at Stanford University and is made available under the BSD 2-clause license. ^[6] Earlier versions of the tool were developed in collaboration with the University of Manchester.
PaceJena API 1.0 Release	PaceJena API is an extension of Jena API. PaceJena API as been customized with the Sax Parser to read RDF/XML documents and graphics. Apache Jena is an open source Semantic Web framework for Java. It provides an API to extract data from and write to RDF

	graphs. The graphs are represented as an abstract "model". A model can be sourced with data from files, databases, URLs or a combination of these. A Model can also be queried through SPARQL 1.1.
Summernote V0.8.2	Super Simple WYSIWYG Editor on Bootstrap Summernote is a JavaScript library that ... Summernote is licensed under MIT and maintained by the community. The software can be accessed from http://summernote.org/getting-started/

4.4 Web Tutorial Knowledge Representation

This section gives an example of the web tutorial knowledge representation of a web technology domain to demonstrate how the collection of learning objects are represented, structured and organized. This section describes how the web technology domain is used in this dissertation to provide the reasoning, methods and functional capabilities of the tutorial systems adaptive features using the knowledge graph to support and improve personalized web-based learning. The core ingredients of the knowledge graph includes a set of concepts, properties, and relations between the elements of these sets. The introduction to web technology offers the classification level to describe how this dissertation extends the OWL language using the "include" and "partOf" relations, along with custom relations to produce a knowledge graph. The knowledge graph is an overview of the domain concepts and relations among them.

The following constraints present some partial code of the web technology knowledge graph to illustrate the OWL-based description logics, including class, OWL properties,

subclass, individual, transitive property, inverse property, and hierarchy such as generalization and specialization respectively.

Definition of Class – OWL Expression

A Class is a set of individuals. A Class in OWL is a classification of individuals into groups, which share common characteristics. If an individual is a member of a class, it tells a machine reader that it falls under the semantic classification given by the OWL class.

Definition of OWL Properties – OWL Expression

Individuals in OWL are related by properties. There are two types of property in OWL:

- Object properties (owl:ObjectProperty) relates individuals (instances) of two OWL classes.
- Datatype properties (owl:DatatypeProperty) relates individuals (instances) of OWL classes to literal values.

Definition of Subclass - OWL Expression

Subclass denotes a class is a subset of another class, meaning that an implication via inference is that all members of a subClass are members of the superClass. There is no concept of inheritance in OWL. Figure 63, is an example of subclass.

```
<!-- http://csis.pace.edu/semweb/webTutorial.owl#Tomcat -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Tomcat">
  <rdfs:subClassOf rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebServer"/>
</owl:Class>
```

Figure 63 SubClass

Semantic meaning: The Tomcat is a subclass of the WebServer.

Rule expression: If Tomcat (x) then WebServer (x)

Definition of Individuals

Individual instances are members of classes. Instances of a class in Figure 64, are the following concepts HTTP, SessionManagement, WebArchitecture, and WebPage are individuals and a member of Web Technology. So we can enumerate a class for which all members can be listed. The “include” relation is used to express that these individual classes are a member and a “partOf” WebTechnology.

```

<!-- http://csis.pace.edu/semweb/webTutorial.owl#WebTechnology -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#WebTechnology">
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTTP"/>
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#SessionManagement"/>
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebArchitecture"/>
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebPage"/>
</owl:Class>

```

Figure 64 Sample of Individual Classes and Members



Figure 30 Sample of Knowledge Graph of Individual Classes and Members

Definition of Transitive Property - partOf Expression

Semantic meaning: Parts of parts are part of the whole. If A is part of B and B is part of C, then A is part of C. If WebPage include with HTML include with HyperLinks then WebPage include with HyperLinks. Transitive is functional and inverse functional.

Rule expression: If include (x, y) and include (y, z) Then include (x, z)

Definition of Inverse Property - partOf Expression

Semantic meaning: The WebPage is partOf WebTechnology, HTML is partOf WebPage, Then WebTechnology must have WebPage as one of its parts and WebPage must have HTML as one of its parts.

Rule expression: If WebPage (x, y), Then standard (y, x)

```
<!-- http://csis.pace.edu/semweb/webTutorial.owl#WebPage -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#WebPage">
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#CSS"/>
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTML"/>
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#JavaScript"/>
  <rel:partOf rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebTechnology"/>
</owl:Class>
```

```
<!-- http://csis.pace.edu/semweb/webTutorial.owl#HTML -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#HTML">
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HyperLinks"/>
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HtmlForm"/>
  <rel:partOf rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebPage"/>
</owl:Class>
```

Figure 31 Transitive and Inverse Property of “part of” Relations

In Figure 67, is a partial representation of the webTutorial.owl file code of how each concept is represent to extend the OWL language and use the customized namespace to represent these relation's. In Figure 67, is a partial representation of the knowledge graph for the concept HTTP.

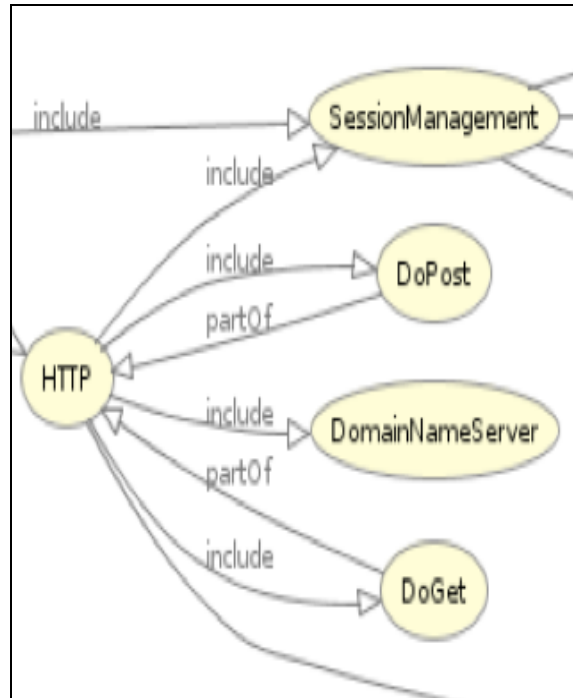


Figure 67 Partial Knowledge Graph Representation of “include and “partOf” Relation

Definition of Generalization and Specialization Relations

In this section we introduce the methods to describe and to infer generalization and specialization relations between concepts and for deriving concepts and learning object resource hierarchies. There exists a generalization and specialization relation between entities for Web Technology, which describes WebPage, HTTP, Session Management, and Web Architecture. Where WebPage is a specialization of Web Technology, and HTML is a specialization of WebPage, and Hyperlink is a specialization of HTML. In Figure 68, is a partial representation of how the user would navigate from top level to the next level and back to the previous level. Figure 69 is a Knowledge Graph that represents the generalization and specialization.

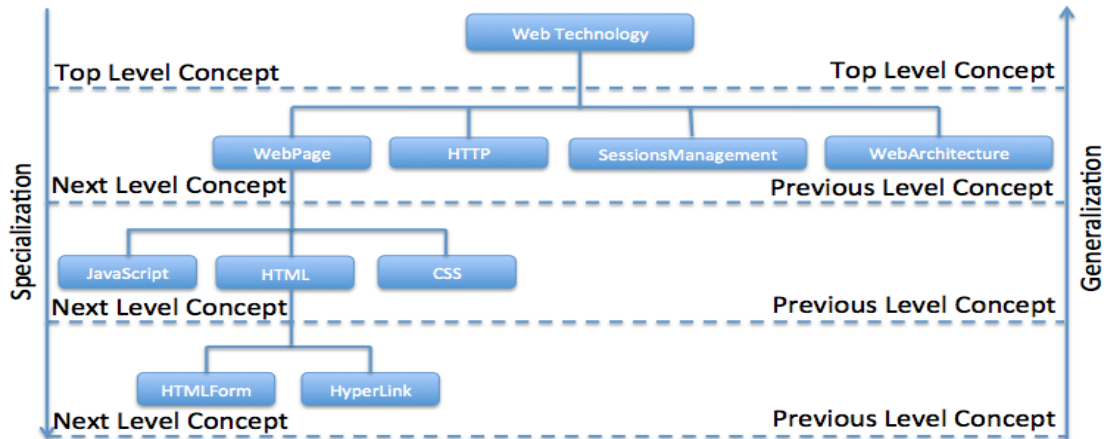


Figure 68 Partial Generalization and Specialization Relations

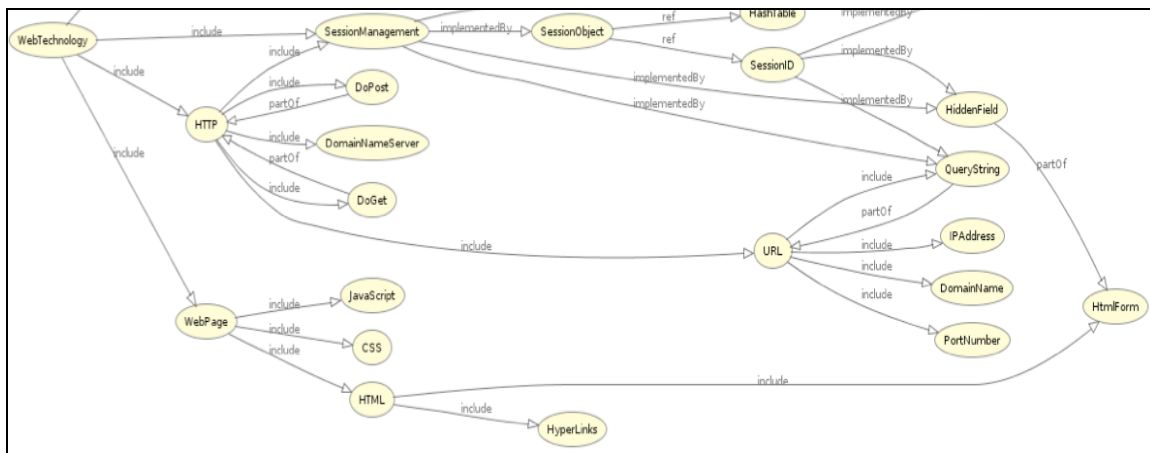


Figure 32 Partial Knowledge Graph of Generalization and Specialization Relations

Definition of the IS-A Relation

Semantic meaning: The is-a relation is transitive, which means that if A is a B, and B is a C, we can infer that A is a C. It should be noted that is-a does not mean ‘is an instance of’. An ‘instance’, ontologically speaking, is a specific example of something; like a cat is-a mammal, but Garfield is an instance of a cat, rather than a subtype of cat. Web Tutorial, like most ontology, does not use instances, and the terms in Web Tutorial represent a class of entities.

```

<!-- http://csis.pace.edu/semweb/webTutorial.owl#Apache -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Apache">
  <rdfs:subClassOf rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebServer"/>
</owl:Class>

```

Figure 33 Represents the IS-A Relation

Definition of the Include Relation

Figure 71 is a sample representation of the “include” relation. This figure describes the root concept to be the Web Technology. The Web Technology uses the “include” relation to describe the next level of concepts such as “HTTP”, “SessionManagement”, “WebArchitecture”, and “WebPage”.

```

<!-- http://csis.pace.edu/semweb/webTutorial.owl#WebTechnology -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#WebTechnology">
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTTP"/>
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#SessionManagement"/>
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebArchitecture"/>
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebPage"/>
</owl:Class>

```

Figure 34 Sample Knowledge Representation of Include Relation

Definition of the New Custom Relation implementedBy

The new custom relation implementedBy represents the concept Application Server which is implementedBy EjbContainer and Transaction Server. Figure 72 is an example of the usage of implementBy and Figure 73 is a representation of the knowledge graph. The inverse of implementedBy is implement.

```

<!-- http://csis.pace.edu/semweb/webTutorial.owl#ApplicationServer -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#ApplicationServer">
  <rel:implementedBy rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#EjbContainer"/>
  <rel:implementedBy rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#MsTransactionServer"/>
</owl:Class>

```

Figure 72 New Custom Relation ImplementedBy

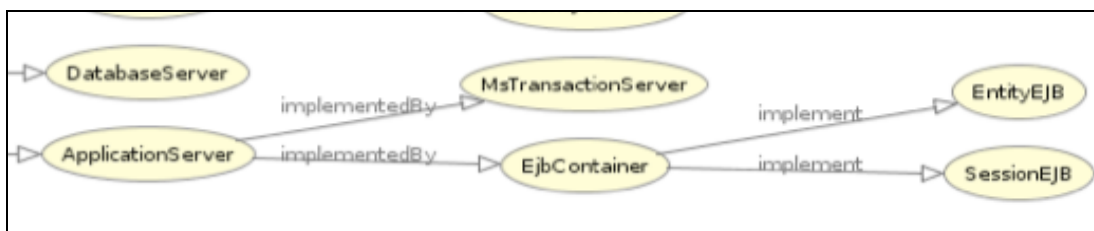


Figure 73 Knowledge Graph of New Custom Relation ImplementedBy

4.5 Web Tutorial File Structure Representation

The knowledge representation is determined by the richness of the ontological modeling to support the course subject. The current industry standard is to use Web Ontology Language (OWL) for representing knowledge structure. But OWL only supports one "first-class" relation, "is-a", between the concepts, and different knowledge areas usually need different custom relations to describe the relations among the concepts. For example "part-of" and time dependency are important relations to represent most engineering knowledge bodies. OWL has to use object properties to emulate such custom relations, leading to awkward knowledge representation hard for domain experts to code, validate and use such knowledge bases. This research uses Pace University's extension to OWL, named Knowledge Graph (KG), to support knowledge representation with custom relations. The instructors can use Pace University extended Protégé IDE to declare and apply custom relations in a single document. The instructor teaching experience is also

coded in the KG to better support custom learning order by students with different backgrounds as described in section 4.10.2. This research also used the Pace University's extended version of OWLViz, which aided in the knowledge graph development of visualizing the concepts and their relations to each other. The ontology namespace and header contains the follow information from Figure 74. The first two declarations identify the namespace associated with this ontology. The first makes it the *default* namespace, stating that prefixed qualified names refer to the current ontology. The second identifies the base URI for this document. The third identifies rdf: prefix refers to things drawn from the namespace called <http://www.w3.org/1999/02/22-rdf-syntax-ns#>. The fourth namespace declaration says that in this document, elements prefixed with owl: should be understood as referring to things drawn from the namespace called <http://www.w3.org/2002/07/owl#>. This is a conventional OWL declaration, used to introduce the OWL vocabulary. The next two customized namespace declarations such as xmlns:pace represents the element prefix for identifying the customized individuals that represent the different learning levels and order for this ontology. The xmlns:rel namespace represents the customized relations. The next two namespace declarations make similar statements about the XML Schema datatype (xsd:) and RDF Schema (rdfs:) namespace. Once namespaces are established we normally include a collection of assertions about the ontology grouped under an owl:Ontology tag. These tags support such critical housekeeping tasks as comments, version control and inclusion of other ontologies. The rdf:about attribute provides a name or reference for the ontology. Where the value of the attribute is "", the standard case, the name of the ontology is the base URI of the owl:Ontology element. Typically, this is the URI of the document containing the

ontology. An exception to this is a context that makes use of `xml:base` which may set the base URI for an element to something other than the URI of the current document. `rdfs:comment` provides the obvious needed capability to annotate an ontology. `owl:priorVersion` is a standard tag intended to provide hooks for version control systems working with ontologies. `owl:imports` provides an include-style mechanism. `owl:imports` takes a single argument, identified by the `rdf:resource` attribute. Figure 74 declares the prefix names that are commonly used throughout this specification and custom namespaces for this tutorial system.

```
<rdf:RDF xmlns="http://csis.pace.edu/semweb/webTutorial.owl#"
  xml:base="http://csis.pace.edu/semweb/webTutorial.owl"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:pace="http://csis.pace.edu/semweb#"
  xmlns:rel="http://www.pace.edu/rel-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <owl:Ontology rdf:about="http://csis.pace.edu/semweb/webTutorial.owl">
    <rdfs:label>Web Technology Concepts</rdfs:label>
    <owl:versionInfo>WebTutorial.owl 1.0</owl:versionInfo>
    <rdfs:comment>Web Tutorial Knowledge Representation</rdfs:comment>
  </owl:Ontology>
```

Figure 35 Declarations of the Standard Prefix Names and Custom Namespaces

The New Custom Relations section for the Web Tutorial Ontology requires the customized namespace “rel” prefix in order to represent the NewRelation such as “include”, “partOf”, “ref”, “implement”, “implementBy”, “level”, and “name”. The “NewRelation” element is used to declare the custom relations in the knowledge graphic. The “rdf:about” is used to provide an absolute URI for resources. The custom relations can be used to define other types of functional and non functional relations.

```

<!--
//
// Relations
//
-->
<rel:NewRelation rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#include"/>
<rel:NewRelation rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#partOf"/>
<rel:NewRelation rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#ref"/>
<rel:NewRelation rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#implement"/>
<rel:NewRelation rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#implementedBy"/>
<rel:NewRelation rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#level"/>
<rel:NewRelation rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#name"/>

<!--
//
//

```

Figure 75 Knowledge Representation of Customized New Relations

4.6 Web Tutorial Learning Order Specification

First, it is important to clearly identify the target audience, such as identifying the learners, their skill levels and what the learners should be able to do and achieve during the learning process. In Table X, describes the language features for learning order. The NamedIndividual language feature captures the URL Individual names.

For example, `rdf:about= http://csis.pace.edu/semweb/webTutorial.owl#Beginner`, and
`rdf:about= http://csis.pace.edu/semweb/webTutorial.owl#Intermediate`.

In this prototype only two NamedIndividual where used but when developing a new subject, the instructor can add to this language feature. For example, the instructor may want to add a learning level for an advance user or an expert user. The `&pace:LearningOrder` represents an ampersand (&) which is the escape character that begins any entity reference. In this case it would be the pace referencing the learning order. The `pace:level` represents the numeric number of the NamedIndividual. For example the NamedIndividual is Beginner with the `pace:level` of “0” expressed as `<pace:level>0</pace:level>`. The `pace:name` represents each learning level by name. For

example `<pace:name>Beginner</pace:name>`. The `pace:ref` represents each resource concept URL by referencing the URL location of each resource. For example, `<pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl# WebPage"/>`, represents the URL location of the resource concept for “WebPage” at the `http://csis.pace.edu/semweb/webTutorial.owl#` location. The general concepts for each learning level is represented in differently for each learner.

Table 8 Knowledge Representation of Learning Order

Language Features	Description
NamedIndividual	Obtains the individual as a named individual if it is indeed named. The individuals for this tutorial are <code>rdf:about=http://csis.pace.edu/semweb/webTutorial.owl#Beginner</code> , and <code>rdf:about=http://csis.pace.edu/semweb/webTutorial.owl#Intermediate</code>
<code>&pace:LearningOrder</code>	The ampersand (&) is the escape character that begins any entity reference. In this case it will be the pace referencing the learning order.
<code>pace:level</code>	The <code>pace:level</code> represents the numeric number of the NamedIndividual. The <code>pace:level</code> of Beginner is “0”, and the intermediate level is “1”.
<code>pace:name</code>	The <code>pace:name</code> represents each learning level by name. For example <code><pace:name>Beginner</pace:name></code> .
<code>pace:ref</code>	The <code>pace:ref</code> represents each resource concept URL by referencing the URL location of each resource. For example, <code><pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebPage"/></code> .

Figure 76, is a knowledge representation of the learning order structure as described in Table 8.

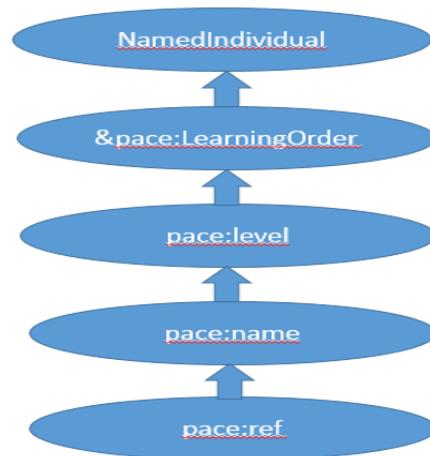


Figure 76 Knowledge Representation of Learning Order Structure

Scenario – The tutorial system needs to store two types of information about the Beginner and the Intermediate learner.

- Store information about a learner who is a Beginner
- Beginner contains information such as: NameIndividual, pace:LearningLevel, pace:level, pace:name, pace:ref
- Store information about a learner who is an Intermediate
- Beginner contains information such as: NameIndividual, pace:LearningLevel, pace:level, pace:name, pace:ref
- NameIndividual in this domain includes a Beginner and Intermediate learner
- pace:Level and pace:LearningOrder is the unique identifier that differentiates the Beginner learner from Intermediate learner

The Beginner level is a user whose starting point is at the beginning of the course learning process because they have no knowledge of the course content. The Intermediate

level is a user that starts at a different starting point position in the knowledge space because they have some knowledge of the course content. Although we have two different learning types the course content material remains the same for both learner, which allows for reusability of learning material.

```

<!-- http://csis.pace.edu/semweb/webTutorial.owl#Beginner -->
<owl:NamedIndividual rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Beginner">
  <rdf:type rdf:resource="&space;LearningOrder"/>
  <pace:level>0</pace:level>
  <pace:name>Beginner</pace:name>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebPage"/>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTTP"/>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#SessionManagement"/>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebArchitecture"/>
</owl:NamedIndividual>

```

Figure 77 Defining Learning Order for Beginner

```

<!-- http://csis.pace.edu/semweb/webTutorial.owl#Intermediate -->
<owl:NamedIndividual rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Intermediate">
  <rdf:type rdf:resource="&space;LearningOrder"/>
  <pace:level>1</pace:level>
  <pace:name>Intermediate</pace:name>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTTP"/>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#SessionManagement"/>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebArchitecture"/>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebPage"/>
</owl:NamedIndividual>

```

Figure 78 Defining Learning Order for Intermediate

Figure 79, below illustrates a sample of the tutoring system web page hierarchy for both the Beginner and the Intermediate student just as it is designed in the learning order specifications.

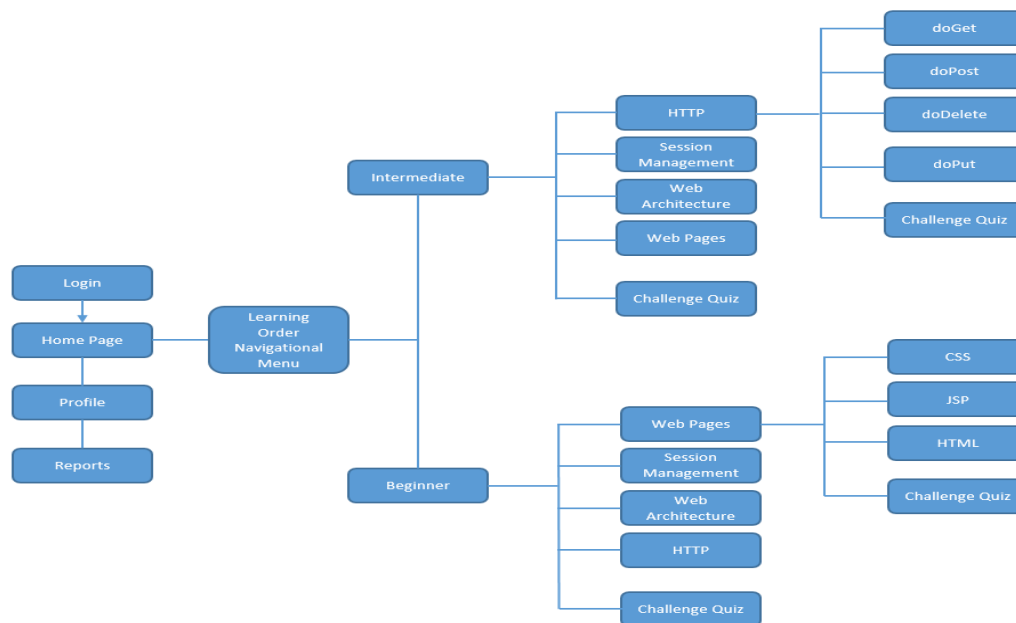


Figure 36 A Sample of Tutoring System Learning Order Diagram

4.7 Web-Based Technology Implementation

The Web-Based components to this tutoring system is based on Tomcat and Spring MVC Framework, with a customized DispatcherServlet which uses PaceJena API along with Data Access Objects with MYSQL to retrieve data from the Knowledge Graph. The Spring annotated controllers accept and process requests for various web resources. When a request is dispatched to the web server, Spring MVC runs the appropriate controller code. This code accesses a predetermined model object and responds to the request. The web view is a rendered Java Server Page. The JavaServer Pages Standard Tag Library is used to perform basic processing on data within a Java Server Page. There are five controllers in this tutoring system: Home Controller, Learning Topic Controller, Owl Navigator Controller, Quiz Controller, and User Controller. These controllers contains the logic to perform all the operations in a specific web section.

4.7.1 Tomcat Web Server

The main function of the Tomcat server is to act as a container for Java web applications. The concept of a web application was introduced with the release of the Java servlet specification 2.2. According to this specification, “a web application is a collection of servlets, html pages, classes, and other resources that can be bundled and run on multiple containers from multiple vendors.” What this really means is that a web application is a container that can hold any combination of the following list of objects:

- Servlets
- Java Server Pages (JSPs)
- Utility classes
- Static documents, including HTML, images, JavaScript libraries, cascading stylesheets, and so on
- Client-side classes
- Meta-information describing the web application

One of the main characteristics of a web application is its relationship to the ServletContext. Each web application has one and only one ServletContext. This relationship is controlled by the servlet container and guarantees that no two web applications will clash when accessing objects in the ServletContext.

4.7.1.1 Tomcat Configuration

In Figure 80, the server configuration is divided into seven sections, we have the general information, server locations, server options, publishing, timeouts, ports, and MIME mappings. The general information section specifies the server name Tomcat v7 Server at

localhost, the host name is localhost and the runtime environment is Apache Tomcat v7.0, the configuration path is /Servers/Tomcat v7.0 Server at localhost-config. The server location section specifies the server path and deploy path. The server path is .metadata/.plugins/org.eclipse.wst.server.core/tmpCore and the deploy path is wtpwebapps. The server options selected is modules auto reload by default. The publishing section, the automatically publish when resources change was selected. The timeout section specifies the time limited to complete server operations which is set at start 45 seconds and stop 15 seconds. The ports section specifies the selected ports names and numbers used for this application, which are the Tomcat admin port, which is on port number 8005, the HTTP/1.1, which is on port number 8080, and the AJP/1.3, which is on port number 8009. The last section is the MIME mappings, which are configured for this application.

Overview

General Information
Specify the host name and other common settings.

Server name: Tomcat v7.0 Server at localhost
Host name: localhost
Runtime Environment: Apache Tomcat v7.0
Configuration path: /Servers/Tomcat v7.0 Server at localhost-con Browse...

[Open launch configuration](#)

Server Locations
Specify the server path (i.e. catalina.base) and deploy path. Server must be published with no modules present to make changes.

Use workspace metadata (does not modify Tomcat installation)
 Use Tomcat installation (takes control of Tomcat installation)
 Use custom location (does not modify Tomcat installation)

Server path: .metadata/.plugins/org.eclipse.wst.server.core/tmpC Browse...
[Set deploy path to the default value \(currently set\)](#)
Deploy path: wtpwebapps Browse...

Server Options
Enter settings for the server.

Serve modules without publishing
 Publish module contexts to separate XML files
 Modules auto reload by default
 Enable security
 Enable Tomcat debug logging (not supported by this Tomcat version)

Publishing
Modify settings for publishing.

Never publish automatically
 Automatically publish when resources change
 Automatically publish after a build event

Publishing interval (in seconds):
Select publishing actions:
 Update context paths

Timeouts
Specify the time limit to complete server operations.

Start (in seconds):
Stop (in seconds):

Ports
Modify the server ports.

Port Name	Port Number
Tomcat admin port	8005
HTTP/1.1	8080
AJP/1.3	8009

MIME Mappings

Figure 80 Tomcat Server Configuration

4.7.1.2 Tomcat Directory Structure

The container that holds the components of the tutoring system web application is the directory structure in which it exists. Table 11 illustrates the directory structure for tutoring system web application, and a description of what each of its directories should contain. Each of these directories should be created from the application base directory of the web application container, the root directory is where web applications are stored and accessed from the servlet container. The default application base directory for Tomcat is CATALINA_HOME/webapps directory.

Table 9 Tomcat Webapp Tutoring System Directory Structure

Directory	Description
/pacejena	The root directory of the web application. All JSP and HTML files should be stored here. Usually each type of static content is stored in a separate sub-directory (images/, styles/, js/).
/pacejena/webapp/resources	Contains all the folders for CSS files, HTML files, and Image files.
/pacejena/webapp/WEB-INF	Contains all resources related to the application that are not in the document root of the application. This is where your web application deployment descriptor is located (defined in the next section). Note that the WEB-INF directory is not part of the public document. No files contained in this directory can be requested directly by a client.
/pacejena/webapp/WEB-INF/Views	Contains the JSP view template files of the header.jsp, footer.jsp, includes.jsp, home.jsp, registration.jsp, login.jsp, quiz.jsp and score.jsp.
/pacejena/webapp/WEB-INF/spring-database.xml	Contains the configuration connection login resources to the MYSQL database.
/pacejena/webapp/WEB-INF/spring-security.xml	Contains all the intercepts URL patterns for login access, resources access, webjar access, user register access and */** access.
/pacejena/webapp/WEB-	Contains the package to connect to the domain URL

INF/SpringDispatcher-servlet.xml	which is the edu.pace.pacejena, maps to the pacejena, WEB-INF/views, all *.jsp, resources, and webjars.
/pacejena/webapp/WEB-INF/web.xml	Contains the XML files that maps and loads the SpringDispatcher servlet, spring-database, and Spring-Security.

4.7.2 Tutorial POM Descriptor

The POM descriptor stands for “Project Object Model”. It is an XML representation of a Maven project held in a file named pom.xml. The POM contains all the necessary information about a project as well as the configurations of plugins to be used during the build process. In Figure 81, it specifies the overview POM descriptor configuration which contains the following sections; Artifact, Parent, Properties, Modules, Project, Organization, SCM, Issue Management, and Continuous Integration.

Overview

Artifact

Group Id:

Artifact Id:

Version:

Packaging:

Parent

Properties

Modules

Project

Name:

URL:

Description:

Inception:

Organization

SCM

Issue Management

Continuous Integration

Figure 81 Overview POM Descriptor Configuration

The artifact section specifies the Group id: pace-jena, the artifact id: pace-jena, version is a snapshot 0.0.1 snapshot, package used is war. The properties specifies the java version 1.6, spring.version: 3.2.17 release, and the spring.security.version: 3.2.9. release. The project section specifies the name: pace-jena, using the URL <http://maven.apache.org> . In Figure 82 defines a sample of the tutorial system dependencies. And finally, Figure 82 is a sample of the POM.XML file which defines the POM elements used for this tutorial application.



Figure 82 Sample of POM Dependencies

The following elements are needed to create a POM.XML file structure which is shown in Table 10. Figure 11 is a sample of how the POM file was used for this tutoring system.

Table 10 Elements of POM.XML File Structure

Element	Description
project	It is the root element of pom.xml file.
modelVersion	It is the sub element of project. It specifies the modelVersion. It should be set to 4.0.0.
groupId	It is the sub element of project. It specifies the id for the project group.
artifactId	It is the sub element of project. It specifies the id for the artifact (project). An artifact is something that is either produced or used by a project. Examples of artifacts produced by Maven for a project include: JARs, source and binary distributions, and WARs.
version	It is the sub element of project. It specifies the version of the artifact under given group.

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3
  <modelVersion>4.0.0</modelVersion>

  <groupId>pace-jena</groupId>
  <artifactId>pace-jena</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>pace-jena</name>
  <url>http://maven.apache.org</url>

  <properties>
    <java.version>1.6</java.version>
    <spring.version>3.2.17.RELEASE</spring.version>
    <spring.security.version>3.2.9.RELEASE</spring.security.version>
    <cglib.version>2.2.2</cglib.version>
  </properties>

  <dependencies>
    <!-- Spring core & mvc -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-orm</artifactId>
      <version>${spring.version}</version>
      <type>jar</type>
      <scope>compile</scope>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-test</artifactId>
      <version>${spring.version}</version>
      <type>jar</type>
      <scope>test</scope>
    </dependency>
  </dependencies>

```

Figure 83 Sample of POM.XML File

4.7.3 Tutorial Deployment Descriptor

The deployment descriptor is an XML file named web.xml. The deployment descriptor is located in the WEB-INF/ directory within the main application directory. It contains configuration information for the entire web application. It provides the configuration options for web applications such as defining mapping between URL and servlet class. The information that is contained in the tutoring system deployment descriptor includes the following elements in Table 11.

Table 11 Deployment Descriptor Elements

Elements	Description
<code><display-name>pace-jena</display-name></code>	Can be used to provide a name that will be displayed by a GUI interface, if one exists. The <code><display-name></code> is pace-jena.
<code><listener-class> org.springframework.web.context.ContextLoaderListener</listener-class></code>	Name of the class that responds to a Web application event.
<code><servlet-name> SpringDispatcher</servlet-name></code>	Contains the name of the servlet, which is used to reference it from elsewhere in Web.xml.
<code><servlet-class> org.springframework.web.servlet.DispatcherServlet</servlet-name></code>	Contains the fully qualified class name of the servlet. All servlet classes will be contained in the WEB-INF\classes directory, so that the servlet name is relative to that directory.
<code><load-on-startup>1</load-on-startup></code>	Allow's you to indicate the order in which servlets should be loaded when the server is started up. This element should contain a positive integer; servlets will be loaded from lowest to highest value according to this element. If you do not include a <code><load-on-startup></code> element, servlets can be loaded in any order. Note that if you use the <code><jsp-file></code> element in combination with a <code><load-on-startup></code> element, the JSP referred to will be precompiled instead of being read on-the-fly when accessed.
<code><servlet-mapping></servlet-mapping></code>	Must correspond to a <code><servlet-name></code> element in a <code><servlet></code> element. This

	indicates which servlet is being mapped to the URL.
<context-param></context-param>	Element contains the declaration of a Web application's servlet context initialization parameters.
<param-name> contextConfigLocation</param-name>	Defines the name of this attribute.
<param-value> /WEB-INF/spring-database.xml /WEB-INF/spring-security.xml </param-value>	Defines a String value for this attribute.
<filter-name> springSecurityFilterChain </filter-name>	The name of the filter to which you are mapping a URL pattern or servlet. This name corresponds to the name assigned in the <filter> element with the <filter-name> element.
<filter-class> org.springframework.web.filter.Delegating Filterproxy </filter-class>	The fully-qualified class name of the filter.
<url-pattern>/*</url-pattern>	Contains the URL pattern that will be added to your host name/application name URL and used as a unique URL for the servlet.
<session-config> <session-timeout>1</session- timeout> </session-config>	Element allows you to specify how long a session with this Web application lasts. <session-config> contains a single child element, <session-timeout>, whose value should be an integer representing the session length in minutes.

```

WEB.XML
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-
app_2_4.xsd">
  <display-name>pace-jena</display-name>

  <!-- Spring MVC -->
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-
class>
  </listener>
  <servlet>
    <servlet-name>SpringDispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>SpringDispatcher</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
      /WEB-INF/spring-database.xml,
      /WEB-INF/spring-security.xml
    </param-value>
  </context-param>

  <!-- Spring Security -->
  <filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

  <session-config>
    <session-timeout>30</session-timeout>
  </session-config>

</web-app>

```

Figure 84 Tutoring System Deployment Descriptor: Web.XML File

The tutoring system web.xml file in Figure 65 defines a servlet named “SpringDispatcher”, implemented in “org.springframework.web.servlet.DispatcherServlet” servlet class and maps to url-pattern “/” which denotes the context root of the webapp to pace-jena and is loaded on startup by the sub element <load-on-startup>. The number inside the <load-on-startup>1</load-on-startup> element tells the servlet container in what sequence the servlets should be loaded. The lower numbers are loaded first. If the value is negative, or unspecified, the servlet container can load the servlet at any time. Figure 85 is a screenshot of the servlet and servlet mapping section from the web.xml file.

```
<servlet>
  <servlet-name>SpringDispatcher</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>SpringDispatcher</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
```

Figure 85 Servlet and Servlet Mapping

The <servlet> element declares the DispatcherServlet. When the dispatcherServlet is initialized, the framework will try to load the application context from a file named “SpringDispatcher-servlet.xml”, located in the /WEB-INF/ directory. The ,servlet-mapping> element of web.xml file specifies what URLs will be handled by the DispatcherServlet. The DispatcherServlet is also responsible for loading a Spring Application Context that used to perform wiring and dependency injection of managed

components. At this point we specify some init parameters to servlet which configure the Application Context. The follow actions take place;

- We register the DispatcherServlet as a Servlet called SpringDispatcher
- We map this Servlet to handle incoming requests (relative to the app path) starting with “/”.
- We use the contextConfiguration init parameter to customize the location for the base configuration XML file for Spring Application Context that is loaded by the DispatcherServlet.

In Figure 86, of the SpringDispatcher-servlet.xml file the `org.springframework.web.servlet.view.InternalResourceViewResolver` is defined as a bean, and is used as an internal resource views resolver, which means it will find the JSP and HTML files in the `/WEB-INF/` folder. The prefix and suffix properties can be set to the view name to generate the final view page URL.


```

<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:util="http://www.springframework.org/schema/util"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.0.xsd
    http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util.xsd">

  <context:component-scan base-package="edu.pace.pacejena.*" />

  <bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix">
      <value>/WEB-INF/views/</value>
    </property>
    <property name="suffix">
      <value>.jsp</value>
    </property>
  </bean>

  <mvc:resources mapping="/resources/**" location="/resources/" />
  <mvc:resources mapping="/webjars/**" location="classpath:/META-INF/resources/webjars/" />

  <bean id="jacksonMessageChanger"
    class="org.springframework.http.converter.json.MappingJacksonHttpMessageConverter">
    <property name="supportedMediaTypes" value="application/json" />
  </bean>

  <bean
    class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter">
    <property name="messageConverters">
      <util:list id="beanList">
        <ref bean="jacksonMessageChanger" />
      </util:list>
    </property>
  </bean>
</beans>

```

Figure 86 SpringDispatcher – servlet.xml File

The context-param element is the subelement of web-app, which is used to define the initialization parameter in the application scope. The parm-name is <parm-name>contextConfigLocation</parm-name> which defines the parm-value from the /WEB-INF/spring-database.xml and the /WEB-INF/spring-security.xml and are the subelements of the context-param. Figure 87 is a screenshot of this section from the web.xml file.

```

<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    /WEB-INF/spring-database.xml,
    /WEB-INF/spring-security.xml
  </param-value>
</context-param>

```

Figure 87 Context Parameter

The /WEB-INF/spring-database.xml and the /WEB-INF/spring-security.xml files below initialize the entire application. The spring-database.xml file initializes the application database and the spring-security.xml file initializes the security and makes a bean call to the “pacejenaauthenticationSuccessHandler” file. The spring-security.xml contains the following interceptors-url patterns; “login”, “resources”, “webjars”, “user/register”, and “*/**”.

```

<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

  <bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">

    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://localhost:3306/pacejena" />
    <property name="username" value="root" />
    <property name="password" value="123456" />
  </bean>
</beans>

```

Figure 88 Spring-Database.xml File

```

<beans:beans xmlns="http://www.springframework.org/schema/security"
  xmlns:beans="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security-3.2.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">
  <context:component-scan base-package="edu.pace.pacejena.*" />
  <context:annotation-config />
  <!-- enable use-expressions -->
  <http auto-config="true" use-expressions="true">
    <intercept-url pattern="/login" access="permitAll" />
    <intercept-url pattern="/resources/**" access="permitAll" />
    <intercept-url pattern="/webjars/**" access="permitAll" />
    <intercept-url pattern="/users/register" access="permitAll" />
    <intercept-url pattern="/**" access="authenticated"/>
    <!-- access denied page -->
    <access-denied-handler error-page="/403" />
    <form-login
      login-page="/login"
      default-target-url="/home"
      authentication-failure-url="/login?error"
      username-parameter="username"
      password-parameter="password"
      always-use-default-target="true"
      authentication-success-handler-ref="paceJenaAuthenticationSuccessHandler"/>
    <logout logout-success-url="/login?logout" invalidate-session="true" delete-cookies="JSESSIONID" />
    <!-- enable csrf protection -->
    <csrf/>
  </http>
  <!-- Select users and user_roles from database -->
  <!--
  <authentication-manager>
    <authentication-provider>
      <jdbc-user-service data-source-ref="dataSource"
        users-by-username-query=
          "select USER_ID, PASSWORD, true from USERS where USER_ID=?"
        authorities-by-username-query=
          "select USER_ID, ROLE from USER_ROLES where USER_ID=?" />
    </authentication-provider>
  </authentication-manager>
  -->
  <authentication-manager>
    <authentication-provider user-service-ref="userDetailsService" />
  </authentication-manager>
  <beans:bean id="paceJenaAuthenticationSuccessHandler"
    class="edu.pace.pacejena.auth.PaceJenaAuthenticationSuccessHandler" />
  <beans:bean id='userDetailsService' class='edu.pace.pacejena.service.UserDetailsServiceImpl'/>
</beans:beans>

```

Figure 89 Spring-Security.xml File

4.7.4 Tutorial Web Dispatcher Customization

The Spring Web model-view-controller (MVC) framework is designed around a DispatcherServlet that handles all the HTTP Servlet requests and responses. The request processing workflow of the Spring Web MVC DispatcherServlet for the tutoring system is illustrated in the following diagram:

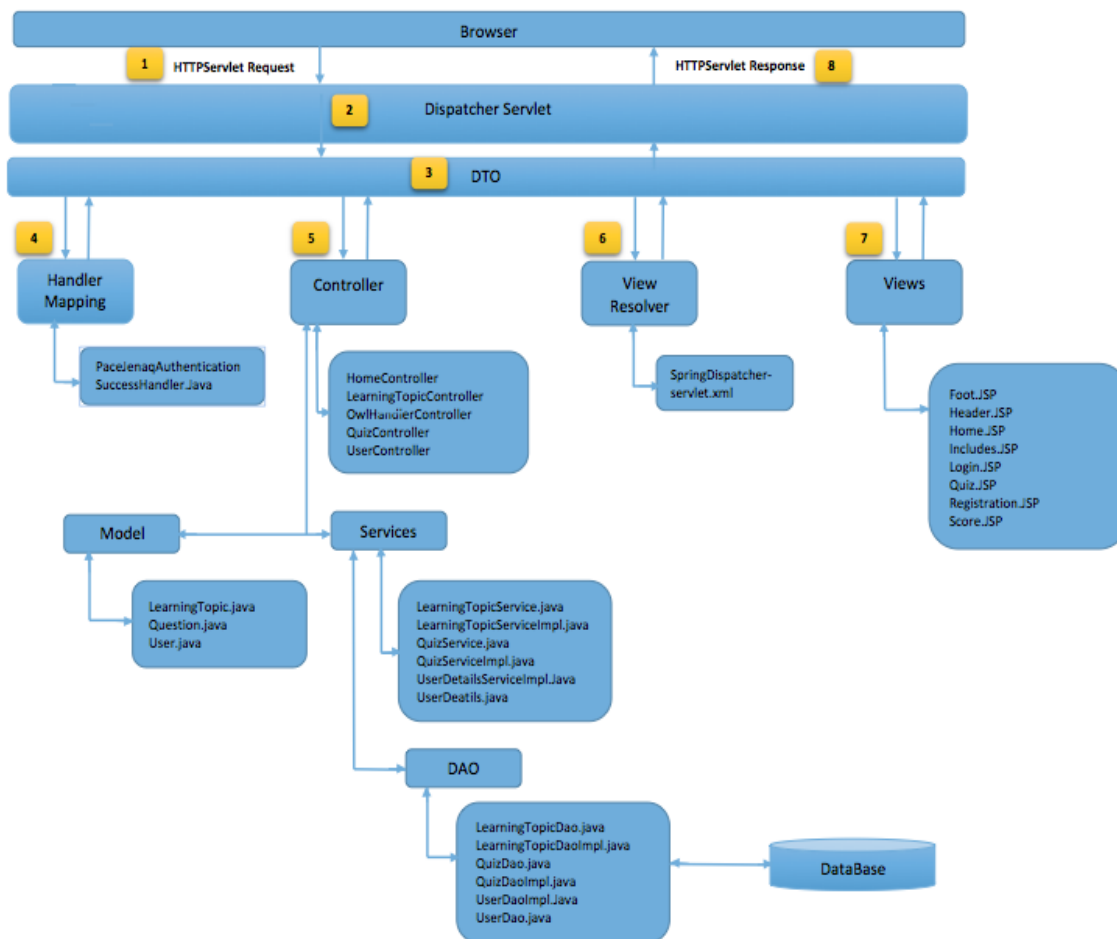


Figure 90 Tutoring System DispatcherServlet Diagram

Table 12 explains the tutoring system application elements.

Table 12 Tutoring System Elements

No.	Packages	Description
1	HTTPServlet Request	<p>public interface HttpServletRequest extends ServletRequest</p> <p>Extends the ServletRequest interface to provide request information for HTTP servlets.</p> <p>The servlet container creates an HttpServletRequest object and passes it as an argument to the servlet's service</p>

		methods (doGet, doPost, etc).
2	DISPATCHERSERVLET or DEPLOYMENT DESCRIPTOR:WEB.XML	Java web applications use a deployment descriptor file to determine how URLs map to servlets, which URLs require authentication, and other information. This file is named web.xml, and resides in the app's WAR under the WEB-INF/ directory. web.xml is part of the servlet standard for web applications.
2	SpringDispatcher	The web.xml file contains a servlet called SpringDispatcher which will load on startup. This file contains the package edu.pace.pacejena which will invoke the Handler mapper file called PaceJena-AuthenticationSuccessHandler.Java.
2	Spring-Database.XML	The web.xml file contains parameters to locate and establish connection to the database xml file.
2	Spring-Security.XML	The web.xml file contains parameters to locate and establish connection to the security xml file.
2	pace-jena	The web.xml file contains parameters to locate and retrieve the pace-jena domain.
3	DTO (Data Transfer Object)	A DTO is a data transfer object that carries data between processes. The motivation for its use is that communication between processes is usually done resorting to remote interfaces like web services, where each call is an expensive operation. Because the majority of the cost of each call is related to the round-trip time between the client and the server, one way of reducing the number of calls is to use an object (the DTO) that aggregates the data that would have been transferred by the several calls, but that is served by one call only.
4	HANDLER MAPPER	
	PaceJenaAuthenticationSuccess Handler.Java	This file is the handler mapper for handling the login authentication request and execute

		the webTutorial.owl file that will map request to the user's learner level and learner order.
5	CONTROLLER	
	HomeController.java	The HomeController maps to the login page.
	LearningTopicController.java	The LearningTopicController gets the topic and content of the topic to be presented to the LearningTopicService.
	OwlHandlerController.java	The OwlHandlerController creates an ArrayList to supports dynamic arrays that can grow as needed. Standard Java ArrayList are fixed and not dynamic.
	QuizController.java	The QuizController is a template that creates the ModelandViewer for the retrieval of the dynamic questions.
	UserController.java	The UserController is a Model and Viewer that maps to the registration page.
5	SERVICE	
	LearningTopicService.java	The LearningTopicService is used to load, save, and update Learning Topics.
	LearningTopicServiceImpl.java	The LearningTopicServiceImpl is used to implement the LearningTopicService to prepare it to be saved in the database.
	QuizService.java	The QuizService is used to list the questions in the quiz module.
	UserDetailsServiceImpl.java	The UserDetailsServiceImpl implements the UserDetailsService to retrieve Username then is mapped to UserDetails.
	UserService.java	The UserService is used to find users that are register and update user.
5	MODEL	
	LearningTopic.java	The LearningTopic gets the topic and content and prepares to model the information for the viewer.

	Question.java	The Question models the questions and question choices by questionID.
	User.java	The User.java implements the user details from the registration form.
5	DAO	
	LearningTopicDao.java	The LearningTopicDao is used to insert, get, and update a topic from the database.
	LearningTopicDaoImpl.java	The LearningTopicDaoImpl implements LearningTopicDao to map to the database to insert, get and update the topic and content in the database.
	QuizDao.java	The QuizDao retrieves the questions from the database.
	QuizDaoImpl.java	The QuizDaoImpl is use as a template for the questions to be displayed and to save the scores.
	UserDao.java	The UserDao gets, delete, and updates the userID in the database.
	UserDaoImpl.java	The UserDaoImpl injects and inserts the user information from the registration form into the MYSQL user database.
6	VIEW RESOLVER	
	SpringDispatcher-Servlet.xml	The SpringDispatcher-Servlet is used to map the WEB-INF/views of all the JSP files also maps to the resources and webjars.
7	VIEWS	
	Foot.jsp	This is the foot template for all web pages.
	Header.jsp	This is the header template for all web pages.
	Home.jsp	This is the home page that will appear after user as authenticated into the system.
	Includes.jsp	The includes file directive includes the header and footer JSP file.

	Login.jsp	This is the Login form to authenticate an existing user.
	Quiz.jsp	This is a quiz template that is used dynamically for presenting quiz questions.
	Registration.jsp	This is the registration form to create a new user account login and profile.
	Score.jsp	This is score template that is used dynamically for presenting each user with their score results.
8	HttpServletResponse	<p>public interface HttpServletResponse extends ServletResponse</p> <p>Extends the ServletResponse interface to provide HTTP-specific functionality in sending a response. For example, it has methods to access HTTP headers and cookies.</p> <p>The servlet container creates an HttpServletResponse object and passes it as an argument to the servlet's service methods (doGet, doPost, etc).</p>

All actions here surround the front controller, which is the DispatcherServlet. This DispatcherServlet and all other components are part of the WebApplicationContext of Spring. Following steps illustrate the process depicted above.

- Client sends HttpServletRequest to DispatcherServlet (via web container)
- DispatcherServlet requests HandlerMapping for a proper controller against this client request
- HandlerMapping returns appropriate controller information
- DispatcherServlet requests that Controller for action

- Controller (business logic is either implemented here or service is invoked from here) processes the request and returns ModelAndView object with logical view name
- DispatcherServlet requests ViewResolver for the actual view implementation
- ViewResolver returns view information
- DispatcherServlet requests View to prepare the response with the input Model
- View returns rendered response
- DispatcherServlet returns HTTPResponse to the Client (via web container)

4.7.5 Tutorial Application File Structure

The tutorial system application file structure in Figure 92, illustrates the constructs of where the files are located.

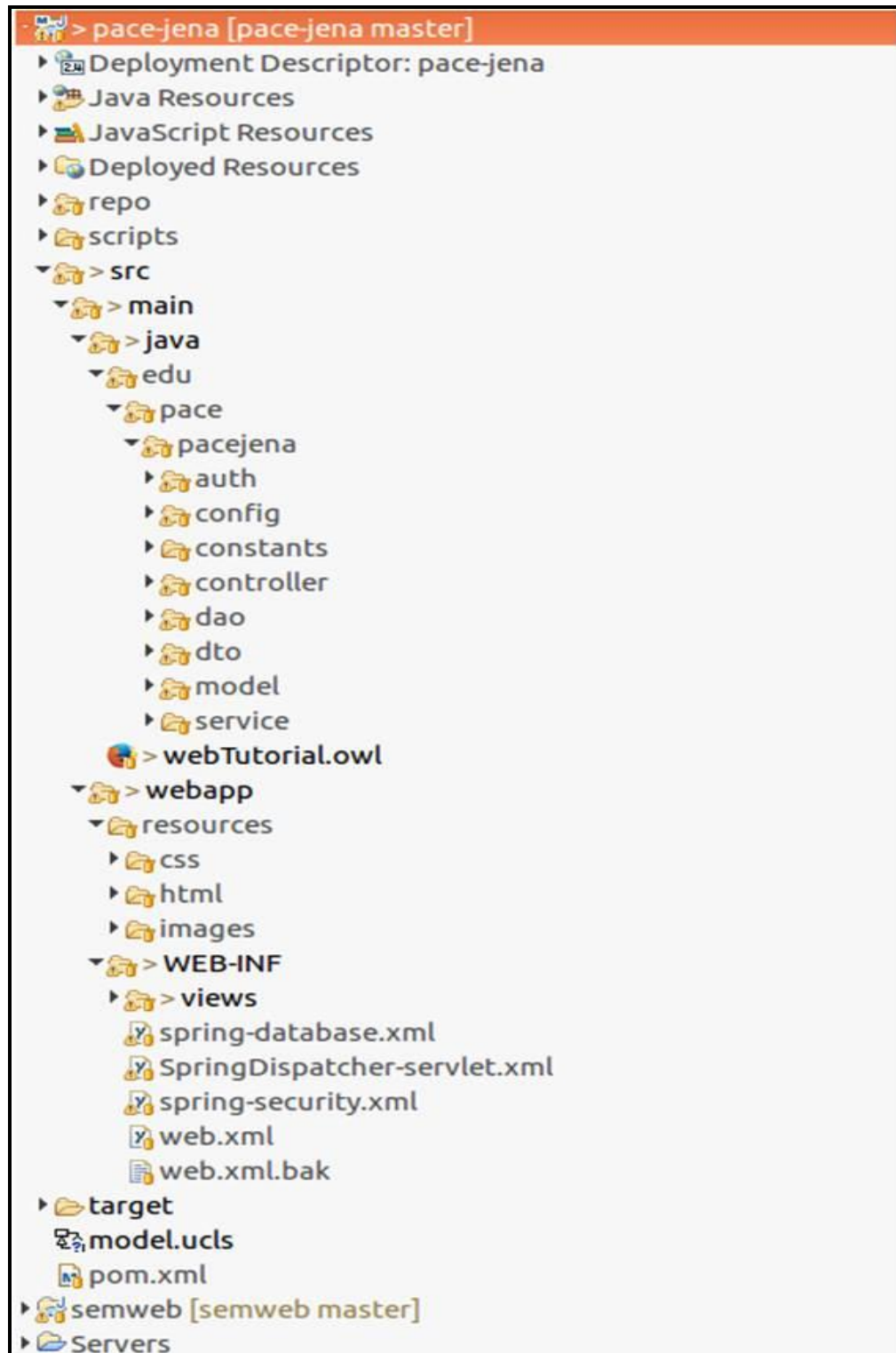


Figure 92 Tutoring System Application File Structure

4.7.6 Tutorial Annotation Web-Based Controllers

User Controller

The User Controller is invoked when the user needs to create a user account for the tutoring system. In Figure 93 the `@RequestMapping` has a value which is equal to `/register` and the RequestMethod is called to GET and POST the Model and View which is resolved by the Dispatcher Servlet. User whom are added successfully to the tutoring system will be redirected to the login page.

```

package edu.pace.pacejena.controller;
import org.springframework.beans.factory.annotation.Autowired;

@Controller
@RequestMapping(value = "/users")
public class UsersController {

    @Autowired
    private UserService userService;

    @RequestMapping(value = "/register", method = RequestMethod.GET)
    public ModelAndView showRegistrationForm(WebRequest request, Model model) {
        UserDto user = new UserDto();
        model.addAttribute("user", user);
        return new ModelAndView("registration");
    }

    @RequestMapping(value = "/register", method = RequestMethod.POST)
    public String registerUserAccount(@ModelAttribute("user") UserDto user,
        BindingResult result, WebRequest request, Errors errors, final RedirectAttributes redirectAttributes) {
        User registered = null;
        if (!result.hasErrors()) {
            registered = userService.register(user);
        }
        if (registered == null) {
            redirectAttributes.addFlashAttribute("css", "warning");
            redirectAttributes.addFlashAttribute("msg", "User already exists!");
        }
        else if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("css", "danger");
            redirectAttributes.addFlashAttribute("msg", "Error adding user!");
        }
        else {
            redirectAttributes.addFlashAttribute("css", "success");
            redirectAttributes.addFlashAttribute("msg", "User added successfully! You will be redirected to the login page in 5 seconds.");
        }
        //return new ModelAndView("registration", "user", user);
        return "redirect:/users/register";
    }
}

```

Figure 93 User Controller

Home Controller

The Home Controller is invoked when the user logs into the tutoring system. In Figure 94, the `@RequestMapping` has a value equal to `/login` and the RequestMethod is called to GET Authentication `auth = SecurityContextHolder.getContext().getAuthentication()` and displays the Model and View to the home page.

```

package edu.pace.pacejena.controller;
import java.io.IOException;

@Controller
public class HomeController {

    @RequestMapping(value="/")
    public ModelAndView test(HttpServletRequest response) throws IOException{
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        if (!(auth instanceof AnonymousAuthenticationToken)) {
            /* The user is logged in :) */
            return new ModelAndView("home");
        }
        return new ModelAndView("login");
    }

    @RequestMapping(value="/home")
    public ModelAndView home(HttpServletRequest response) throws IOException{
        return new ModelAndView("home");
    }

    @RequestMapping(value = "/login", method = RequestMethod.GET)
    public ModelAndView login(@RequestParam(value = "error", required = false) String error,
        @RequestParam(value = "logout", required = false) String logout,
        final RedirectAttributes redirectAttributes) {
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        if (!(auth instanceof AnonymousAuthenticationToken)) {
            /* The user is logged in :) */
            return new ModelAndView("home");
        }
        ModelAndView model = new ModelAndView();
        if (error != null) {
            // model.addObject("error", "Invalid username and password!");
        }

        if (logout != null) {
            model.addObject("msg", "You've been logged out successfully.");
        }
        model.setViewName("login");
        return model;
    }
}

```

Figure 94 Home Controller

Learning Topic Controller

The Learning Topic Controller is invoked when the user logs into the tutoring system. In Figure 95, the `@RequestMapping` which as a value equal to `/topic` makes a `RequestMethod` call to `GET` and load the course topics and content to the home page. The `@RequestMapping` which as a value equal to `/save` makes a `RequestMethod` call to `POST` the `getTopic` and `get Content` from the `MYSQL` database. The `learningTopicService` makes updates to the `getTopic` and `getContent`.

```

package edu.pace.pacejena.controller;

import org.springframework.beans.factory.annotation.Autowired;

@Controller
@RequestMapping(value = "topic")
public class LearningTopicController {
    @Autowired
    LearningTopicService learningTopicService;

    @ResponseBody
    @RequestMapping(value = "/save" , method = RequestMethod.POST)
    public String save(@RequestBody TopicDto dto) {
        System.out.println(dto.getTopic());
        System.out.println(dto.getContent());
        if(StringUtils.isEmpty(learningTopicService.loadTopicContent(dto.getTopic()))){
            learningTopicService.saveTopicContent(dto.getTopic(), dto.getContent());
        } else {
            learningTopicService.updateTopicContent(dto.getTopic(), dto.getContent());
        }
        return "success";
    }

    @RequestMapping(value = "/{topic}" , method = RequestMethod.GET)
    public @ResponseBody String load(@PathVariable String topic) {
        return learningTopicService.loadTopicContent(topic);
    }
}

```

Figure 95 Learning Topic Controller

Owl Navigator Controller

The Owl Navigator Controller is invoked when the user logs into the tutoring system to retrieve their profile with their learning order. In Figure 96, the @RequestMapping which as a value /parent/{nodeName} and /children/{nodeName} makes a RequestMethod call to GET the getParent and getChildren.

```

package edu.pace.pacejena.controller;

import java.util.ArrayList;

@Controller
@RequestMapping(value = "/owl")
public class OwlNavigatorController {
    @RequestMapping(value = "/children/{nodeName}", method = RequestMethod.GET)
    public @ResponseBody List getChildren(HttpServletRequest request, @PathVariable String nodeName) {
        List<String> children = new ArrayList<String>();
        OntologyNode treeNode = (OntologyNode) request.getSession().getAttribute(PaceJenaConstants.ONTOLOGY_ROOT_NODE_SESSION_KEY);
        if(StringUtils.isEmpty(nodeName) || nodeName.equals("root")){
            List<OntologyNode> nodes = treeNode.getChildren();
            for (OntologyNode node : nodes) {
                String name = node.getName();
                children.add(name);
            }
        } else {
            OntologyNode nextChild = treeNode.findChildByName(nodeName);
            if (nextChild != null) {
                List<OntologyNode> nodes = nextChild.getChildren();
                for (OntologyNode node : nodes) {
                    String name = node.getName();
                    children.add(name);
                }
            }
        }
        return children;
    }

    @RequestMapping(value = "/parent/{nodeName}", method = RequestMethod.GET)
    public @ResponseBody String getParent(HttpServletRequest request, @PathVariable String nodeName) {
        OntologyNode treeNode = (OntologyNode) request.getSession().getAttribute(PaceJenaConstants.ONTOLOGY_ROOT_NODE_SESSION_KEY);
        OntologyNode thisNode = treeNode.findChildByName(nodeName);
        if (thisNode != null && thisNode.getParent() != null) {
            return thisNode.getParent().getName();
        }
        return "";
    }
}

```

Figure 96 Owl Navigator Controller

Quiz Controller

In Figure 97, the Quiz Controller is invoked when the user clicks on the challenge quiz button. The `@RequestMapping` which as a value of `/quiz` and `{qidx} / {ans}` makes a Request Method to GET and loads the `loadQuiz`. The questions and answers are evaluated from the `@RequestMapping` to the `{qidx} / {ans}` which make a call to GET the questions from the quiz database and dynamically display each quiz questions one at a time. Each quiz contains 10 questions.

```

@Controller
@SessionAttributes({"questions", "answers"})
@RequestMapping(value = "/quiz")
public class QuizController {
    @Autowired
    QuizService quizService;

    @Autowired
    UserService userService;

    List<Question> questions = null;
    List<Integer> answers = null;

    @RequestMapping(method = RequestMethod.GET)
    public ModelAndView loadQuiz(HttpServletRequest request, HttpServletResponse response, Model model) throws IOException {
        HttpSession session = request.getSession();
        User authUser = (User) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        questions = quizService.getQuestionSet(authUser.getLearningLevel());
        answers = new ArrayList<Integer>(10);
        model.addAttribute("question", this.questions.get(0));
        model.addAttribute("qidx", new Integer(1));
        return new ModelAndView("quiz");
    }

    @RequestMapping(value="/{qidx}/{ans}", method = RequestMethod.GET)
    public ModelAndView nextQ(HttpServletRequest request, HttpServletResponse response, Model model,
        @PathVariable int qidx, @PathVariable int ans) throws IOException {
        System.out.println("question "+qidx+" ans "+ans);
        HttpSession session = request.getSession();
        User authUser = (User) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        if(this.answers.size()<qidx){
            this.answers.add(ans);
        } else {
            this.answers.set(qidx-1, ans);
        }
        if(this.questions.size()==this.answers.size()){
            int score = 0;
            for(int i=0;i<this.questions.size();i++){
                if(this.answers.get(i).intValue()==this.questions.get(i).getCorrectAns()){
                    score++;
                }
            }
            model.addAttribute("score", new Integer((score*100)/this.questions.size()));
            quizService.saveScore(authUser, score);
            if(((score*100)/this.questions.size())>=70){
                authUser.setLearningLevel(authUser.getLearningLevel()+1);
                userService.update(authUser);
                PaceJena paceJena = (PaceJena)session.getAttribute("PaceJena");
                PaceJenaAuthenticationSuccessHandler.loadOntologyTreeIntoSession(session, paceJena, authUser.getLearningLevel());
            }
            return new ModelAndView("score");
        } else {
            model.addAttribute("question", this.questions.get(qidx));
            model.addAttribute("qidx", new Integer(++qidx));
            return new ModelAndView("quiz");
        }
    }
}

```

Figure 97 Quiz Controller

During the quiz session user will only be allow to answer the quiz question and click on the next button. When the user reaches the last question the submit button will display only for the user to submit his her answers to the quiz module to be evaluated. The quiz evaluation process is an IF, THEN and ELSE statement. The rubric score for each quiz is 100. If the quiz score is less then 70 then student will be recommended to go back to previous topic and re-read the failed subject. If the quiz score is greater then 70 then student will be recommended to go to next topic.

4.8 Adaptive Tutorial Navigational Features

4.8.1 Adaptive Navigational Knowledge Space

The adaptive navigational menu is generated by the knowledge graph according to the user's profile. During users login process the learning profile is retrieved and generates the learner's navigational knowledge space initial starting state. This initial state changes as the student progresses with their learning knowledge. The quiz assessment, which is explained in section 4.5.5, will either progress the students learning or digress the students learning depended on their quiz evaluation outcome. The Knowledge Graph supports integrated assessments, using assessment results to custom student learning order or material, and lets the student freely navigate in the knowledge space from general to specific or the opposite, and following various custom relations. This feature provides the user the ability to navigate the knowledge graph freely in any direction and will give guidance on the concepts in which the user needs to learn and the order.

4.8.2 Adaptive Navigational Knowledge Space Guidance

In this feature the user is sequentially guided through the concept knowledge space. The tutoring system will display the learning object concepts in sequentially order in which he/she needs to learn but is free to choose any display concept and is not limited by its displayed sequential order. The adaptive knowledge graph is used to help the learner find their path in the hyperspace as mention in section 4.5.1. The adaptive navigational feature which guides the user in the knowledge space as been created using buttons that respond to links. These links are stored in the learning topic database table and retrieved from the PaceJena API where the knowledge graph file is read into the Sax Parser and is parsed according to the user's interaction with the tutoring system. The button feature guides the user in the knowledge graph. If the user needs more specific details on a concept then he / she will click on the concept name and the tutoring system will produce a navigation of buttons that only relates to that practical concept and will also have the previous concept named display. If the user needs to go back to the previous concept the navigational pane will have this concept name available so that the user can navigate back up the knowledge graph. These navigational features guide the user to concepts that are specific to concepts that are general.

4.8.3 Adaptive Navigational Knowledge Space Hiding

This method excludes the possibility of visiting pages with no relevant information. Links leading to such pages are hidden or disabled. This restriction avoids the learner from being lost in the knowledge hyperspace. For example, user maybe at a learning concept which is at a generalized level of the domain knowledge but needs to understand

the more specific details of the general topic. The learner will then navigate the knowledge graph which will display only the more specific subtopics while the generalized super topics are hidden except for the main concept topic the user is interested in understand. This will enable the learner to navigate back up the knowledge graph and display the generalized super topics and hide the specialized subtopics. This feature helps the user stay focus on only the topics of interest and not topics that are unrelated. This features helps in learning specific concepts.

4.8.4 Adaptive Navigational Knowledge Organization

The adaptive linked data within the knowledge graphic are structured to organize concepts according to the domain knowledge model, the representation of its hierarchy and the relations between concepts. Currently the tutorial system is setup to display the navigational list in the browser interface at the left side pane window. The navigational feature of the links are displayed as buttons with the concept names displayed in the buttons. The collection of the data from the knowledge graph is organized to display the concepts the way it is delivered from the PaceJena API where the knowledge graph is read into the sax parser for which it will trigger the event that has been requested by the user. The organization of this information is depended on what the user is requesting from the navigational window pane.

4.8.5 Adaptive Navigational Knowledge Assessment

This method is used at the completion of each challenge quiz. The tutoring system will evaluate the quiz results by the outcome of their score. If student receives less than a 70 then the system will navigate the student back to the current topic to review and try again. If great then 70 then the system will navigate the student to the next learning topic. This is further explained in quiz integration section.

4.9 Managing Student Learning Profiles

The student's learning profile will be managed from the web-based interface from PaceJena Tutoring System. The user account management area in PaceJena Tutoring System will only be accessible to the system administrator and instructor. The system administrator will be able to view, add, edit, delete, disable, and enable the user account. In Figure 97, the changes to this information will reflect the "USERS" table from the "pacejena" table. If administrator needs to make changes to the database columns then he/she can use the MYSQL Workbench. The figure below depicts a sample of how this information looks inside MYSQL.

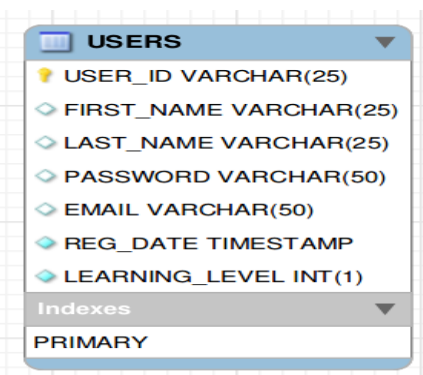


Figure 97 MYSQL USER Database A

#	USER_ID	FIRST_NAME	LAST_NAME	PASSWORD	EMAIL	REG_DATE	LEARNING_LEVEL
1	admin	admin	person	admin	admin@test.com	2016-08-23 16:39:03	0
2	msette	Maria	Sette	msette	msette@yahoo.com	2016-09-16 09:27:12	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 98 MYSQL USERS Database B

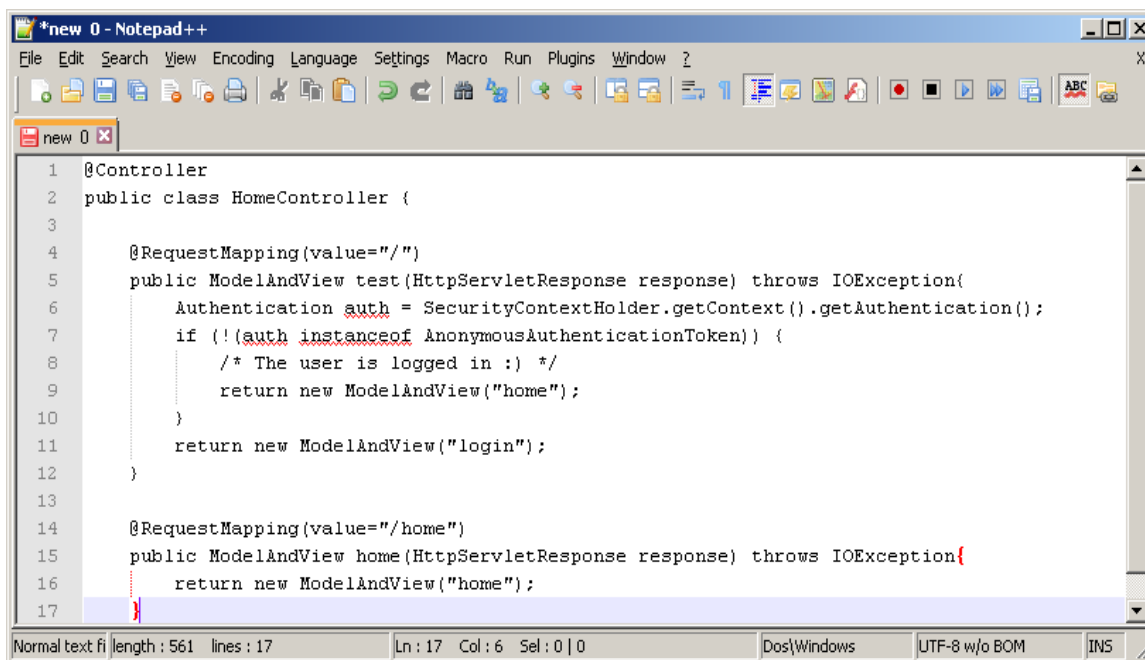
The USERS Database B reflects the following information:

Table 13 USERS Database Table Information

USERS Database Column Names	USERS Database Description
USER_ID	The USER_ID identifies the user in the tutoring systems and authorizes the authentication with the correct login credentials.
FIRST_NAME	The system will register the user's first name during the registration process and store this information in the USERS database.
LAST_NAME	The system will register the user's last name during the registration process and store this information in the USERS database.
PASSWORD	The PASSWORD identifies the user with the USER_ID. The user must provide both user name and password information during the login process in order to access the tutoring system.
EMAIL	The EMAIL address is used also to identify a user. This can be used to reset a user's password. This was not part of this dissertation work.
REG_DATE	The REG_DATE is user log the date and time the user access the system.
LEARNING_LEVEL	The Learning_Level is the critical information that is needed during the registration process because it sets the users learning order.

4.10 Managing Knowledge Object Collection and Organization

The knowledge object collection and organization for this dissertation was developed using a knowledge graph, PaceJena, Spring Web MVC templates, models, views, controllers, and JSP web pages. First it is important to ensure that the Spring MVC application and engine is properly configured and we are able to invoke it from the browser. The code below defines the Spring Web MVC HomeController and the actions required to handle the HTTP GET for the root URL of the project “/“. When someone opens `http://localhost:8080/` from a Web browser, the action below will be called. It returns the “home” view and this means to render a template “home.html” located in the file `src/main/resources/templates/home.html`. And return a new ModelAndView called “login”.



```

1  @Controller
2  public class HomeController {
3
4      @RequestMapping(value="/")
5      public ModelAndView test(HttpServletRequest response) throws IOException{
6          Authentication auth = SecurityContextHolder.getContext().getAuthentication();
7          if (!(auth instanceof AnonymousAuthenticationToken)) {
8              /* The user is logged in :) */
9              return new ModelAndView("home");
10         }
11         return new ModelAndView("login");
12     }
13
14     @RequestMapping(value="/home")
15     public ModelAndView home(HttpServletRequest response) throws IOException{
16         return new ModelAndView("home");
17     }

```

In Figure 99 is a sample of the home.html page customized master web page template, which is formatted and organized with a header that displays the Pace University logo,

the jpeg picture with the PaceJena Tutoring System name. The footer will display the copyright information with the year and the name of contributor. This information will not change and is static. The navigational window pane, which is location on the left side panel, will be dynamic and will contain the knowledge navigational features, which will display the collection of data concepts as data linked buttons.

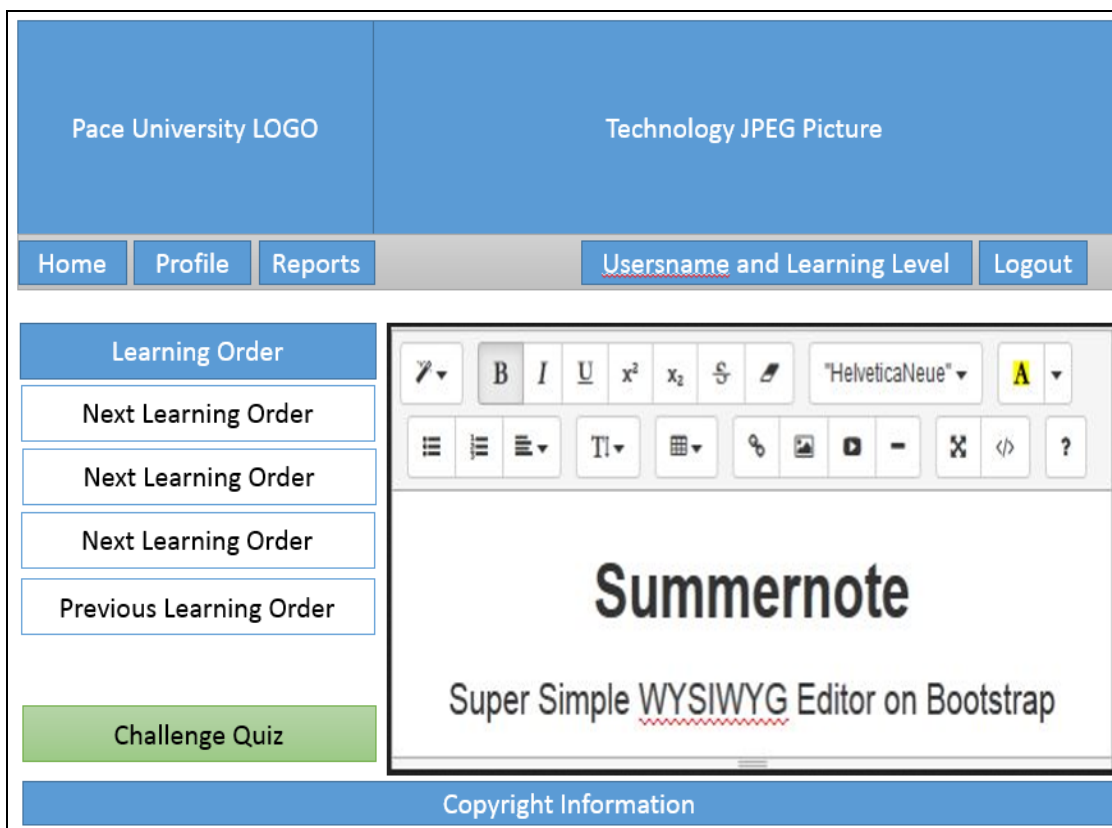


Figure 99 Sample of Customized Master Web page Template

The buttons with the next learning order indicates that these buttons will navigate the user to the next learning level of concepts for that specific topic. The previous learning level will navigate the user back to the previous topic. Also on this panel is the challenge quiz button, which I will explain later. The body area, which is located in the middle, is also dynamic and will display the concepts content information. Depending on which concept is active that information will be displayed as the learning resource in that time and

moment. The body area also contains hidden features, which are to be used by the administrator and instructor only. This is hidden from the students. The student will only see the concept content in this area. As the students navigate their way using the knowledge navigational buttons the concepts will either display concepts that are specific to the topics and hide the general topics except for the previous topic which will be displayed last in this navigation sequence. This allows the students to move freely and will display only the information need to understand a specific topic. The student has the option to that a challenge quiz anytime during his/her learning assignments. The challenge quiz will appear in the body area of the web page. The student will answer 10 multiple choice questions and at the end submit his/her answer to be evaluated by the tutoring system. The tutoring system will respond back with recommendation to either proceed forward to the next topic or to return back to the previous topic. The quiz results will determine the student's next step and using that information to navigate the user to either to proceed forward or to go back. Depending on the state and position in the knowledge graph will determine the collection and organization of the student's knowledge learning path.

4.11 Managing Students Learning Topic: Editing, Saving and Update Process

The instructor can manage the students learning resources with the integrated summernote tool bar editor plugin which has been implemented on the home.html page to allow only the administrator and the instructor the capability to make the necessary changes to learning topics.



Figure 100 Summernote Editor Tool [44]

Figure 100, is a sample of the summernote tool editing bar features.

1. The instructor and system administrator must login to the tutoring system with administration privileges in order to edit and save course content into the tutoring system.
2. Instructor logs into the tutoring system with administrator privileges.
3. Instructor would like to edit and save a topic.
4. In the navigational pane the instructor will have an option to click on a learning concept.
5. In the Instructor mode the instructor will see the HTML editor along with menu options such as edit and save.
6. Instructor will click on a concept word to be edited.
7. Instructor must click on edit to edit the concept content. The edit and save tabs are located below the concept content widow.
8. The instructor will click on the edit button. This will present the HTML editor to the instructor indicating that he/she is in edit mode and can edit the course concept content.
9. When all changes have be completed the instructor must save these changes by clicking on the save icon in the HTML editor menu.
10. The javascript function sends an Ajax request to LearningTopicController using the jquery framework.
11. The LearningTopicController.save (TopicDto dto) is called.

12. The LearningTopicService.save Topic Content (Topic Content) or LearningTopicService.updateTopicContent.save (Topic Content) is called.
13. The LearningTopicDao.insert (Topic Content) or LearningTopicDao.update (Topic Content) is called.
14. The INSERT or UPDATE command in MYSQL is send to the database using the Spring JDBC.
15. The system will respond with either a success or failure response, which will propagate back to the controller.
16. These changes will be stored and saved to the database.

4.12 Managing Instructors New Subject Topics

The management of instructor's new subject topic can be implemented by replacing the current webTutorial.owl file with the new webTutorial.owl file in the directory file structure. In Figure 101, this file structure in Ubuntu is located in the home / workspace / pace-jena / pace-jena / src / main / java.

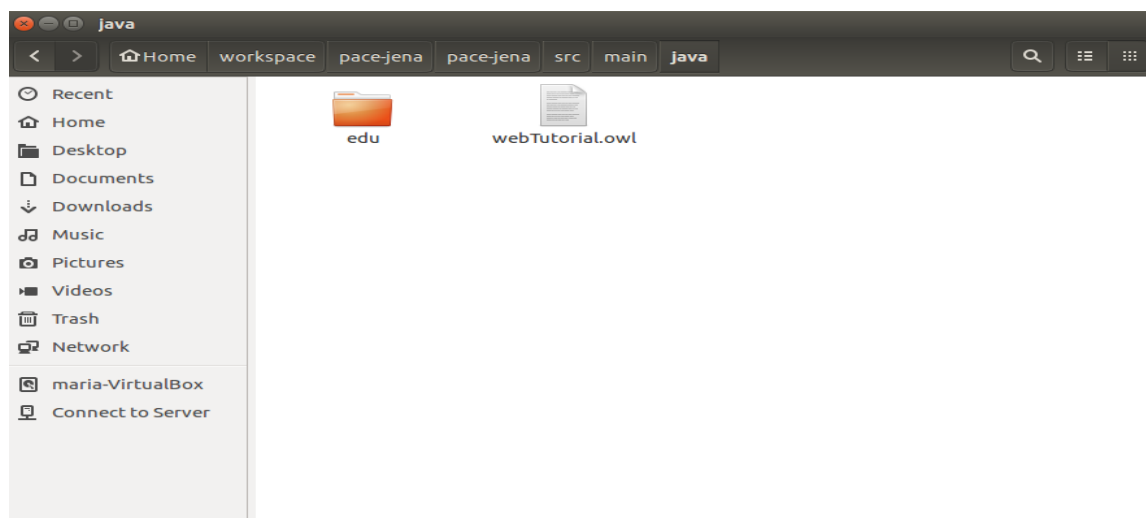


Figure 101 Tutoring System File Structure in Ubuntu

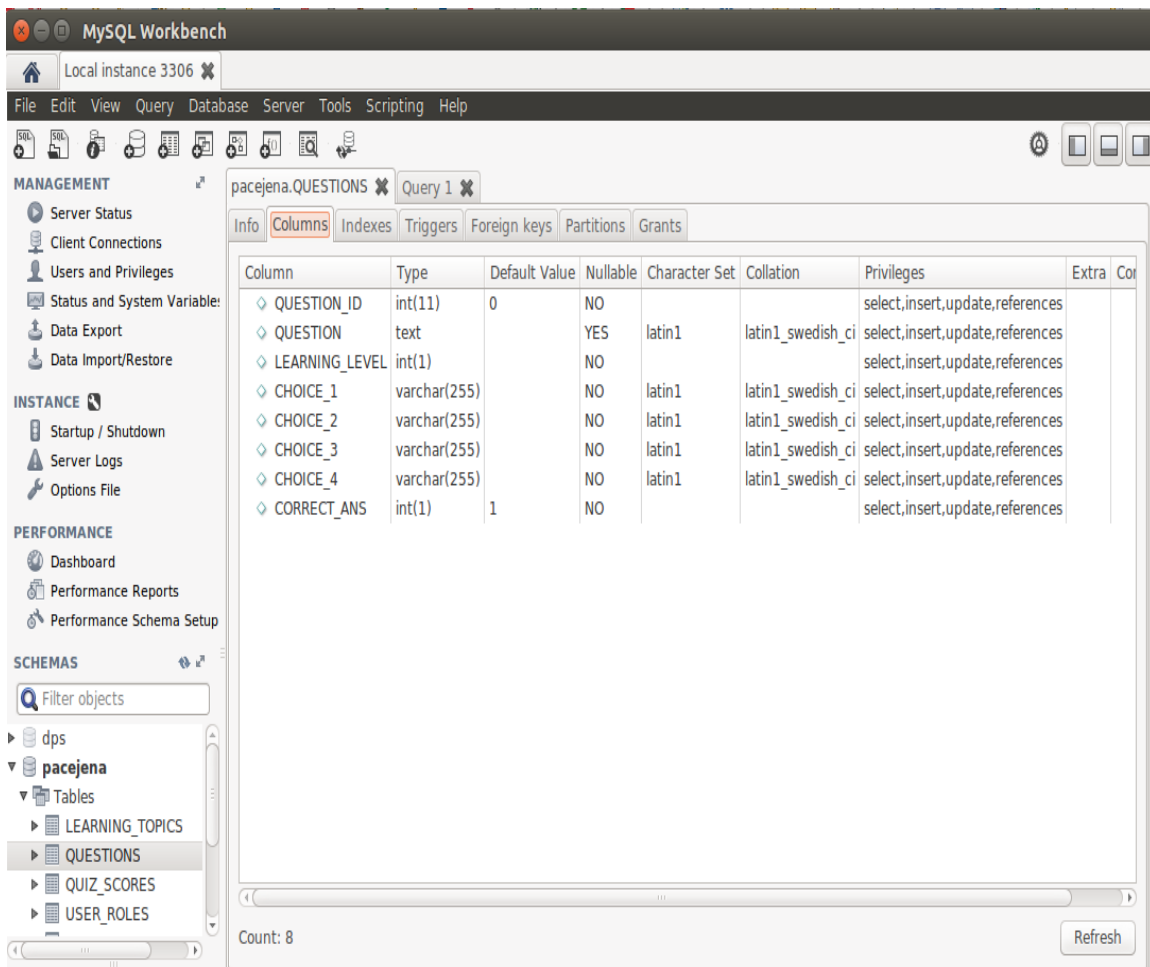
This can also be done in the instructor's management system. The web-based interface for this feature has not been development but can be implemented by providing a feature to browse the file structure and replacing the current OWL file with the new OWL file to be used for a specific course.

4.13 Quiz Development and Integration

4.13.1 Challenge Quiz Overview

The challenge quiz assessment module is integrated, which means combining and aggregating web-based resources, so that they can be collectively usable and reusable. This is also a form of interoperability, which describes the extent to which the quiz module can interpret the quiz assessment results and recommend the learner to the next level of learning order form the web tutorial ontology knowledge graph. It then delivers this information to the servlet that takes the incoming request and delegates the processing of that request to one of a number of handlers. The mapping, which is specific in the dispatcher servlet configuration, prepares the dynamic views back to the user's session. The quiz development and integration in the tutoring system application helps to evaluate and recommend the next step for each student's. The development of the quiz questions and answers are done in XML format and implemented into the MYSQL database. In Figure 102, the quiz question and answer database contains the following information. Question_ID with type int(11), Question type is text, Learning_Level type is int(1), Choice_1, Choice_2, Choice_3, Choice_4 all with type varchar(255) and Correct_Ans type int(1). In Figure 103, the quiz scores table is where the scores are

retained after they have been evaluated by the tutoring system. This database table contains the user_ID is the students user name, score_TS is the date and time, Learning_Level is a number which translates back to the level of the learner, and score is how the tutoring system will evaluate the students performance and determine the next steps.



Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Extra	Cor
QUESTION_ID	int(11)	0	NO			select,insert,update,references		
QUESTION	text		YES	latin1	latin1_swedish_ci	select,insert,update,references		
LEARNING_LEVEL	int(1)		NO			select,insert,update,references		
CHOICE_1	varchar(255)		NO	latin1	latin1_swedish_ci	select,insert,update,references		
CHOICE_2	varchar(255)		NO	latin1	latin1_swedish_ci	select,insert,update,references		
CHOICE_3	varchar(255)		NO	latin1	latin1_swedish_ci	select,insert,update,references		
CHOICE_4	varchar(255)		NO	latin1	latin1_swedish_ci	select,insert,update,references		
CORRECT_ANS	int(1)	1	NO			select,insert,update,references		

Count: 8 Refresh

Figure 102 Quiz Question Database

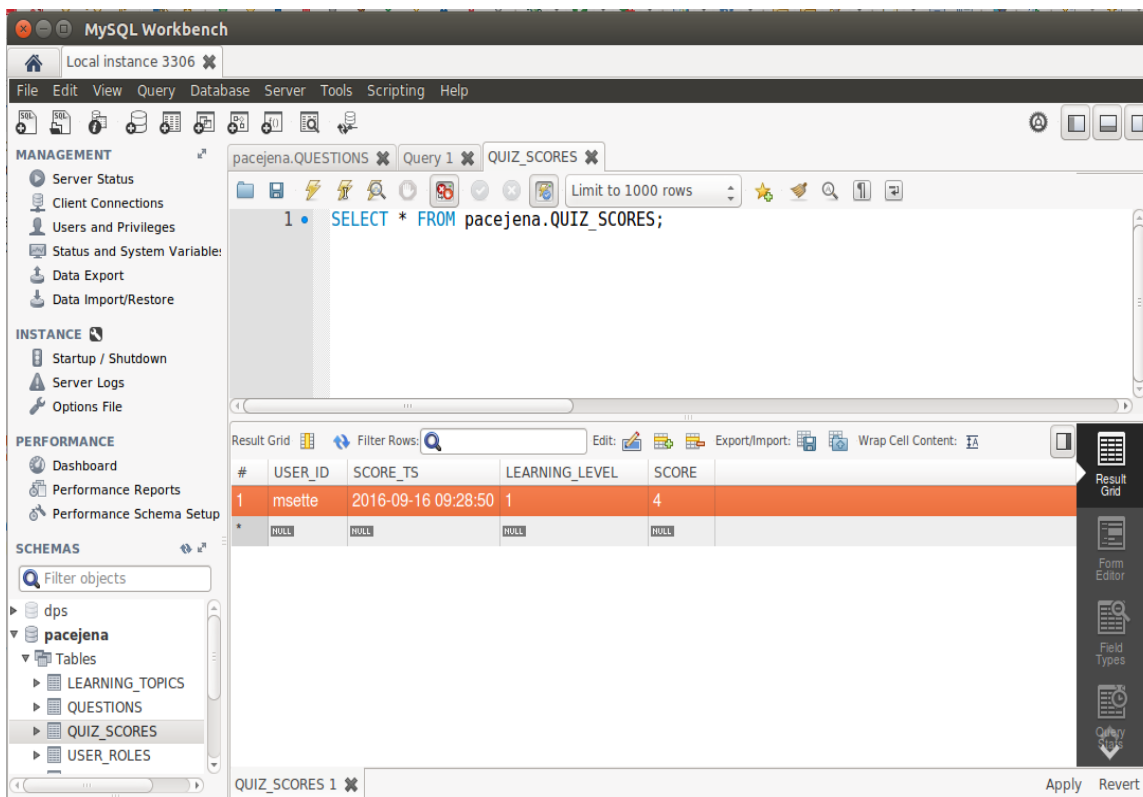


Figure 103 Quiz Scores

4.13.2 Students Knowledge Exploration

4.13.2.1 Users Access Process

In Figure 104 and Figure 105 describes the user's access process, which is first to use the authentication and authorization configuration files, handled by the Spring Security Framework. All user's data are accessed and stored in the "pacejena" database. Upon successful login, the application checks to see if the ontology is already parsed and loaded. If it isn't, it will parse the web ontology language file called 'WebTutorial.owl' and stores the resulting knowledge graph structure in memory. In Figure 104, the dispatcher knows to get this information from the PaceJenaAuthenticationSuccessHandler.java. Users are then directed by the dispatcher

servlet's security configuration file to go to the home page. The home page for each user profile will display the sequential knowledge navigational learning order for each learning level. In Figure 106, is a sample of a user flow diagram from the knowledge learning path to the knowledge quiz module and finally the quiz results, which adapts the users learning path for the next steps.

```

1 package edu.pace.pacejena.auth;
2
3 import java.io.IOException;
4
24
25 public class PaceJenaAuthenticationSuccessHandler implements AuthenticationSuccessHandler {
26     protected Log logger = LoggerFactory.getLog(this.getClass());
27
28     private RedirectStrategy redirectStrategy = new DefaultRedirectStrategy();
29
30 @Override
31 public void onAuthenticationSuccess(HttpServletRequest request,
32     HttpServletResponse response, Authentication authentication) throws IOException {
33     clearAuthenticationAttributes(request);
34     HttpSession session = request.getSession();
35     User authUser = (User) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
36     session.setAttribute("username", authUser.getUsername());
37     session.setAttribute("authorities", authentication.getAuthorities());
38
39     PaceJena paceJena = (PaceJena)session.getAttribute("PaceJena");
40     if (paceJena == null) {
41         //ServletContext sc = session.getServletContext();
42         //String fullPath = sc.getRealPath("/WEB-INF/owl/webTutorial.owl");
43         //System.out.println(fullPath);
44         paceJena = new PaceJena("/webTutorial.owl");
45         session.setAttribute(PaceJenaConstants.PACE_JENA_SESSION_KEY, paceJena);
46     }
47     loadOntologyTreeIntoSession(session, paceJena, authUser.getLearningLevel());
48
49     //set our response to OK status
50     response.setStatus(HttpServletResponse.SC_OK);
51
52     //since we have created our custom success handler, its up to us to where
53     //we will redirect the user after successfully login
54     response.sendRedirect("/pace-jena/home");
55 }
56
57
58 protected void clearAuthenticationAttributes(HttpServletRequest request) {
59     HttpSession session = request.getSession(false);
60     if (session == null) {
61         return;
62     }
63     session.removeAttribute(WebAttributes.AUTHENTICATION_EXCEPTION);
64 }
65
66 public static void loadOntologyTreeIntoSession(HttpSession session, PaceJena parser, int learningOrder) {
67     final List<String> rootClassNames = parser.getRootClassNames();
68     final String parentName = rootClassNames.get(0);
69     final OntologyNode rootNode = new OntologyNode(null, parentName);

```

Figure 104 PaceJenaAuthenticationSuccessHandler.Java

```

1 package edu.pace.pacejena.config;
2
3 import javax.sql.DataSource;
11
12 @Configuration
13 @EnableWebSecurity
14 public class SecurityConfiguration extends WebSecurityConfigurerAdapter {
15
16     @Autowired
17     DataSource dataSource;
18
19     @Autowired
20     public void configAuthentication(AuthenticationManagerBuilder auth) throws Exception {
21
22         auth.jdbcAuthentication().dataSource(dataSource)
23             .usersByUsernameQuery(
24                 "select USER_ID, PASSWORD from USERS where USER_ID=?"
25             ).authoritiesByUsernameQuery(
26                 "select USER_ID, ROLE from USER_ROLES where USER_ID=?");
27     }
28
29     @Override
30     protected void configure(HttpSecurity http) throws Exception {
31
32         http.authorizeRequests()
33             .antMatchers("/").permitAll()
34             .anyRequest().authenticated()
35             .and()
36             .formLogin().loginPage("/login").loginProcessingUrl("/j_spring_security_check").failureUrl("/login?error").defaultSuccessUrl("/home")
37             .usernameParameter("userId").passwordParameter("password").permitAll()
38             .and()
39             .logout().logoutSuccessUrl("/login?logout").permitAll()
40             .and()
41             .exceptionHandling().accessDeniedPage("/403")
42             .and()
43             .csrf();
44     }
45 }

```

Figure 105 SecurityConfiguration.java

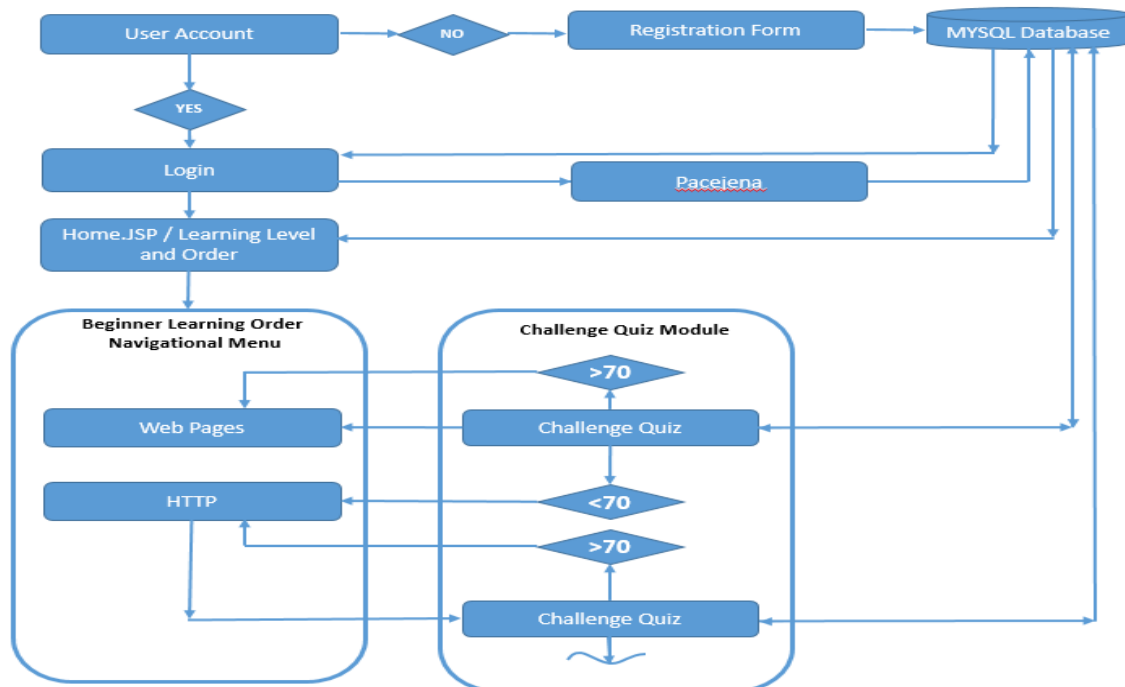


Figure 106 A Sample of User Flow Diagram

4.13.2.2 Users Registration and Login Process

1. Before a user can use the PaceJena Tutoring System he/she must have a user account.
2. If user doesn't have a user account then he/she must register for an account by clicking on the registration button.
3. The registration.jsp web page is called and presented to the user.
4. User must complete and answer all questions in the registration form in order to proceed.
5. When registration form is completed the user will submit his/her information. This information is then stored in the MYSQL database as a profile for each student.
6. The key information to be retrieved by each user will be the learning level and order, which is one the key input questions on the registration form. This is important because this is the initial starting point for each user.
7. When the user submits his her login information, the servlet responsible for handling the request is called – Home.jsp
8. The Servlet is responsible for calling the appropriate method in the DAO so that it can indirectly interact with the DB.

It is also responsible for setting and updating data saved in the bean, which will be used later by the DAO.

In this prototype,

- The LoginServlet creates a new instant of the UserBean and fills it with the username and the password entered by the user. The DAO will use this bean later to compare between the user input and the DB data.
- The Servlet calls the "login" method in the "UserDAO" to start performing its task.

9. The Login method, in the DAO, is responsible for checking whether the data entered by the user exists in the DB or not.

In addition, it has to update the Bean's data that will be used later by the servlet.

In this prototype,

- The DAO uses the ConnectionManager class to get the DB connection
- Query the DB (asks the DB to search for a user having certain username and password) and checks,
 - If the ResultSet is empty, this means that the username and password were invalid (not in the DB).
 - If the ResultSet is not empty, this means that the username and password were valid.
- Updates the UserBean.
 - In case of valid username and password, the DAO fills the bean with the rest of the user's information that will need to be displayed later by the JSP (first and last names).

In addition, it sets the "valid" attribute of the bean to true.

- Otherwise, the DAO sets the "valid" attribute of the bean to false

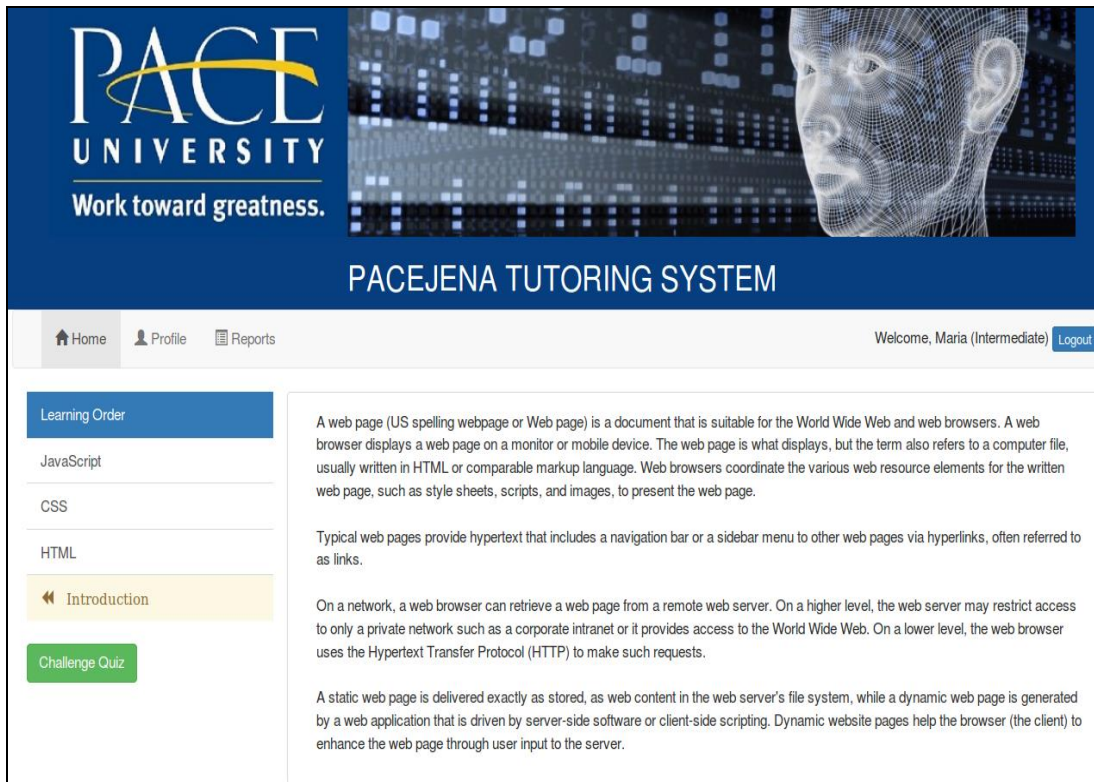
Now we know if the user was registered or not

10. Finally, the Servlet will check the validity of the user (by reading the valid attribute of the bean) and redirect to the appropriate JSP.
 - If valid, the servlet will
 - Add the bean as an attribute to the session. The bean will be used by the JSP to display the user's first and last names
 - Redirect to home.jsp - That will welcome the user to his/her personalized learning level and order home web page.

- If invalid, the servlet will redirect to invalidLogin.jsp - That will ask the user to register.

4.13.2.3 Capturing Knowledge for Generalization and Specialization

The dispatcher servlet captures the representation for both generalization and specialization from the knowledge graph. The generalization and specialization is already captured from using the OWL language but we improve the retrievals of this generalization and specialization to help improve user's personal needs when identifying parts of things and when representing new custom relations for domains that require these types of hierarchical retrieval and identification. Figure 107 illustrates how a user starts at the top level of the domain knowledge graph hierarchy, which represents topics at a broad level. By clicking on one of the topic names, a student will navigate to topics at the next level, which are more specific in the knowledge graph. Figure 108, illustrates topics which are more narrow and specific. It also contains a link to return back to the previous level, which is the "WebPage" topic. This means we can navigate from a specialized topic back to a more general topic. In this dissertation, the OWL language was extended to use the "include" and "part of" relations along with other new custom relations such as "implement" and "implementedBy"



PACE UNIVERSITY
Work toward greatness.

PACEJENA TUTORING SYSTEM

Home Profile Reports Welcome, Maria (Intermediate) Logout

Learning Order

- JavaScript
- CSS
- HTML
- Introduction
- Challenge Quiz

A web page (US spelling webpage or Web page) is a document that is suitable for the World Wide Web and web browsers. A web browser displays a web page on a monitor or mobile device. The web page is what displays, but the term also refers to a computer file, usually written in HTML or comparable markup language. Web browsers coordinate the various web resource elements for the written web page, such as style sheets, scripts, and images, to present the web page.

Typical web pages provide hypertext that includes a navigation bar or a sidebar menu to other web pages via hyperlinks, often referred to as links.

On a network, a web browser can retrieve a web page from a remote web server. On a higher level, the web server may restrict access to only a private network such as a corporate intranet or it provides access to the World Wide Web. On a lower level, the web browser uses the Hypertext Transfer Protocol (HTTP) to make such requests.

A static web page is delivered exactly as stored, as web content in the web server's file system, while a dynamic web page is generated by a web application that is driven by server-side software or client-side scripting. Dynamic website pages help the browser (the client) to enhance the web page through user input to the server.

Figure 107 Capturing Generalization in PaceJena Tutoring System



PACE UNIVERSITY
Work toward greatness.

PACEJENA TUTORING SYSTEM

Home Profile Reports Welcome, Maria (Intermediate) Logout

Learning Order

- XHTML
- HyperLinks
- HtmlForm
- WebPage
- Challenge Quiz

HyperText Markup Language, commonly abbreviated as **HTML**, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology used to create web pages,^[1] as well as to create user interfaces for mobile and web applications. Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically and, before the advent of Cascading Style Sheets (CSS), included cues for the presentation or appearance of the document (web page), making it a markup language, rather than a programming language.

HTML elements form the building blocks of HTML pages. HTML allows images and other objects to be embedded and it can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as `` and `<input />` introduce content into the page directly. Others such as `<p>...</p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages. HTML markup can also refer the browser to Cascading Style Sheets (CSS) to define the look and layout of text and other material. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.^[2]

Figure 108 Capturing Specialization in PaceJena Tutoring System

4.13.3 Instructor's Integrated Assessment

The instructor's assessment is represented in XML form and implemented in MYSQL as a quiz table to be presented to students by the web-based applications interface. The views for the quizzes are dynamically represented from a quiz table in MYSQL. The instructors quiz assessment is represented as a dynamic template containing multiple choice questions and answers that are randomized.

4.13.3.1 Integrated Assessment Work Flow Process

1. In the navigational pane the user will have the option to take a challenge quiz for each topic.
2. User will click on the challenge quiz button, which will invoke a javascript function.
3. The Javascript function will send an Ajax request to the QuizController using jquery framework.
4. The QuizController is autowired to the quizService and userService. The request method gets the ModelAndView to loadQuiz to prepare it to be scored by the quizService which will determine whether or not the student achieved a score which is greater than 70 or less than 70 to then recommend the next step.
5. If student doesn't achieve a score less than 70 then he/she will be recommended to return to the previous topic.
6. If student does achieve a score of greater than 70 then he/she will be recommended to proceed to the next topic.
7. The userService determines the user and updates, records, and stores the user information in the database.
8. When user logs into the system again the tutoring system will know the state in which the student left off last.

4.14 Process to Developing a Tutorial System for a New Subject

4.14.1 Knowledge Representation

The concept for a new subject can be developed using a course material to gather your concept course information and working with an ontology knowledge expert to provide feedback. Developing your knowledge concepts and relations can be done using the Pace Protégé tool. This tool will help support and guide the instructor in building an ontology structure for a new subject. The benefit's of using the Pace Protégé is that it can extend the OWL language and supports custom relations. Open the Protégé IDE, if you are running the revised version for the first time, please click “Window|Reset selected tab to default state” as shown in Figure 109.

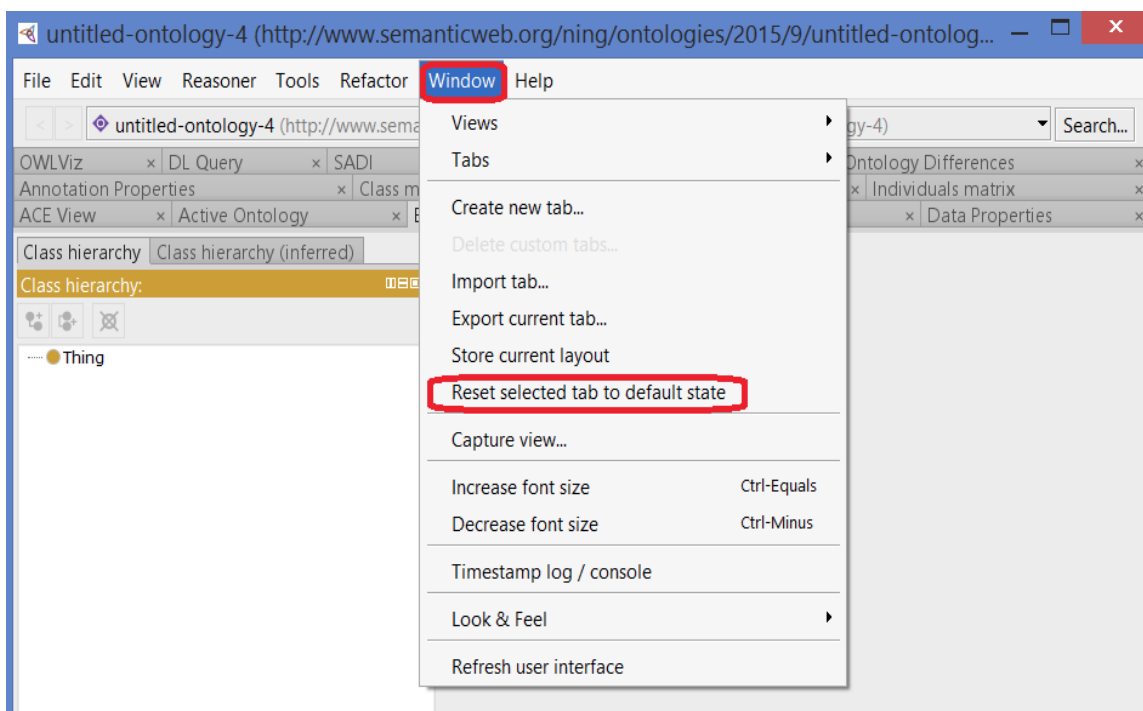


Figure 109 Resetting the Running Environment

Switch to the “Entities” tab. In the “Class hierarchy” view, select root class **Thing**, and click the “Add sub class” to create class “Finger”. While “Finger” is highlighted, click the “Add sibling class” to create classes “Hand” and “Body” as shown in Figure 110.

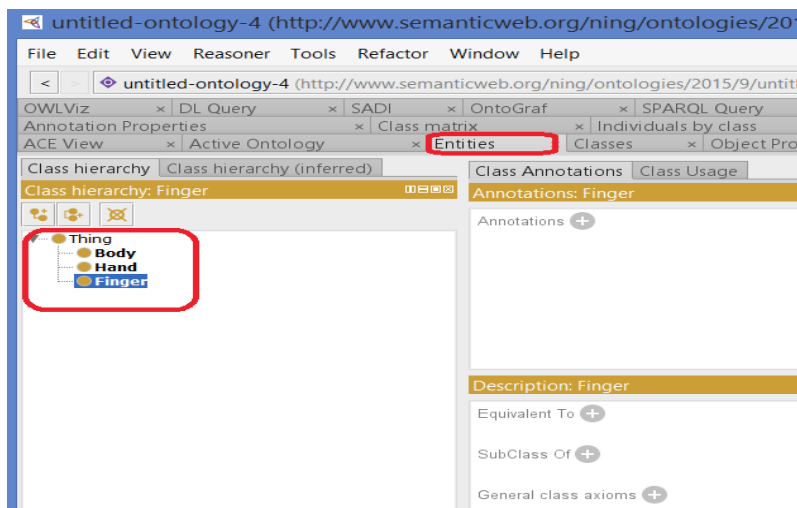


Figure 110 Defining Classes

Choose the “Relations” tab in the left bottom corner (if you don’t see it, click “Window|Reset selected tab to default state”), and choose the “add relation” to create the new relation “partOf” as shown in Figure 111. You need to type the name of the new relation: **partOf** in the Pop-up window and the click “OK” button.

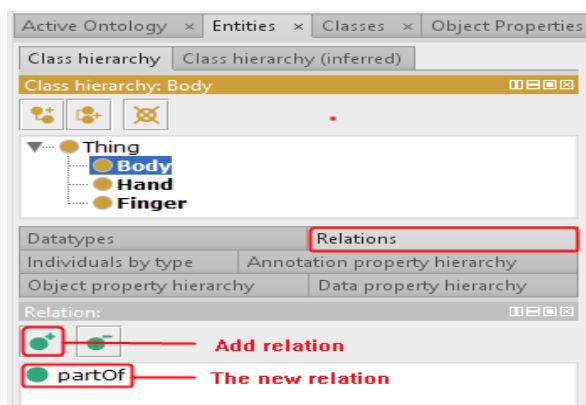


Figure 111 Create a New Relation “partOf”

Switch to “Classes” tab. You need to click “Window|Reset selected tab to default state” again, then click the “Related to” tab in the right pane. Make sure the class “Finger” is selected, and click the “Add relation” icon as shown in Figure 112.

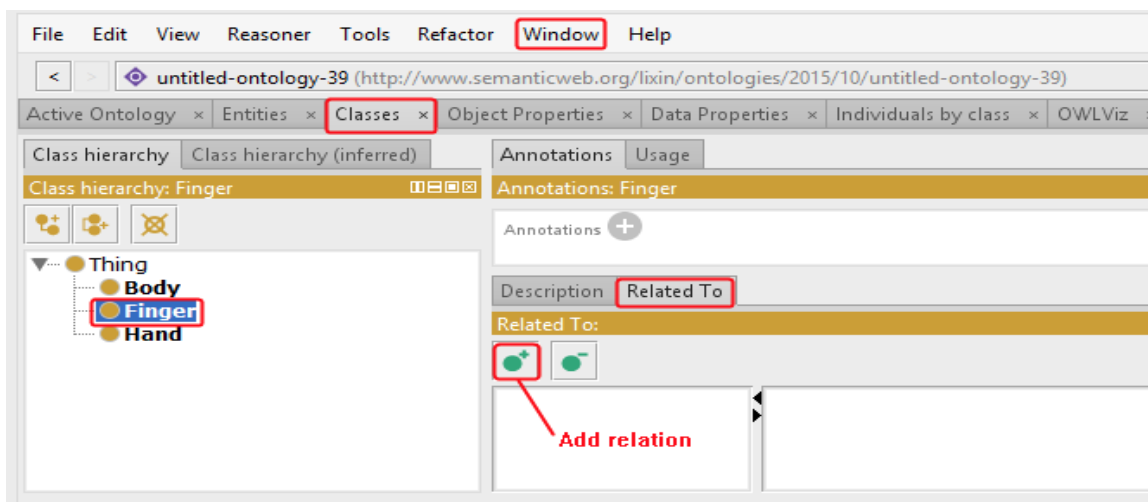


Figure 112 Defining Relations between Classes

A new window would pop up. First select “partOf” in the left pane of relations, then select “Hand” from the right class list, as shown in Figure 5. Click “OK” to close the pop-up window.

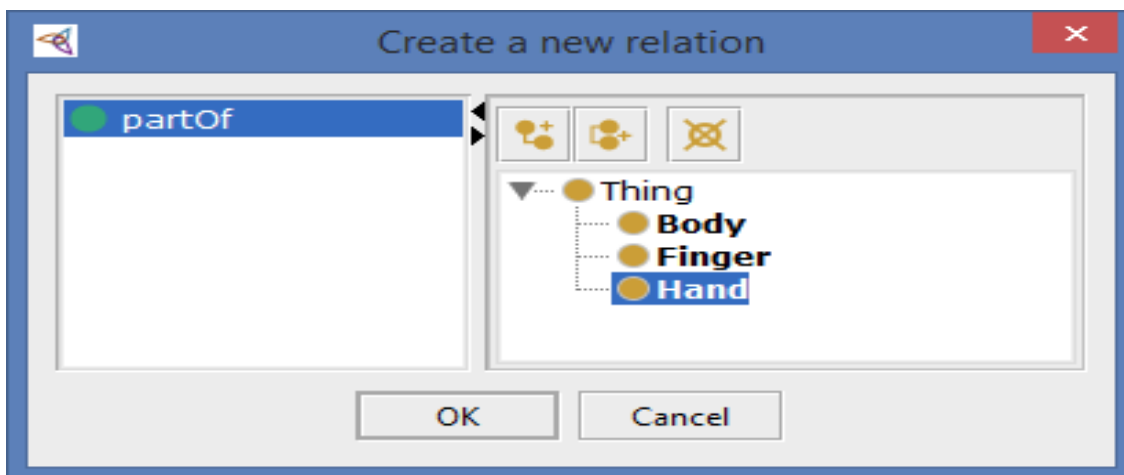


Figure 113 Declaring Relations between Concepts

Similar to steps 4 and 5, make sure the class “Hand” is selected in the class hierarchy, click “Add relation” icon () in the “Related To” tab to pop up the window for declaring a new relation, choose “partOf” in its left pane, then select “Body” in its right pane, as shown in Figure 114. Click the “OK” button to complete the declaration.

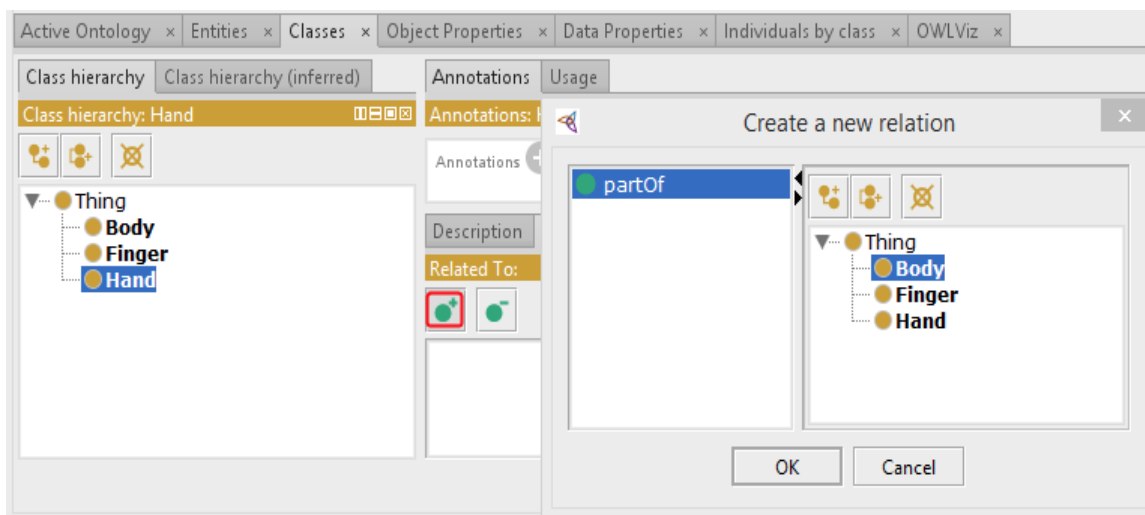


Figure 114 Declare Hand is partOf Body

Switch to “OWLviz” tab and get the visualization of your OWL file as shown in Figure 115.

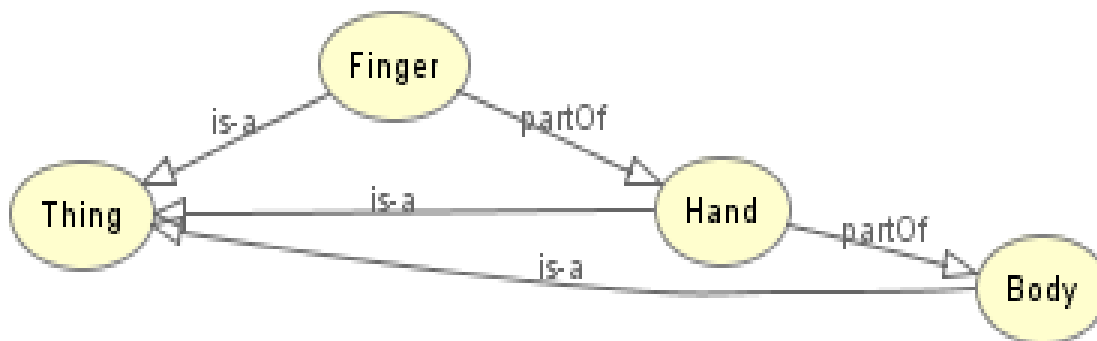


Figure 115 Visualization of the OWL

4.14.2 Learning Order Specification

The learning order specification is represented by the xml namespace called “pace” as its prefix as previously mentioned in previous sections. The relation “ref” is used to reference the topics associated with each learning order and learning level. Beginner represents a student with no experience in the topic on Web Technology. And Intermediate represents a student with some experience. When creating a new topic the instructor may choose to keep these learning levels but wishes to change the topic associated with these levels. To change the learning topics for each level the instructor can use the Pace Protégé tool to alter this information. The instructor may want to add additional learning levels such as an advance learner or expert learner. In Figure 116, the instructor can copy one of the `<owl:NamedIndividual>` start tag and copy it to its `</owl:NamedIndividual>` end tag. Next the instructor will need to assign this new learning level an ID number. Change the name from Beginner to Advance. And finally change the learning order of the topics for an advance level student. No changes are needed on the PaceJena API because it will read in and get the `ids = getLearningOrderIDs();`. In Figure 117 is a sample snapshot of the PaceJena code for learning order.

```

<!--
//
// Individuals
//
//
//
-->

<!-- http://csis.pace.edu/semweb/webTutorial.owl#Beginner -->

<owl:NamedIndividual rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Beginner">
  <rdf:type rdf:resource="&space;LearningOrder"/>
  <pace:level>0</pace:level>
  <pace:name>Beginner</pace:name>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebPage"/>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTTP"/>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#SessionManagement"/>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebArchitecture"/>
</owl:NamedIndividual>

```

Copy and Paste below to add

```

<!-- http://csis.pace.edu/semweb/webTutorial.owl#Intermediate -->

<owl:NamedIndividual rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Intermediate">
  <rdf:type rdf:resource="&space;LearningOrder"/>
  <pace:level>1</pace:level>
  <pace:name>Intermediate</pace:name>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTTP"/>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#SessionManagement"/>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebArchitecture"/>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebPage"/>
</owl:NamedIndividual>

```

Copy and Paste below to add

```

</rdf:RDF>

```

Figure 116 Defining Learning Order of Individuals

```

void printLearningOrders() {
    System.out.println("\nLearning Orders:");
    String[] ids = getLearningOrderIDs();
    for (int i = 0; i < ids.length; i++) {
        String[] names = getLearningOrderNames(ids[i]);
        System.out.print("Learning Order " + ids[i] + ":");
        for (String s : names)
            System.out.print(" " + s);
        System.out.println();
    }
    System.out.println();
}

```

Figure 117 A Sample PaceJena Code for Learning Order

4.14.3 Knowledge Object Collection and Organization

The knowledge representation of learning objects plays a critical role in the success of the tutoring system and how instructors can organize these learning materials to be suitable for adaptive interaction delivery on web-based technology and semantic web languages. In the previous section it was important to identify the learners and their learning skill levels in order to determine the organization of learning content. The next step is to organize these collections of web-based learning objects into a classification from a generalized topic to a more specialized topic for those learners whom find it challenging to grasp the learning concepts. In order to properly represent these knowledge learning objects it was also important to identify the relationships between each concept. The relations between concepts provide organizational structure that converts information known about individual entities into interconnected concepts, which are linked by different types of relations. The purpose of knowledge organization is to organize knowledge objects and to manage the knowledge collection of that information.

The domain knowledge concept for this dissertation research was identified using the reading materials from Dr. Lixin Tao's Security Lab Series called "Introduction to Web Technologies" manual. The concept collection process was to select the data starting from the main topics and to categorize these topics into subtopics in a hierarchical structure. The knowledge object resources are enriched with semantically annotated data, which are machine process able information by linking and extracting concepts from the knowledge graph. Once we have identified the new learning topic it was important to work with an ontology expert to review and evaluate the correctness of how these object collection and organization of information was structured. *Pace Protégé* was used to

develop the ontology because it provide the support of extending OWL and creating custom relations other then the “is-a” relations as mention in section 4.14.1. OWLViz tool was also extended to support the visualization of our knowledge graph. The knowledge graph was then implemented into the tutoring system file structure. In Figure 118, the WebTutorial.owl file was placed in the pace-jena /src / main / java. In Figure 119, a change must be made in the PaceJenaAuthenticationSuccessHandler.java file. Line item 44 PaceJena = new PaceJena (“/webTutorial.owl”); webTutorial.owl file will be replaced with your knowledge graph file.

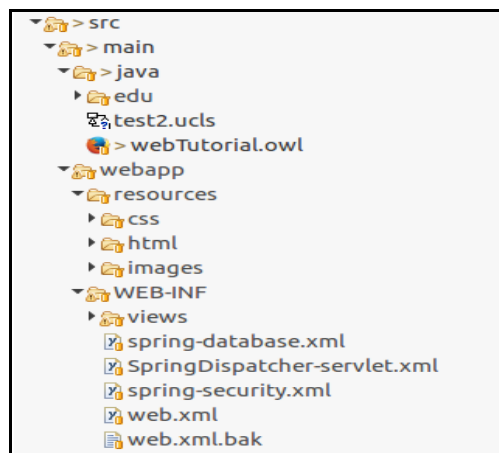


Figure 118 Implement New Ontology File

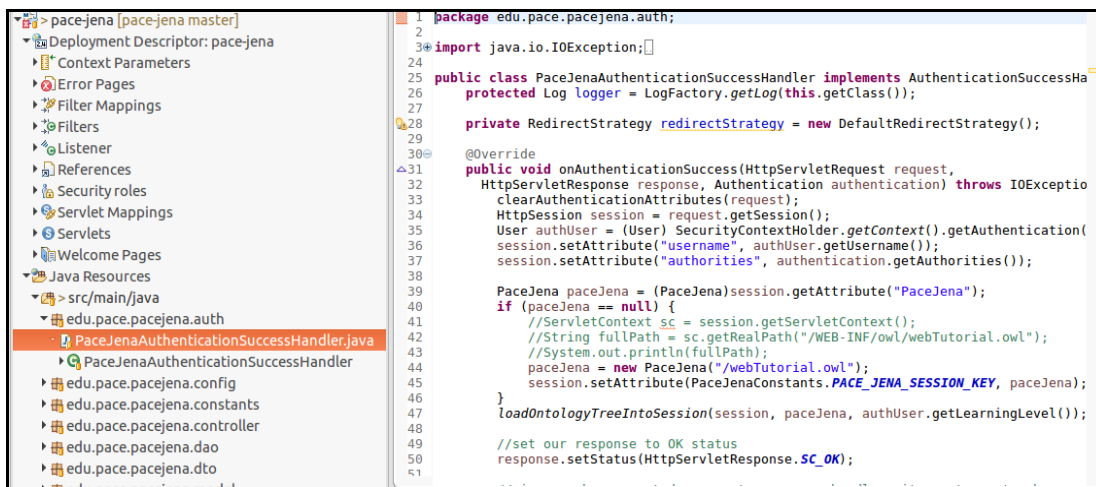


Figure 119 Replacing the webTutorial.owl in PaceJenaAuthenticationSuccessHandler.java with New Subject File

4.14.4 Managing New Topics for Editing, Saving, and Deletion

The management of new topics for editing, saving, and deleting can be easily done from the web-based application interface. As mentioned in section 4.11. The instructors and administrators must login with administrative privileges, which allows for these features to work. In Figure 120, I have logged into the system as administrator. In Figure 121 and Figure, I clicked on the WebPage topic and will enter into edit mode. Instructor must click on the edit button at the bottom of the page.

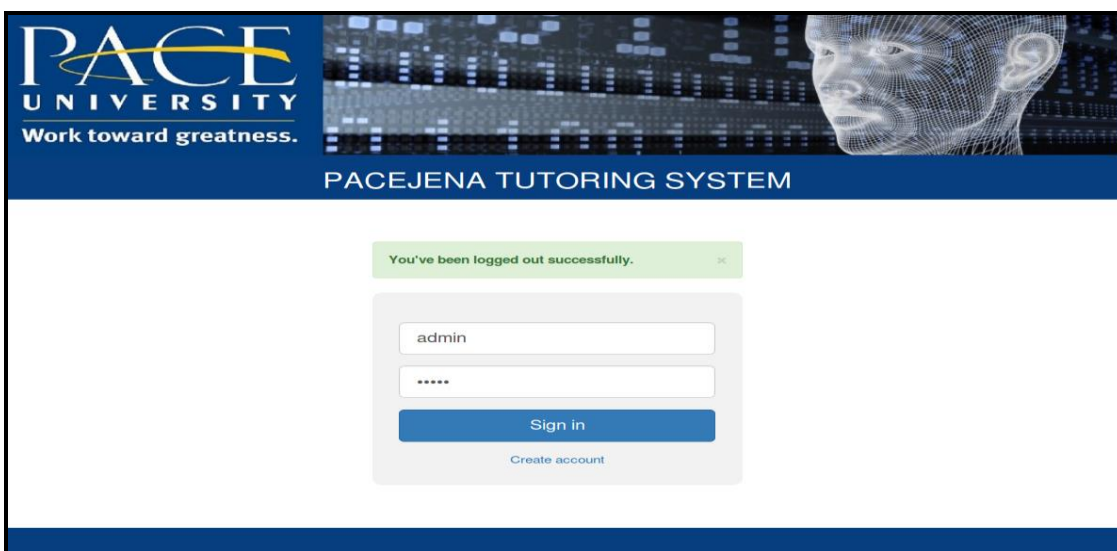


Figure 120 Administrator Login Access

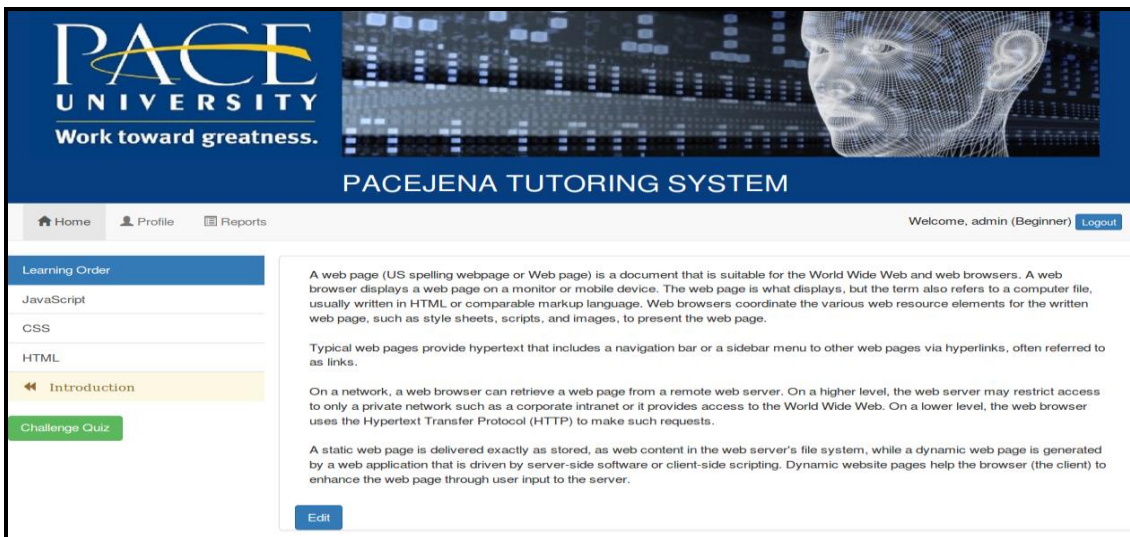


Figure 121 Click on Editing Mode

After clicking on the edit button we are ready to use the summernote editing tool which has been integrated into tutoring system which will allow the instructor to make changes to the concept content materials. The top menu allows the user to select options to make changes and save the content after the user has completed their editing function. The save button is located at the far right hand side of menu. The top menu offers other features in sequential order from left to right such as B is for BOLD letters, I for Italic, U for underline, remove font style, strikethrough, font size, unordered list, order list, paragraph, line height, full screen, code view and save. Must click on save to save your changes. Then you can click on the close button to get out of edit mode.

The screenshot displays the PACEJENA Tutoring System interface. At the top, the PACE UNIVERSITY logo is visible with the tagline "Work toward greatness." Below the logo, the text "PACEJENA TUTORING SYSTEM" is prominently displayed. The interface includes a navigation bar with links for Home, Profile, and Reports, and a user greeting "Welcome, admin (Beginner)" with a Logout button. On the left side, there is a "Learning Order" sidebar with a list of topics: JavaScript, CSS, HTML, Introduction (highlighted), and Challenge Quiz. The main content area features a rich text editor with a toolbar containing icons for Bold (B), Italic (I), Underline (U), strikethrough, font size (14), bulleted list, numbered list, text color, link, unlink, and code view. The editor contains text about web pages, including definitions and examples of static and dynamic web pages. A "Close" button is located at the bottom left of the editor area.

Figure 122 Managing Editing Mode Features

Figure 123, the course concept topics are stored in a MYSQL database table called Learning_Topics. This table contains two columns. One called Topic and the other Content. Topics contain all the learning concepts and Content contains all the content information regarding the topics for Web Technology. The system administrator would have the access to the MYSQL database.

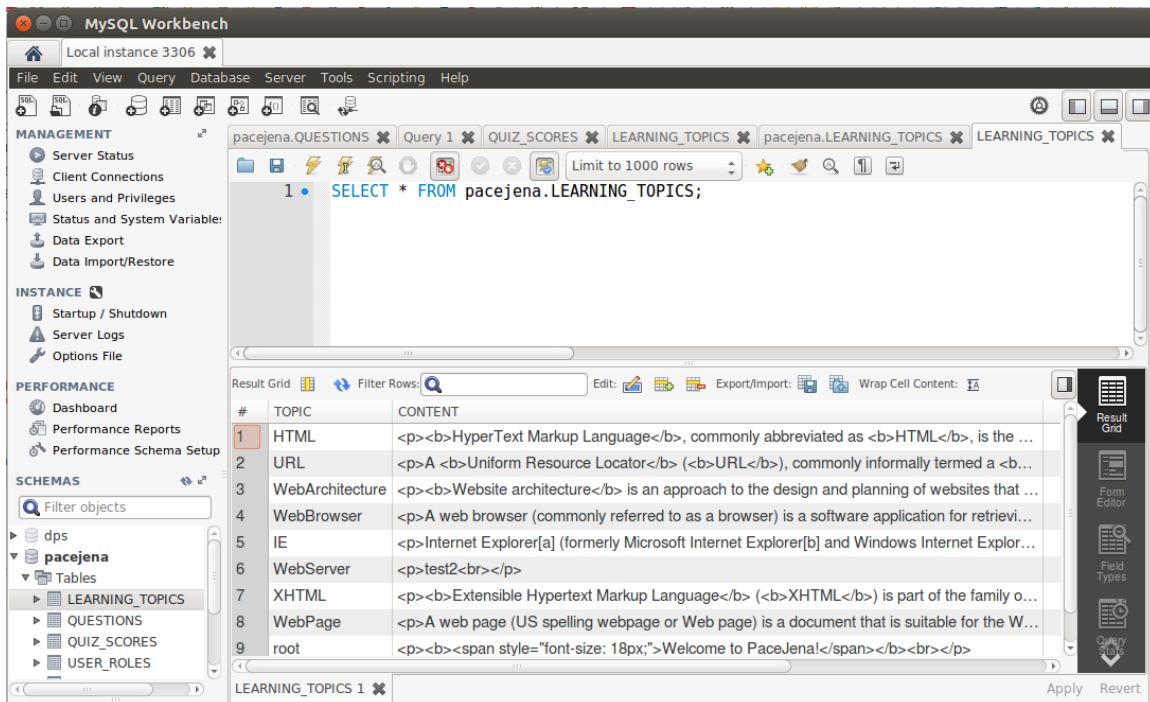


Figure 123 Learning Topic Database Table

4.14.5 Navigating the Knowledge Space

The navigation model is a knowledge graph that represents different concepts that form the navigational knowledge space. Such as a Web page requesting a data link to the knowledge graph which reads the request into PaceJena Sax Parser methods which response back to the dispatcher servlet framework and sends this request back to the user interface with the results. The Web pages, and the relations of the concepts represent

knowledge information from the domain that is displayed during a learners tutoring session. The navigational features are explained in Section 4.7 called Adaptive Tutorial Navigational Features.

4.14.6 New Quiz Development and Integration

The new quiz development and integration for a new subject topic can be easy implemented into the tutoring system. The current structure of the quiz in the MYSQL database can stay the same structure as mentioned in section 4.11.3. Quiz questions and answers can be developed as a XML file and imported into the MYSQL database to replace the existing file.

Figure 124 illustrates the command to load your XML file.

```
LOAD XML LOCAL INFILE '/pathtofile/file.xml' INTO TABLE my_tablename SET ID=NULL;
```

Figure 124 Load XML file into MYSQL Database

4.15 Summary

This section summarizes the web-based intelligent tutorial systems architecture and implementation, specification on knowledge representation, learning order, learning topics, adaptive navigational features, and process to developing a new course topic.

Chapter 5 Validating System Effectiveness through Prototype Application

This chapter discusses how the Tutoring System Prototype was evaluated to see whether it achieved its objectives. Section 5.1 explains the evaluation process in detail. It describes how the participants were selected and also discusses the procedures and instruments that were used during the evaluation process. Section 5.2 compares and contrast traditional learning with the PaceJena Tutoring System and how they were used for the purpose of analysis. Section 5.3 discusses the results of the evaluation and Section 5.4 summarizes the chapter.

5.1 Evaluation Process of the Tutorial System Prototype

The evaluation process addressed such aspects as the adaptability, usability, re-use of the system, improvement in students' knowledge representation, navigational knowledge space and assessment recommendation guidance of web-based learning objects due to use of the system, the appropriateness of the subject matter taught, the effectiveness of the teaching module and the validity of the student module.

5.1.1 Participates

The participants in the evaluation process were undergraduate students from Fairfield University School of Engineering whom agreed to evaluate the tutorial system, which is setup with the course work for “Introduction to Web Technology”. The student’s where not all from the same major concentration. The breakdown of how this is represented is as follows in the matrix below.

Major Concentration	Technology	Computer Science	Computer Engineering	Software Engineering
No. Of Students for Pre-Quiz	4	7	8	13
No. Of Students for Paper Quiz	2	3	4	7
No. Of Students for Web-Based Quiz	2	4	4	6

This course is offered at Pace University’s Computer Science Department for undergraduates. The course material was developed and taught by Dr. Lixin Tao. This course domain was choose to be used for this dissertation along with the course manual to develop the knowledge representation and knowledge graph for the tutorial system prototype. The students taking part in this evaluation had some or no prior background in Web Technology. Based on these requirements, 16 students were chosen to learn Web Technology from the traditional method and 16 students where instructed to use the tutorial system prototype. Although it would have been better to have more students, only this number showed an interest and satisfied the necessary requirements for participation.

5.1.2 Procedures and Instruments

5.1.2.1 Pre-Quiz

All participants were required to take a pre-quiz exam prior to taking traditional method exam and the web-based exam. The purpose of the pre-quiz exam was to use it in the validation results to better understand and gauge students' initial understanding of the course topic. The pre-quiz exam was given as a paper exam with the same course content material from the course topic "Introduction to Web Technology" by Dr. Lixin Tao's manual. This pre-quiz exam contains 40 multiple choice questions on all the topics that are contained in the manual. Each correct answer is worth 2.5 points. These questions cover the general topics, which are WebPage, HTTP, Session Management, and Web Architecture. The pre-quiz contains 10 questions for each topic with 4 multiple choice answers for each question. Only one correct answer is needed to complete each pre-quiz question.

5.1.2.2 Traditional Classroom Learning Method

The traditional classroom learning method is where participants are in a classroom setting with the instructor. The instructor provides the learning material resources as a manual to be read in sequential order and then at the end of each chapter the students are required to take the concept quiz for each chapter. The quizzes contain 10 multiple choice questions. Students are to select the correct answer for each question. The students are given two hours to complete traditional classroom assignments. Every student is learning the course material at the same starting point.

5.1.2.3 Web-Based Tutorial System Learning Method

The tutoring system is a cloud web-based application, which can be accessed anywhere and anytime using a browser interface and the HTTP protocol. The instructor is not available in this method, a web-based application tutoring agent is provided to support the user's interaction during each session. The participants were required to register with the tutoring system in order to obtain user credentials and access to the system. During the registration process the participants were required to select a learning level. The learning level was the initial determining fact of the knowledge space starting point for each learner. As stated in previous chapters for this prototype only two learning levels were implemented to keep it simplified. Once this was completed and submitted to the system it was then recorded and stored as a profile in the user profiles database table. Participants were required to login with their user name and password. The tutoring system would then take the successful login credentials and identified the users learning level from their stored profile and retrieve this information from the PaceJena API where it would then retrieve the knowledge graph file from the stored file structure path which would then be read and parsed by the Sax Parser. The Sax Parser would then retrieve the users learning level and match this information with the appropriate learning order. The knowledge collection and organization of the users learning material would then be displayed in a web-based browser interface for visualization and navigation into the tutoring system. The concept navigational structure would be displayed in the left pane window and the concept content would be displayed in the body area. Each concept data would have its own button form. The challenge quiz would be displayed below the concept navigational structure in a button form. At any point and time during the learning

process the user has the option of when he / she is ready to take the challenge quiz for a practical topic. The challenge quiz would then display the selected concept quiz. Each quiz is setup to display 10 multiple choice questions. Participants are to select one answer for each question and do not have the ability to go back to the previous question. The next button is display after each selected answer in order to move forward in the quiz process. The last question would only display a submit button. The submit button would then evaluate the quiz questions to produce a scoring of either greater than 70 percent or less than 70 percent. The tutoring system would then recommend and determine the next learning state for the user. The scoring is stored in the user's database profile. This process ends when the user has completed all concept quizzes successfully.

5.1.3 Comparing and Contrasting Different Learning Methods

5.1.3.1 Traditional Classroom Learning Method

This method does not provide the students with the ability to start the learning process according to their knowledge level. Students read the material then take the quiz. The instructor grades quizzes later after the students complete the entire assignment. This method lacked in providing students with more information if they lacked the understanding of the concept. This also method does not provide students with guidance of real-time results of each learning concept. Students are left with not knowing whether or not they understood the concept from each chapter due to the lack of real-time quiz feedback which can lead to discouragement and strangling with understand the other course concepts. Real-time feedback is important for students so that it can guide and support the learning process of each student.

5.1.3.2 Web-Based Tutorial System Learning Method

The benefits of this web-based tutoring system provides the student the ability to select his / her learning level during the registration process. This is different than the traditional learning method because student have different learning starting points but are learning the same materials. This method also supports and provides the student with an adaptive navigational feature, which will guide the student during the learning process. These adaptive navigational features as mention in section 4.8 provides the student the support needed to navigate to more specific concept from the general concept. This type of feature is useful for those students whom may need more information on the general concept. The more specific concepts are broken down into smaller concepts that are specific to the general concept. This provides the student with more detail information in order to grasp the concept as a whole. The quiz module provides the student with real-time feedback after taking the quiz. The tutoring system will recommend whether the student can proceed to the next concept or whether the student needs to go back and revisit the previous concept due to the lack of understanding which is evaluate from the quiz results. This type of feedback is not preferred during the traditional method.

5.1.4 Students Feedback

5.1.4.1 Traditional Classroom Learning Method

Students feedback on the traditional learning method were not satisfied with the learning material due to the lack of information needed if you were a beginner in this area learning this course concept. Other's whom where more knowledgeable with the course concepts where not satisfied with the learning process of having to learning the subject matter from

the beginning of the learning lesson. Students all agreed that the quiz process lacked the necessary feedback for evaluating the student's performance after completing each assignment section along with the quiz and also lacked in the support needed to guide students in the learning process.

5.1.4.2 Web-Based Tutorial System Learning Method

Students experience with the web-based tutoring system prototype, find the process to be very user friendly. Both students with beginner and intermediate experience find the registration form of selecting the initial learning level to be very helpful and how the system was able to retrieve their learning order and display this information in a web-based browser application. Students also felt that the adaptive navigational features provide much needed support in understanding the course concepts and the abilities to freely navigate to different learning concepts. As stated in previous chapter's students with less knowledge benefited from being able to navigate to more specific details to better understand the general topic. The real-time adaptive quiz module provides the necessary feedback needed to evaluate and recommend the students next learning steps. Student's also commented on the prototype limitation, which doesn't provide different types of learning object resources. These would include PowerPoint presentations, videos, audios, and much more.

5.2 Analyzing Quiz Results

This section will discuss the validation of the results of all quizzes taken by all students. Table 14, 15, & 16 Quiz Results is a description of the number of students that have participated in this study and the three different types of quizzes used to validate this dissertation research with a comparative study for each group. Table 14 describes the 16 students whom took the pre-quiz and paper quiz. The numbers under each quiz are the score results and the overall total average is calculated at the bottom of this table.

Table 14 Analyzing and Comparing Pre-Quiz and Paper Quiz Results

No. Of Students	Pre-Quiz	Paper Quiz
1	42.5	50
2	52.5	60
3	52.5	60
4	32.5	50
5	65	70
6	35	40
7	65	70
8	55	60
9	55	60
10	72.5	80
11	55	60
12	60	60
13	55	60
14	55	60
15	75	70
16	62.5	70
Average	55.625	61.25

Table 15 describes the 16 students whom took the pre-quiz and web-based quiz. The numbers under each quiz are the score results and the overall total average is calculated at the bottom of this table.

Table 15 Analyzing and Comparing Pre-Quiz and Web-Based Quiz Results

No. Of Students	Pre-Quiz	Web-Base Quiz
1	50.5	60
2	55.5	70
3	62.5	80
4	52.5	60
5	70	80
6	32.5	50
7	55	60
8	62.5	70
9	60	70
10	72.5	90
11	65	70
12	62.5	80
13	65	70
14	60	70
15	65.5	90
16	65	80
Average	59.78125	71.875

Table 16 analyzes and compares both students who took the paper quiz and the web-based quiz. The numbers under each quiz are the score results and the overall total average is calculated at the bottom of this table.

Table 16 Analyzing and Comparing Paper Quiz and Web-Based Quiz Results

No. Of Students	Paper Quiz	Web-Base Quiz
1	50	60
2	60	70
3	60	80
4	50	60
5	70	80
6	40	50
7	70	60
8	60	70
9	60	70
10	80	90
11	60	70
12	60	80
13	60	70
14	60	70
15	70	90
16	70	80
Average	61.25	71.875

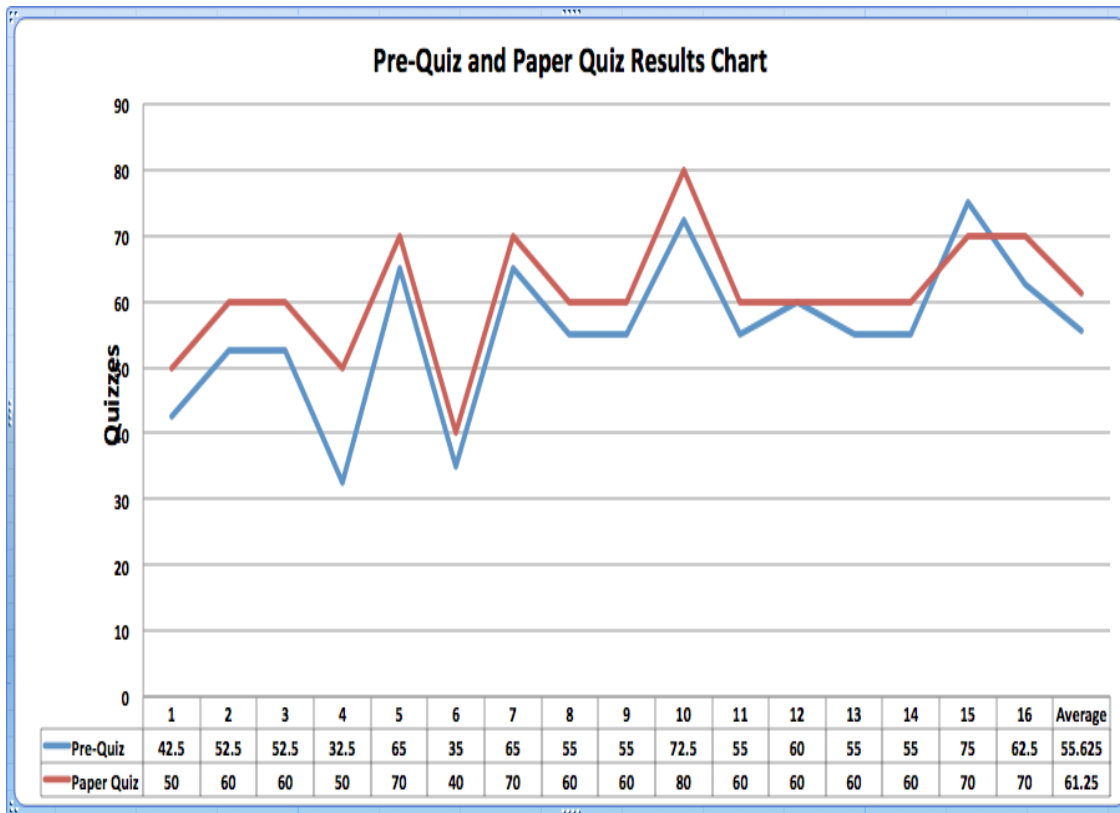


Figure 125 Analyzing and Comparing Pre-Quiz and Paper Quiz Results Chart

This is analyzing and comparing Pre-Quiz with the Paper Quiz.

Calculating Pre-Quiz and Paper Quiz Results

Calculating the difference between Pre-Quiz and Paper Quiz.

$$\text{Formula} = (\text{Paper Quiz} - \text{Pre-Quiz} / \text{Pre-Quiz}) \times 100$$

$$\text{Quiz Average} = (61.25 - 55.625 / 55.625) \times 100 = 10.11 \text{ or } 10\%$$

This means that the Paper Quiz is 10% more effective than the Pre-Quiz.

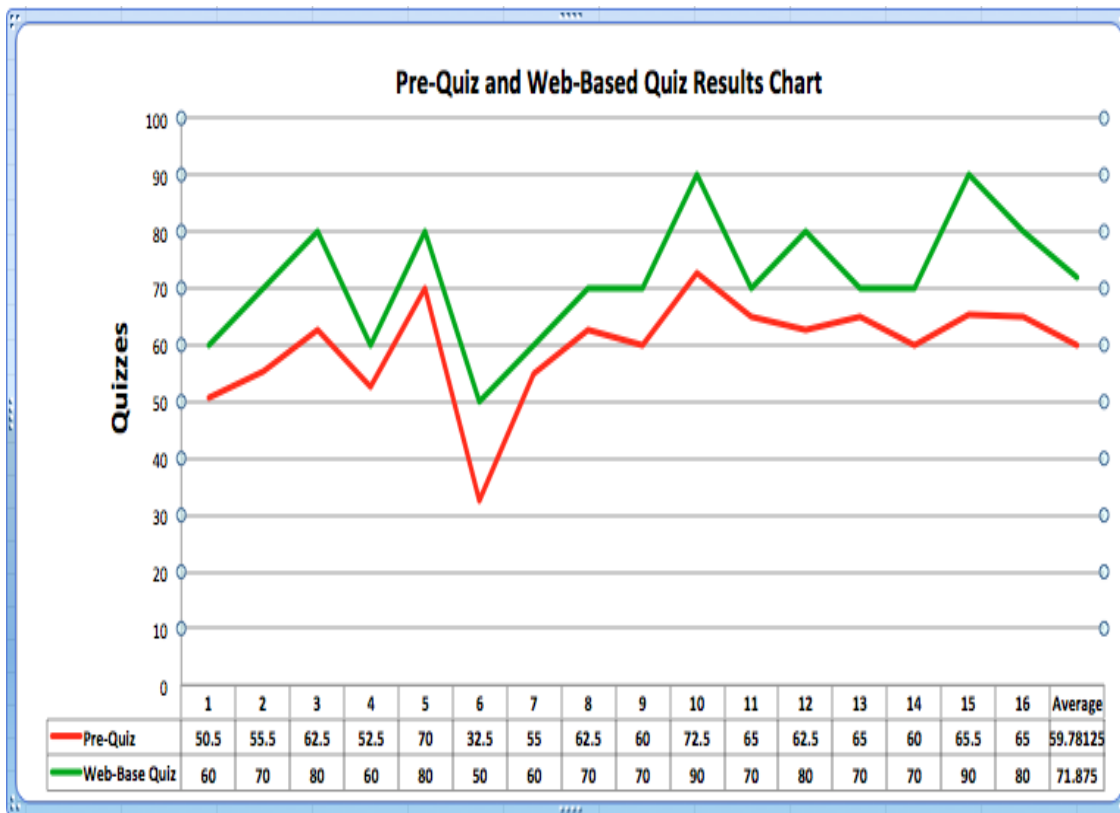


Figure 126 Analyzing and Comparing Pre-Quiz and Web-Based Quiz Results Chart

This is analyzing and comparing Pre-Quiz with the Web-Based Quiz.

Calculating Pre-Quiz and Web-Based Quiz Results

Calculating the difference between Pre-Quiz and Web-Based Quiz.

$$\text{Formula} = (\text{Web-Based Quiz} - \text{Pre-Quiz} / \text{Pre-Quiz}) \times 100$$

$$\text{Quiz Average} = (71.875 - 59.78 / 59.78) \times 100 = 20.23 \text{ or } 20\%$$

This means that the Web-Based Quiz is 20% more effective than the Pre-Quiz.

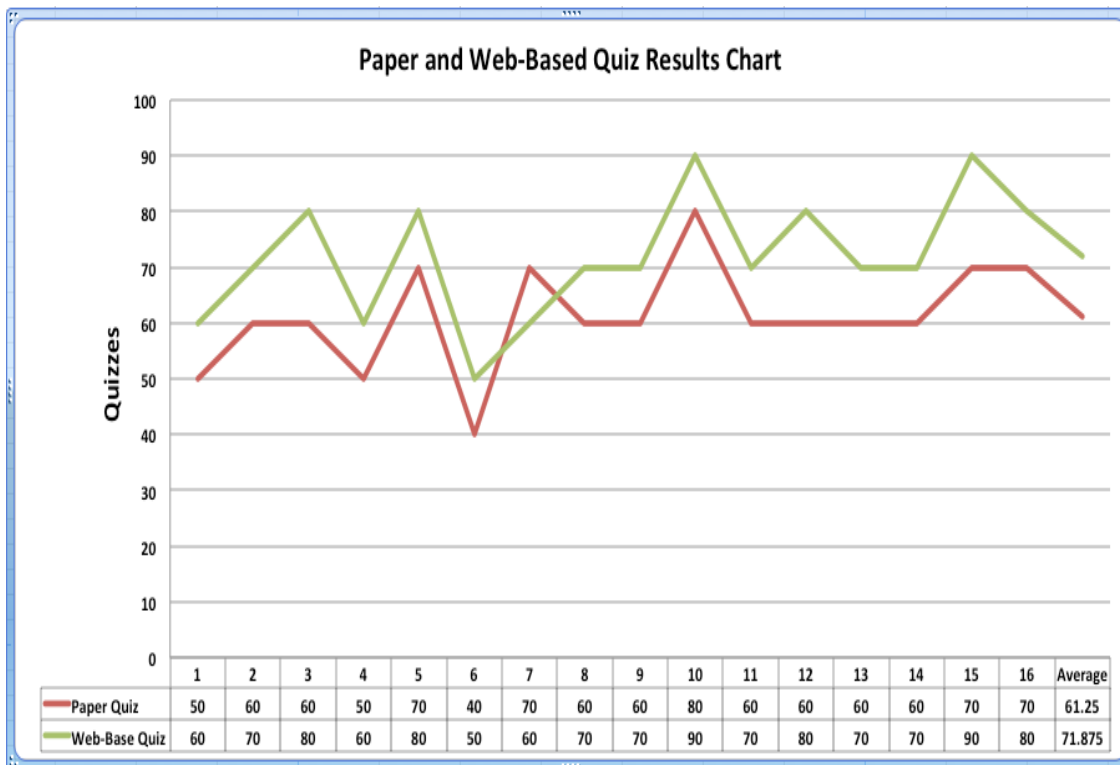


Figure 127 Analyzing and Comparing Paper and Web Based Quiz Results Chart

This is analyzing and comparing Paper Quiz with the Web-Based Quiz.

Calculating Paper Quiz and Paper Quiz Results

Calculating the difference between Pre-Quiz and Paper Quiz.

$$\text{Formula} = (\text{Web-Based Quiz} - \text{Paper-Quiz} / \text{Paper-Quiz}) \times 100$$

$$\text{Quiz Average} = (71.875 - 61.25 / 61.25) \times 100 = 14.78 \text{ or } 15\%$$

This means that the Web-Based Quiz is 15% more effective then the Paper Quiz.

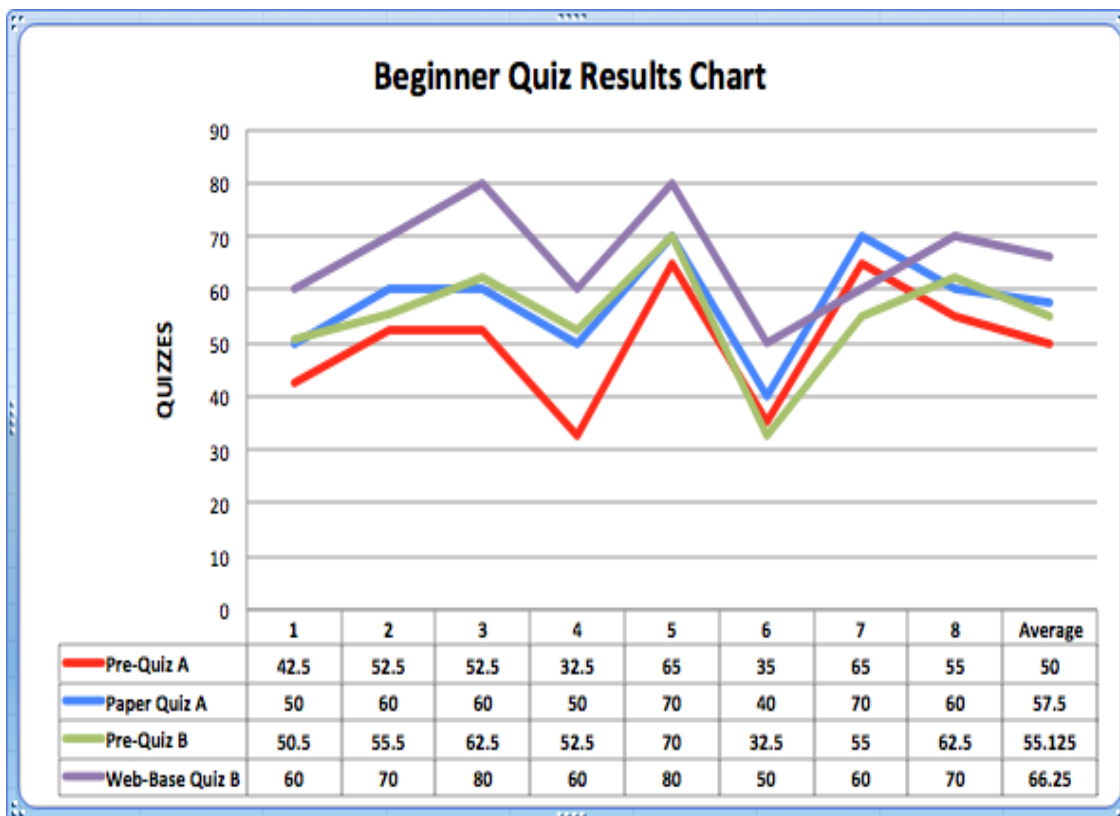


Figure 128 Analyzing and Comparing Beginner Quiz Results Chart

This is analyzing and comparing Beginner students whom took the paper quiz with Beginner students whom took the web-based quiz.

Calculating Beginner Quiz Results

Calculating the difference between Paper Quiz and the Web Based Quiz for Beginner students.

$$\text{Formula} = (\text{Web-Based Quiz B} - \text{Paper Quiz A} / \text{Paper Quiz A}) \times 100$$

$$\text{Beginner Quiz Average} = (66.25 - 57.5 / 57.5) \times 100 = 15.21 \text{ or } 15\%$$

This means that the Web-Based Quiz for a Beginner is 15% more effective than the Paper Quiz.

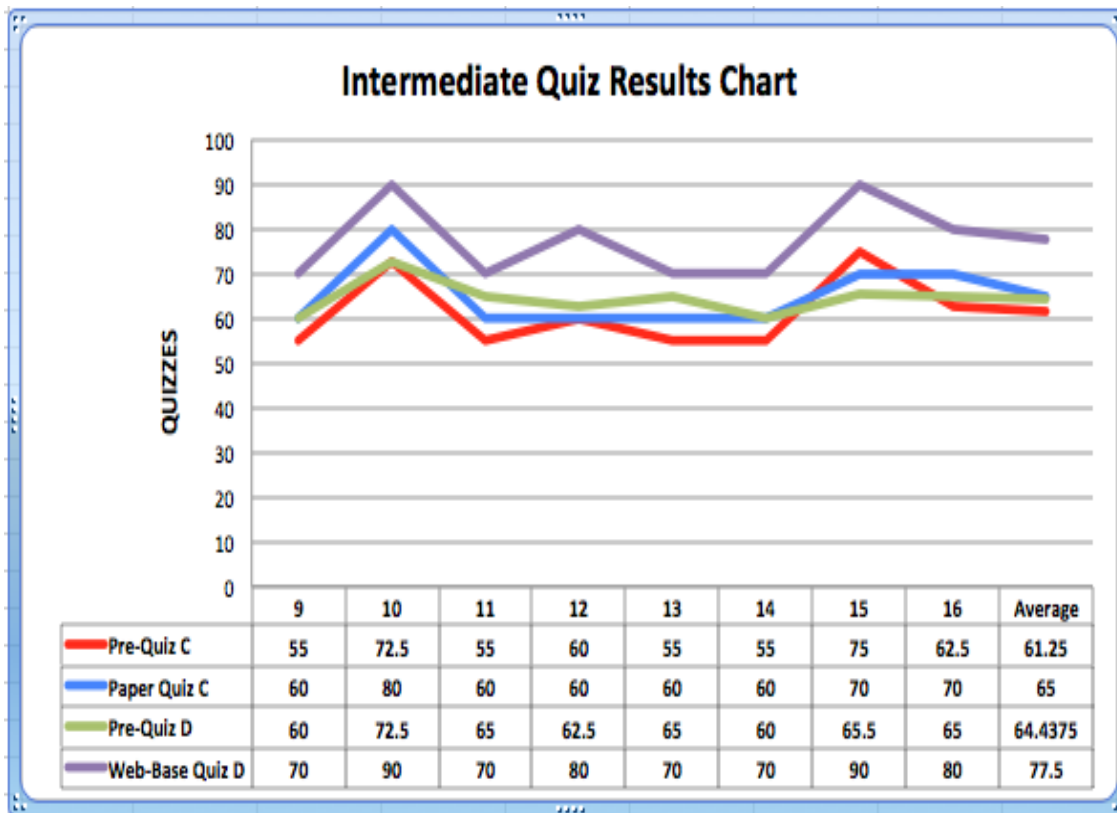


Figure 129 Analyzing and Comparing Intermediate Quiz Result Chart

This is analyzing and comparing Intermediate students whom took the paper quiz with Intermediate students whom took the web-based quiz.

Calculating Intermediate Results

Calculating the difference between Paper Quiz and the Web Based Quiz for Beginner students.

$$\text{Formula} = (\text{Web-Based Quiz D} - \text{Paper Quiz C} / \text{Paper Quiz C}) \times 100$$

$$\text{Intermediate Quiz Average} = (77.5 - 65 / 65) \times 100 = 19.23 \text{ or } 19\%$$

This means that the Web-Based Quiz is 19% more effective than the Paper Quiz.

5.3 Conclusion

We can conclude that our prototype effectively validates both the pre-defined learning path and adaptive learning. We can also qualify that by extending the OWL language, using custom relations and knowledge graph to describe our knowledge structure, knowledge space and open source technology infrastructure does indeed provide the reusability, flexibility and adaptability that our tutoring system requires. The tutoring system quiz results reveal that student's performance improved in real-time with a recommender, which is a knowledge driven graph, helped support students learning process.

Chapter 6 Conclusion

In concluding this research, touches on many areas such as Cyberlearning, learning objects, semantic web technology, open source technology, open source tools and our very own customized semantic tools to demonstrate how it was possible to developing the knowledge structure and knowledge graph for classification, customized relations, graph visualization using these customized relations, the ability for learning object navigation of our knowledge structure and knowledge evaluation to determine student learning level and the order in which learning materials are presented.

The development of PaceJena Tutoring System was developed to demonstrate student personalization and how all these technologies and tools where used to demonstrate student's ability to easily navigate the learning objects and crawl the knowledge graph.

Finally the implementation for PaceJena Tutoring System and its customized supporting tools and other solutions such as the semantic web technologies and open source technologies can be used with other technologies like databases, enterprise systems, JSP, Jena, Apache, and much more. We can conclude by the design and implementation that semantic web solutions provide a great flexibility to adopt new requirements about data classification and there relations without change the technology architecture structure. Semantic web languages and its technologies are very powerful concept for knowledge

graph representation and how this knowledge data is retrieved using our customized PaceJena program. This work also describes our customized Pace Protégé tool in developing our customized relations and customized graph, using OWLViz. In order to retrieve this knowledge data, our knowledge structure was expressed using the basic is-a relations, the partOf relations and we had to extend and develop some new customized relations such as the include, implemented, and implementedBy relations. These concepts and relations gave meaning to our data and the use of ontology made it possible for our data to be machine processed. It was very difficult to find help with the technology that was used for this research.

6.1 Major Research Contributions

This dissertation makes contributions in the area of knowledge structure and representation in adaptive and web-based tutoring system application framework to improve and recommend students learning path and learning order. A prototype tutoring system called PaceJena Tutoring System has been developed to provide students the ability to personalize their learning path according to their learning abilities which helped to enhance the ease of adaptive personalized learning features along with adaptive navigational features which guides students by the knowledge graph structure to support students learning. Another contribution the prototype provides students with feedback to real-time quiz results and guidance. And finally the instructors editing tool, which was integrated into the learning content to support different types of editing features for

learning concepts. This research has been published and is available to other student and researchers whom are interested in extending this work.

6.2 Potential Future Work

Future research in this area of study can be extended for many opportunities in the area of analyzing personalized learning with different type of semantic web languages and technology, analyzing different types of teaching strategies, analyzing different type of performance strategies, analyzing different types of semantic architecture frameworks, technologies, tools and different types of inferences, logic and application programming interfaces. Also extend Knowledge Graphs to support and improve effective design and development for Internet of Things (IoTs) and Cloud Base systems.


```

////////////////////////////////////
-->

<!-- http://csis.pace.edu/semweb#LearningOrder -->
  <owl:Class rdf:about="&pace;LearningOrder">
</owl:Class>

  <!-- http://csis.pace.edu/semweb#ref -->
  <owl:Class rdf:about="&pace;ref">
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#Apache -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Apache">
    <rdfs:subClassOf
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebServer"/>
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#ApplicationServer -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#ApplicationServer">
    <rel:implementedBy
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#EjbContainer"/>
    <rel:implementedBy
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#MsTransactionServer"/>
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#CSS -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#CSS">
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#Cookie -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Cookie">
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#DatabaseServer -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#DatabaseServer">
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#DoGet -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#DoGet">
    <rel:partOf rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTTP"/>
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#DoPost -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#DoPost">
    <rel:partOf rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTTP"/>
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#DomainName -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#DomainName">

```

```

</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#DomainNameServer -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#DomainNameServer">
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#EjbContainer -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#EjbContainer">
    <rel:implement
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#EntityEJB"/>
    <rel:implement
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#SessionEJB"/>
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#EntityEJB -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#EntityEJB">
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#Firefox -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Firefox">
    <rdfs:subClassOf
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebBrowser"/>
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#GlassFish -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#GlassFish">
    <rdfs:subClassOf
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#ApplicationServer"/>
    <pace:name>GlassFish Server</pace:name>
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#HTML -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#HTML">
    <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HyperLinks"/>
    <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HtmlForm"/>
    <rel:partOf rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebPage"/>
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#WebPage -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#WebPage">
    <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#JavaScript"/>
    <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#CSS"/>
    <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTML"/>
</owl:Class>

  <!-- http://csis.pace.edu/semweb/webTutorial.owl#HTTP -->
  <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#HTTP">
    <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#DoGet"/>
    <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#DomainNameServer"/>
    <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#DoPost"/>
    <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#SessionManagement"/>

```

```

    <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#URL"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#HashTable -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#HashTable">
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#HiddenField -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#HiddenField">
  <rel:partOf rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HtmlForm"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#HtmlForm -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#HtmlForm">
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#HyperLinks -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#HyperLinks">
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#IE -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#IE">
  <rdfs:subClassOf
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebBrowser"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#IPAddress -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#IPAddress">
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#IPv4 -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#IPv4">
  <rdfs:subClassOf
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#IPAddress"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#IPv6 -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#IPv6">
  <rdfs:subClassOf
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#IPAddress"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#JSF -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#JSF">
</owl:Class>

```

```
<!-- http://csis.pace.edu/semweb/webTutorial.owl#JSP -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#JSP">
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#JavaScript -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#JavaScript">
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#MsTransactionServer -->
<owl:Class
rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#MsTransactionServer">
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#MySQL -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#MySQL">
  <rdfs:subClassOf
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#DatabaseServer"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#Oracle -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Oracle">
  <rdfs:subClassOf
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#DatabaseServer"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#PHP -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#PHP">
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#Plugins -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Plugins">
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#PortNumber -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#PortNumber">
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#QueryString -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#QueryString">
  <rel:partOf rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#URL"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#SQL -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#SQL">
  <rdfs:subClassOf
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#DatabaseServer"/>
</owl:Class>
```

```

<!-- http://csis.pace.edu/semweb/webTutorial.owl#SecuritySandbox -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#SecuritySandbox">
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#Servlet -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Servlet">
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#ServletContainer -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#ServletContainer">
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#JSP"/>
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#JSF"/>
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#Servlet"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#SessionEJB -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#SessionEJB">
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#SessionID -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#SessionID">
  <rel:implementedBy
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#Cookie"/>
  <rel:implementedBy
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HiddenField"/>
  <rel:implementedBy
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#QueryString"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#SessionManagement -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#SessionManagement">
  <rel:implementedBy
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#Cookie"/>
  <rel:implementedBy
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HiddenField"/>
  <rel:implementedBy
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#QueryString"/>
  <rel:implementedBy
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#SessionObject"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#SessionObject -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#SessionObject">
  <rel:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HashTable"/>
  <rel:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#SessionID"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#Tier1 -->
<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Tier1">
  <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebBrowser"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#Tier2 -->

```

```

    <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Tier2">
      <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebServer"/>
    </owl:Class>

    <!-- http://csis.pace.edu/semweb/webTutorial.owl#Tier3 -->

    <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Tier3">
      <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#ApplicationServer"/>
    </owl:Class>

    <!-- http://csis.pace.edu/semweb/webTutorial.owl#Tier4 -->

    <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Tier4">
      <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#DatabaseServer"/>
    </owl:Class>

    <!-- http://csis.pace.edu/semweb/webTutorial.owl#Tomcat -->

    <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Tomcat">
      <rdfs:subClassOf
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebServer"/>
    </owl:Class>

    <!-- http://csis.pace.edu/semweb/webTutorial.owl#URL -->

    <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#URL">
      <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#DomainName"/>
      <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#IPAddress"/>
      <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#PortNumber"/>
      <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#QueryString"/>
    </owl:Class>

    <!-- http://csis.pace.edu/semweb/webTutorial.owl#WebArchitecture -->

    <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#WebArchitecture">
      <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#Tier4"/>
      <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#Tier3"/>
      <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#Tier2"/>
      <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#Tier1"/>
    </owl:Class>

    <!-- http://csis.pace.edu/semweb/webTutorial.owl#WebBrowser -->

    <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#WebBrowser">
      <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#Plugins"/>
      <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#SecuritySandbox"/>
    </owl:Class>

    <!-- http://csis.pace.edu/semweb/webTutorial.owl#WebLogic -->

    <owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#WebLogic">
      <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#ApplicationServer"/>
      <rel:include

```



```

rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebServer"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#WebPage -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#WebPage">
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#CSS"/>
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTML"/>
  <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#JavaScript"/>
  <rel:partOf
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebTechnology"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#WebServer -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#WebServer">
  <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebServerCore"/>
  <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebServerExtension"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#WebServerCore -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#WebServerCore">
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#WebServerExtension -->

<owl:Class
rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#WebServerExtension">
  <rel:implementedBy
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#PHP"/>
  <rel:implementedBy
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#ServletContainer"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#WebSphere -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#WebSphere">
  <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#ApplicationServer"/>
  <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebServer"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#WebTechnology -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#WebTechnology">
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTTP"/>
  <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#SessionManagement"/>
  <rel:include
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebArchitecture"/>
  <rel:include rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebPage"/>
</owl:Class>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#XHTML -->

<owl:Class rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#XHTML">
  <rdfs:subClassOf

```

```

rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTML"/>
</owl:Class>

<!--

////////////////////////////////////
//
// Individuals
//
////////////////////////////////////
-->

<!-- http://csis.pace.edu/semweb/webTutorial.owl#Beginner -->

<owl:NamedIndividual
rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Beginner">
  <rdf:type rdf:resource="&pace;LearningOrder"/>
  <pace:level>0</pace:level>
  <pace:name>Beginner</pace:name>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebPage"/>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTTP"/>
  <pace:ref
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#SessionManagement"/>
  <pace:ref
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebArchitecture"/>
  </owl:NamedIndividual>

<!-- http://csis.pace.edu/semweb/webTutorial.owl#Intermediate -->

<owl:NamedIndividual
rdf:about="http://csis.pace.edu/semweb/webTutorial.owl#Intermediate">
  <rdf:type rdf:resource="&pace;LearningOrder"/>
  <pace:level>1</pace:level>
  <pace:name>Intermediate</pace:name>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#HTTP"/>
  <pace:ref
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#SessionManagement"/>
  <pace:ref
rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebArchitecture"/>
  <pace:ref rdf:resource="http://csis.pace.edu/semweb/webTutorial.owl#WebPage"/>
  </owl:NamedIndividual>

</rdf:RDF>

<!-- Generated by the OWL API (version 3.5.1) http://owlapi.sourceforge.net -->

```

Appendix B

PaceJena API

```

package edu.pace.semweb;
import java.io.*;
import java.util.*;
import org.xml.sax.*;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.ext.LexicalHandler;

import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;

public class PaceJena extends DefaultHandler implements LexicalHandler {
    StringBuffer textBuffer;
    Stack<OwlElement> stack_E = new Stack<OwlElement>(); // stack of element enum
values
    Stack<Object> stack_object = new Stack<Object>(); // stack of OWL elements
    int pass = 1;

    Hashtable<String, Ontology> ontologyHash = new Hashtable<String, Ontology>();
    Ontology currentOntology;

    public static void main(String argv[]) {
        if (argv.length < 1 || argv.length > 2) {
            System.err.println("Usage: java PaceJena Ontology-file");
            System.exit(1);
        }
        PaceJena o = new PaceJena();
        o.readOntology(argv[0]);
        o.printOntology();
        o.printClasses();
        o.printProperties();
        o.printDataTypes();
        o.printLearningOrders();

        System.exit(0);
    }

    public PaceJena() {}

    public PaceJena(String ontologyFileName) {
        PaceJena o = new PaceJena();
        readOntology(ontologyFileName);
    }

    String removeFilePath(String fileName) {
        int i1 = fileName.lastIndexOf('\\');
        int i2 = fileName.lastIndexOf('/');
        if (i1 > i2) i2 = i1;
        if (i2 == -1) return fileName;
        else return fileName.substring(i2+1);
    }

    public void readOntology(String ontologyFileName) {
        currentOntology = new Ontology();
        currentOntology.filePath = ontologyFileName;
    }

```

```

String ontologyNameBase = removeFilePath(ontologyFileName);
currentOntology.fileName = ontologyNameBase;
ontologyHash.put(ontologyNameBase, currentOntology);

boolean hasError = false;
SAXParserFactory factory = SAXParserFactory.newInstance();
factory.setNamespaceAware(true);

InputStream stream = null;
try {
    // Parse the input
    SAXParser saxParser = factory.newSAXParser();
    XMLReader xmlReader = saxParser.getXMLReader();
    // Use an instance of the class as the SAX event handler
    xmlReader.setProperty("http://xml.org/sax/properties/lexical-handler", this);
    pass = 1;
    stream = getClass().getResourceAsStream(ontologyFileName);
    saxParser.parse(stream, this); // register classes and properties only
    textBuffer = null;
    pass = 2;
    stream.close();
    stream = getClass().getResourceAsStream(ontologyFileName);
    saxParser.parse(stream, this); // process the rest
} catch (SAXParseException spe) {
    // Error generated by the parser
    System.out.println("\n** Parsing error"
        + ", line " + spe.getLineNumber()
        + ", uri " + spe.getSystemId());
    System.out.println(" " + spe.getMessage() );
    // Use the contained exception, if any
    Exception x = spe;
    if (spe.getException() != null)
        x = spe.getException();
    x.printStackTrace();
} catch (SAXException sxe) {
    // Error generated by this application
    // (or a parser-initialization error)
    Exception x = sxe;
    if (sxe.getException() != null)
        x = sxe.getException();
    x.printStackTrace();
} catch (ParserConfigurationException pce) {
    // Parser with specified options can't be built
    pce.printStackTrace();
} catch (IOException ioe) {
    // I/O error
    ioe.printStackTrace();
} catch (Throwable t) {
    t.printStackTrace();
} finally {
    try {
        try {
            if (stream != null) {
                stream.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

// Return a vector of super class OwlClass objects for className
public Vector<OwlClass> superClasses(String className) {
    Vector<OwlClass> classes = new Vector<OwlClass>();
    OwlClass o = findClass(className);
    if (o == null) return null;
    Iterator<OwlClass> i = o.subClassOf.iterator();
    while (i.hasNext())
        classes.add((OwlClass)i.next());
    return classes;
}

```

```

// Return a vector of subclass OwlClass objects for className
public Vector<OwlClass> subClasses(String className) {
    Vector<OwlClass> classes = new Vector<OwlClass>();
    OwlClass o = findClass(className);
    if (o == null) return null;
    Iterator<OwlClass> i = o.superClassOf.iterator();
    while (i.hasNext())
        classes.add((OwlClass)i.next());
    return classes;
}

// Return a vector of equivalent class OwlClass objects for className
public Vector<Object> equivalentClasses(String className) {
    Vector<Object> classes = new Vector<Object>();
    OwlClass o = findClass(className);
    if (o == null) return null;
    Iterator<Object> i = o.equivalentClass.iterator();
    while (i.hasNext())
        classes.add(i.next());
    return classes;
}

// Return a vector of disjoint class OwlClass objects for className
public Vector<OwlClass> disjointClasses(String className) {
    Vector<OwlClass> classes = new Vector<OwlClass>();
    OwlClass o = findClass(className);
    if (o == null) return null;
    Iterator<OwlClass> i = o.disjointWith.iterator();
    while (i.hasNext())
        classes.add((OwlClass)i.next());
    return classes;
}

// Return an array of super class names for className
public String[] superClassNames(String className) {
    Vector<OwlClass> classes = superClasses(className);
    String[] names = new String[classes.size()];
    for (int i = 0; i < classes.size(); i++)
        names[i] = classes.get(i).id;
    return names;
}

// Return an array of subclass names for className
public String[] subClassNames(String className) {
    Vector<OwlClass> classes = subClasses(className);
    String[] names = new String[classes.size()];
    for (int i = 0; i < classes.size(); i++)
        names[i] = classes.get(i).id;
    return names;
}

// Return an array of equivalent class names for className
public String[] equivalentClassNames(String className) {
    Vector<Object> classes = equivalentClasses(className);
    String[] names = new String[classes.size()];
    for (int i = 0; i < classes.size(); i++) {
        Object p = classes.get(i);
        if ((p instanceof String) && (p != null))
            names[i] = (String)p;
        else if ((p instanceof OwlClass) && (p != null))
            names[i] = ((OwlClass)p).id;
    }
    return names;
}

// Return an array of disjoint class names for className
public String[] disjointClassNames(String className) {
    Vector<OwlClass> classes = disjointClasses(className);
    String[] names = new String[classes.size()];
    for (int i = 0; i < classes.size(); i++)
        names[i] = classes.get(i).id;
}

```

```

        return names;
    }

    // Returns className's ontology name
    public String classOntology(String className) {
        OwlClass o = findClass(className);
        return o.ontology.label;
    }

    // Returns propertyName's ontology name
    public String propertyOntology(String propertyName) {
        PropertyClass o = findProperty(propertyName);
        return o.ontology.label;
    }

    // Returns className's namespace
    public String classNamespace(String className) {
        OwlClass o = findClass(className);
        return o.namespace;
    }

    // Returns propertyName's namespace
    public String propertyNamespace(String propertyName) {
        PropertyClass p = findProperty(propertyName);
        return p.namespace;
    }

    // Returns a vector of PropertyClass objects for className
    public Vector<PropertyClass> properties(String className) {
        OwlClass o = findClass(className);
        Iterator i = o.properties.iterator();
        Vector<PropertyClass> v = new Vector<PropertyClass>();
        while (i.hasNext())
            v.add((PropertyClass)i.next());
        return v;
    }

    // Returns an array of property names for className
    public String[] propertyNames(String className) {
        Vector<PropertyClass> p = properties(className);
        String[] names = new String[p.size()];
        for (int i = 0; i < p.size(); i++)
            names[i] = p.get(i).id;
        return names;
    }

    // Returns property type name of propertyName
    public String propertyType(String propertyName) {
        PropertyClass p = findProperty(propertyName);
        return p.propertyType.name();
    }

    // Returns domain name of propertyName
    public String propertyDomain(String propertyName) {
        PropertyClass p = findProperty(propertyName);
        return p.domain.id;
    }

    // Returns range name of propertyName
    public String propertyRange(String propertyName) {
        PropertyClass p = findProperty(propertyName);
        switch (p.propertyType) {
            case DatatypeProperty:
                DatatypeProperty d = (DatatypeProperty)p;
                return d.range;
            case ObjectProperty:
                ObjectProperty o = (ObjectProperty)p;
                return o.range.id;
            default: return "";
        }
    }
}

```

```

    }

    // Returns vector of OwlClass objects for learning order with name "name"
    public Vector<OwlClass> getLearningOrderClasses(String name) {
        return currentOntology.learningOrderHash.get(name);
    }

    String[] classToString(Vector<OwlClass> c) {
        if (c == null) return null;
        String [] names = new String[c.size()];
        for (int j = 0; j < c.size(); j++)
            names[j] = c.elementAt(j).id;
        return names;
    }

    // Returns list of learning order IDs
    public String[] getLearningOrderIDs() {
        Enumeration<String> e = currentOntology.learningOrderHash.keys();
        String[] s = new String[currentOntology.learningOrderHash.size()];
        for (int i = 0; e.hasMoreElements(); i++)
            s[i] = e.nextElement();

        return s;
    }

    // Returns learning order for learning order ID id
    public String[] getLearningOrderNames(String id) {
        return classToString(getLearningOrderClasses(id));
    }

    void printOntology() {
        System.out.println("Ontology general data:");
        if (currentOntology.about != null) System.out.println("Ontology:about: " +
currentOntology.about);
        if (currentOntology.comment != null) System.out.println("Ontology:comment: " +
currentOntology.comment);
        if (currentOntology.label != null) System.out.println("Ontology:label: " +
currentOntology.label);
        if (currentOntology.versionInfo != null)
System.out.println("Ontology:versionInfo: " + currentOntology.versionInfo);
        // Print namespace prefix definitions
        Enumeration<String> j = currentOntology.nsPrefixHash.keys();
        while (j.hasMoreElements()) {
            String prefix = j.nextElement();
            System.out.println("xmlns:" + prefix + "=" +
currentOntology.nsPrefixHash.get(prefix) + "");
        }
        if (currentOntology.base != null) System.out.println("Ontology:base: " +
currentOntology.base);
        if (currentOntology.allDifferent != null)
            System.out.println("Ontology:allDifferent: " + (currentOntology.allDifferent ?
"true" : "false"));
    }

    void printClasses() {
        System.out.println("\nClasses:");
        Enumeration<String> j = currentOntology.owlClassHash.keys();
        while (j.hasMoreElements()) {
            OwlClass o = currentOntology.owlClassHash.get(j.nextElement());
            System.out.println("Class " + o.id + ": namespace = " + o.namespace);
            System.out.println("Class " + o.id + ": ontology = " + o.ontology.label);
            Iterator i = o.subClassOf.iterator();
            while (i.hasNext())
                System.out.println("Class " + o.id + ": subclass of - " +
((OwlClass)i.next()).id);
            i = o.superClassOf.iterator();
            while (i.hasNext())
                System.out.println("Class " + o.id + ": super class of - " +
((OwlClass)i.next()).id);
            i = o.equivalentClass.iterator();
            while (i.hasNext()) {

```

```

Object p = i.next();
if ((p instanceof String) && (p != null))
    System.out.println("Class " + o.id + ": equivalent to - " + (String)p);
else if ((p instanceof OwlClass) && (p != null))
    System.out.println("Class " + o.id + ": equivalent class of - " +
((OwlClass)p).id);
}
i = o.disjointWith.iterator();
while (i.hasNext()) {
    OwlClass c = (OwlClass)i.next();
    if (c != null)
        System.out.println("Class " + o.id + ": disjoint with - " + c.id);
}
i = o.properties.iterator();
while (i.hasNext())
    System.out.println("Class " + o.id + ": property - " +
((PropertyClass)i.next()).id);
i = o.propertyRestrictions.iterator();
while (i.hasNext())
    printPropertyRestriction(o.id, (PropertyRestriction)i.next());

i = o.include.iterator();
while (i.hasNext()) {
    OwlClass c = (OwlClass)i.next();
    if (c != null)
        System.out.println("Class " + o.id + ": includes - " + c.id);
}
i = o.partOf.iterator();
while (i.hasNext()) {
    OwlClass c = (OwlClass)i.next();
    if (c != null)
        System.out.println("Class " + o.id + ": part-of - " + c.id);
}
i = o.implement.iterator();
while (i.hasNext()) {
    OwlClass c = (OwlClass)i.next();
    if (c != null)
        System.out.println("Class " + o.id + ": implements - " + c.id);
}
i = o.implementedBy.iterator();
while (i.hasNext()) {
    Implementor imp = (Implementor)i.next();
    System.out.print("Class " + o.id + ": implemented-by -");
    Iterator ii = imp.impl.iterator();
    while (ii.hasNext()) {
        OwlClass c = (OwlClass)ii.next();
        if (c != null) System.out.print(" " + c.id);
    }
    System.out.println();
}
}
}

void printProperties() {
    System.out.println("\nProperties:");
    Enumeration<String> j = currentOntology.propertyHash.keys();
    while (j.hasMoreElements()) {
        PropertyClass o = currentOntology.propertyHash.get(j.nextElement());
        if (o.namespace != null) System.out.println("Property " + o.id + ": namespace
- " + o.namespace);
        if (o.ontology != null && o.ontology.label != null)
System.out.println("Property " + o.id + ": ontology - " + o.ontology.label);
        if (o.propertyType != null) System.out.println("Property " + o.id + ":
property type - " + o.propertyType.name());
        if (o.domain != null) System.out.println("Property " + o.id + ": domain - " +
o.domain.id);
        if (o.propertyType != null && o.propertyType == PropertyType.DatatypeProperty)
{
            DatatypeProperty dp = (DatatypeProperty)o;
            if (dp.range != null) System.out.println("Property " + o.id + ": range - " +
dp.range);

```



```

    }
    else if (o.propertyType != null && o.propertyType ==
PropertyType.ObjectProperty) {
        ObjectProperty op = (ObjectProperty)o;
        if (op.range != null) System.out.println("Property " + o.id + ": range -
class " + op.range.id);
        if (op.inverseOf != null) System.out.println("Property " + o.id + ": inverse
of property " + op.inverseOf.id);
    }
    Iterator i = o.type.iterator();
    while (i.hasNext())
        System.out.println("Property " + o.id + ": type - " + i.next());
}

void printPropertyRestriction(String className, PropertyRestriction o) {
    if (o.basePropertyName != null) System.out.println("Class " + className + ":
property restriction - base property name - " + o.basePropertyName);
    if (o.baseProperty != null) System.out.println("Class " + className + ":
property restriction - base property obj - " + o.baseProperty.id);
    if (o.type != null) System.out.println("Class " + className + ": property
restriction - type - " + o.type.name());
    if (o.value != null) System.out.println("Class " + className + ": property
restriction - value - " + o.value);
    if (o.valueType != null) System.out.println("Class " + className + ": property
restriction - value type - " + o.valueType);
    if (o.valueClass != null) System.out.println("Class " + className + ": property
restriction - value class - " + o.valueClass.id);
}

void printDataTypes() {
    System.out.println("\nData Types:");
    Enumeration<String> j = currentOntology.dataTypeHash.keys();
    while (j.hasMoreElements()) {
        DataType o = currentOntology.dataTypeHash.get(j.nextElement());
        if (o.about != null) System.out.println("Data type: about = " + o.about);
        if (o.subClassOf != null) System.out.println("Data type: subClassOf = " +
o.subClassOf);
    }
}

void printLearningOrders() {
    System.out.println("\nLearning Orders:");
    String[] ids = getLearningOrderIDs();
    for (int i = 0; i < ids.length; i++) {
        String[] names = getLearningOrderNames(ids[i]);
        System.out.print("Learning Order " + ids[i] + " :");
        for (String s: names)
            System.out.print(" " + s);
        System.out.println();
    }
    System.out.println();
}

//=====
// SAX DocumentHandler methods
//=====

public void setDocumentLocator(Locator l) {
}

public void startDocument() throws SAXException {
}

public void endDocument() throws SAXException {
}

// convert string names to values in enum OwlElement; improving efficiency and
maintainability
OwlElement toEnum(String n) {

```

```

        if (n == null) return OwlElement.NULL;
        final OwlElement elem = OwlElement.valueOf(n.toUpperCase());
        if (elem == null) return OwlElement.NODEF;
return elem;
    }

    // return the value of the attribute with name ending with value of "name"
String attrValue(Attributes attrs, String name) {
    if (attrs == null || name == null) return "";
    for (int i = 0; i < attrs.getLength(); i++) {
        String aName = attrs.getQName(i); // Attr name
        if ("".equals(aName)) aName = attrs.getLocalName(i);
        if (aName.lastIndexOf(name) >= 0)
            return attrs.getValue(i);
    }
    return "";
}

// Reference to class name may not start with "#"
// need revision to deal with general case like global classes with URIs
public OwlClass findClass(String name) {
    if (name.startsWith("#")) name = name.substring(1);
    OwlClass o = currentOntology.owlClassHash.get(name);
    if (o == null) error ("Class " + name + " missing");
    return o;
}

// Reference to property name may not start with "#"
// need revision to deal with general case like global properties with URIs
public PropertyClass findProperty(String name) {
    if (name.startsWith("#")) name = name.substring(1);
    PropertyClass o = currentOntology.propertyHash.get(name);
    if (o == null) error("Property " + name + " missing");
    return o;
}

public DataType findDatatype(String name) {
    if (name.startsWith("#")) name = name.substring(1);
    DataType o = currentOntology.dataTypeHash.get(name);
    if (o == null) error("DataType " + name + " missing");
    return o;
}

static Object stack_at(Stack s, int i) { // index of stack top is 0
    return s.get(s.size() - i - 1);
}

// Current learning order being assembled, and its ID is on stack top
Vector<OwlClass> learningOrderVector;

public void startElement(String namespaceURI,
                        String sName, // simple name
                        String qName, // qualified name
                        Attributes attrs)
throws SAXException {
    OwlElement n = toEnum(sName);
    OwlClass aClass, bClass;
    if (pass == 1) { // solve the problem caused by use before declaration
        switch (n) {
            case CLASS:
                aClass = new OwlClass();
                aClass.namespace = currentOntology.base; // need be revised to the class's
namespace
                aClass.id = attrValue(attrs, "ID");
                aClass.name = attrValue(attrs, "name");
                aClass.ontology = currentOntology;
                currentOntology.owlClassHash.put(aClass.id, aClass);
                break;
            case DATATYPEPROPERTY:
                DatatypeProperty dp = new DatatypeProperty();

```

```

                dp.namespace = currentOntology.base; // of no use, set again in pass
2
                dp.id = attrValue(attrs, "ID");
                dp.ontology = currentOntology;
                dp.propertyType = PropertyType.DatatypeProperty;
                currentOntology.propertyHash.put(dp.id, dp); // generalize it to cover
namespace
                break;
            case DATATYPE:
                DataType dt = new DataType();
                dt.about = attrValue(attrs, "about");
                currentOntology.dataTypeHash.put(dt.about, dt);
                break;
            case OBJECTPROPERTY:
                ObjectProperty op = new ObjectProperty();
                op.namespace = currentOntology.base; // of no use, set again in pass 2
                op.id = attrValue(attrs, "ID");
                op.ontology = currentOntology;
                op.propertyType = PropertyType.ObjectProperty;
                currentOntology.propertyHash.put(op.id, op); // generalize it to cover
namespace
                break;
            default: break;
        }
        return;
    }
    // pass 2 starts here
    switch (n) {
        case RDF:
            stack_object.push(new Object());
            currentOntology.base = attrValue(attrs, "base");
            break;
        case ONTOLOGY:
            stack_object.push(currentOntology);
            currentOntology.about = attrValue(attrs, "about");
            break;
        case COMMENT: stack_object.push(new Object()); break;
        case LABEL: stack_object.push(new Object()); break;
        case VERSIONINFO: stack_object.push(new Object()); break;
        case CLASS:
            aClass = findClass(attrValue(attrs, "ID"));
            if (aClass == null) error("Class " + attrValue(attrs, "ID") + " missing");
            aClass.namespace = currentOntology.base; // need be revised to the class's
namespace
            stack_object.push(aClass);
            break;
        case SUBCLASSOF:
            if (attrValue(attrs, "resource").equals("")) { // base class is a
restriction
                stack_object.push(new Object());
                break;
            }
            if (stack_E.peek() == OwlElement.CLASS) {
                aClass = (OwlClass)stack_object.peek(); // subClassOf must be directly
nested in class or datatype
                bClass = findClass(attrValue(attrs, "resource"));
                aClass.subClassOf.add(bClass);
                bClass.superClassOf.add(aClass);
            }
            else if (stack_E.peek() == OwlElement.DATATYPE) {
                d = (DataType)stack_object.peek(); // subClassOf must be directly
nested in class or datatype
                d.subClassOf = attrValue(attrs, "resource");
            }
            else error("Misplaced subClassOf");
            stack_object.push(new Object());
            break;
        case DISJOINTWITH:
            if (stack_E.peek() != OwlElement.CLASS) error("Misplaced DisjointWith");
            aClass = (OwlClass)stack_object.peek();
            aClass.disjointWith.add(findClass(attrValue(attrs, "resource")));

```

```

stack_object.push(new Object());
break;
    case RESTRICTION:
if (stack_E.peek() != OwlElement.SUBCLASSOF) error("Misplaced Restriction");
PropertyRestriction pr = new PropertyRestriction();
aClass = (OwlClass)stack_at(stack_object, 1);
aClass.propertyRestrictions.add(pr);
stack_object.push(pr);
break;
    case EQUIVALENTCLASS:
if (stack_E.peek() != OwlElement.CLASS) error("Misplaced EquivalentClass");
aClass = (OwlClass)stack_at(stack_object, 0);
if (attrValue(attrs, "resource").startsWith("http:"))
    aClass.equivalentClass.add(attrValue(attrs, "resource"));
else
    aClass.equivalentClass.add(findClass(attrValue(attrs, "resource")));
stack_object.push(new Object());
break;
    case ONPROPERTY:
if (stack_E.peek() != OwlElement.RESTRICTION) error("Misplaced onProperty");
pr = (PropertyRestriction)stack_at(stack_object, 0);
pr.basePropertyName = attrValue(attrs, "resource");
pr.baseProperty = findProperty(pr.basePropertyName);
stack_object.push(new Object());
break;
    case SOMEVALUESFROM:
if (stack_E.peek() != OwlElement.RESTRICTION) error("Misplaced
someValuesFrom");
pr = (PropertyRestriction)stack_at(stack_object, 0);
pr.valueClass = (OwlClass)findClass(attrValue(attrs, "resource"));
pr.type = PropertyRestrictionType.SomeValuesFrom;
stack_object.push(new Object());
break;
    case HASVALUE:
if (stack_E.peek() != OwlElement.RESTRICTION) error("Misplaced hasValue");
pr = (PropertyRestriction)stack_at(stack_object, 0);
pr.valueType = attrValue(attrs, "datatype");
pr.type = PropertyRestrictionType.HasValue;
stack_object.push(new Object());
break;
    case ALLDIFFERENT: stack_object.push(new Object()); break; // have
not been worked on yet
    case DATATYPEPROPERTY:
String name = attrValue(attrs, "ID");
DatatypeProperty dp = (DatatypeProperty)findProperty(name);
dp.namespace = currentOntology.base;
stack_object.push(dp);
break;
    case DOMAIN:
if (stack_E.peek() != OwlElement.OBJECTPROPERTY && stack_E.peek() !=
OwlElement.DATATYPEPROPERTY) error("Misplaced domain");
PropertyClass p = (PropertyClass)stack_at(stack_object, 0);
OwlClass o = (OwlClass)findClass(attrValue(attrs, "resource"));
p.domain = o;
o.properties.add(p);
stack_object.push(new Object());
break;
    case RANGE:
if (stack_E.peek() == OwlElement.OBJECTPROPERTY) {
    ObjectProperty op = (ObjectProperty)stack_at(stack_object, 0);
    op.range = (OwlClass)findClass(attrValue(attrs, "resource"));
}
else if (stack_E.peek() == OwlElement.DATATYPEPROPERTY) {
    dp = (DatatypeProperty)stack_at(stack_object, 0);
    dp.range = attrValue(attrs, "resource");
}
else
    error("Misplaced range");
stack_object.push(new Object());
break;
    case DATATYPE:

```

```

        DataType dt = findDatatype(attrValue(attrs, "about"));
        stack_object.push(dt);
        break;
        case TYPE:
            if ((stack_E.peek() != OwlElement.OBJECTPROPERTY) && (stack_E.peek() !=
OwlElement.DATATYPEPROPERTY)) error("Misplaced type");
            p = (PropertyClass)stack_at(stack_object, 0);
            p.type.add(attrValue(attrs, "resource"));
            stack_object.push(new Object());
            break;
        case OBJECTPROPERTY:
            name = attrValue(attrs, "ID");
            ObjectProperty op = (ObjectProperty)findProperty(name);
            op.namespace = currentOntology.base;
            stack_object.push(op);
            break;
        case INVERSEOF:
            if (stack_E.peek() != OwlElement.OBJECTPROPERTY) error("misplaced
inverseOf");
            op = (ObjectProperty)stack_at(stack_object, 0);
            ObjectProperty op2 = (ObjectProperty)findProperty(attrValue(attrs,
"resource"));
            op.inverseOf = op2;
            op2.inverseOf = op;
            stack_object.push(new Object());
            break;
        case DISTINCTMEMBERS: stack_object.push(new Object()); break; // have not been
worked on yet
        case LEARNINGORDER:
            String id = attrValue(attrs, "ID");
            if (id == null) id = "default";
            stack_object.push(id);
            learningOrderVector = new Vector<OwlClass>();
            break;
        case REF:
            String resourceName = attrValue(attrs, "resource");
            aClass = findClass(resourceName);
            if (stack_E.peek() == OwlElement.LEARNINGORDER)
                if (aClass != null) learningOrderVector.add(aClass);
            if (stack_E.peek() == OwlElement.IMPLEMENTEDBY) {
                Implementor imp = (Implementor)stack_at(stack_object, 0);
                imp.impl.add(aClass);
            }
            stack_object.push(aClass);
            break;
        case INCLUDE:
            aClass = (OwlClass)stack_object.peek();
            bClass = findClass(attrValue(attrs, "resource"));
            aClass.include.add(bClass);
            bClass.partOf.add(aClass);
            stack_object.push(new Object());
            break;
        case PARTOF:
            aClass = (OwlClass)stack_object.peek();
            bClass = findClass(attrValue(attrs, "resource"));
            aClass.partOf.add(bClass);
            bClass.include.add(aClass);
            stack_object.push(new Object());
            break;
        case IMPLEMENT:
            aClass = (OwlClass)stack_object.peek();
            bClass = findClass(attrValue(attrs, "resource"));
            aClass.implement.add(bClass);
            //bClass.implementedBy.add(aClass);
            stack_object.push(new Object());
            break;
        case IMPLEMENTEDBY:
            Implementor imp = new Implementor();
            String s = attrValue(attrs, "resource");
            bClass = null;
            if (s != null && !s.trim().equals("")) bClass = findClass(s);

```

```

    if (bClass != null) imp.impl.add(bClass);
        stack_object.push(imp);
    break;
        default:
            System.out.println("startElement(): undefined element: " +
sName);
    }
    stack_E.push(n);
}

public void endElement(String namespaceURI,
    String sName, // simple name
    String qName // qualified name
    )
throws SAXException {
    OwlElement n = toEnum(sName);
    if (pass == 1) return;
    // pass = 2 starts here
    Object o = stack_object.pop();
    if (n != stack_E.pop())
        error(sName + " occurred at wrong location");
    switch (n) {
        case RDF: break;
        case ONTOLOGY: break;
        case COMMENT:
            if (stack_E.peek() != OwlElement.ONTOLOGY)
                error(sName + " occurred at wrong location");
            currentOntology.comment = text();
            break;
        case LABEL:
            if (stack_E.peek() != OwlElement.ONTOLOGY)
                error(sName + " occurred at wrong location");
            currentOntology.label = text();
            break;
        case VERSIONINFO:
            if (stack_E.peek() != OwlElement.ONTOLOGY)
                error(sName + " occurred at wrong location");
            currentOntology.versionInfo = text();
            break;
        case CLASS:
        case SUBCLASSOF:
        case DISJOINTWITH:
        case RESTRICTION: break;
        case EQUIVALENTCLASS: break;
        case ONPROPERTY:
        case SOMEVALUESFROM: break;
        case HASVALUE:
            PropertyRestriction pr = (PropertyRestriction)stack_at(stack_object, 0);
            pr.value = text();
            break;
        case ALLDIFFERENT: // have not been worked on yet
        case DATATYPEPROPERTY:
        case DOMAIN:
        case RANGE:
        case DATATYPE:
        case TYPE:
        case OBJECTPROPERTY:
        case INVERSEOF:
        case DISTINCTMEMBERS:
        case INCLUDE:
        case PARTOF:
        case IMPLEMENT:
        case REF: break;
        case IMPLEMENTEDBY:
            if (stack_E.peek() == OwlElement.CLASS) {
                OwlClass c = (OwlClass)stack_object.peek();
                c.implementedBy.add((Implementor)o);
            }
            break;
        case LEARNINGORDER:
            currentOntology.learningOrderHash.put((String)o, learningOrderVector);

```

```

        break;
        default:
            System.out.println("startElement(): undefined element: " +
sName);
            System.exit(1);
        }
    }

    public void startPrefixMapping(String prefix, String uri) throws SAXException {
        currentOntology.nsPrefixHash.put(prefix, uri);
    }

    public void endPrefixMapping(String prefix) throws SAXException {
    }

    public void characters(char buf[], int offset, int len)
    throws SAXException {
        String s = new String(buf, offset, len);
        if (textBuffer == null) {
            textBuffer = new StringBuffer(s);
        } else {
            textBuffer.append(s);
        }
    }

    public void ignorableWhitespace(char buf[], int offset, int len) {
    }

    public void processingInstruction(String target, String data) {
    }

    void error(String message) {
        System.out.println(message);
        //System.exit(1);
    }

    //=====
    // SAX ErrorHandler methods
    //=====

    // treat validation errors as fatal
    public void error(SAXParseException e)
    throws SAXParseException {
        throw e;
    }

    // dump warnings too
    public void warning(SAXParseException err)
    throws SAXParseException {
        System.out.println("*** Warning"
            + ", line " + err.getLineNumber()
            + ", uri " + err.getSystemId());
        System.out.println("    " + err.getMessage());
    }

    //=====
    // LexicalEventListener methods
    //=====

    public void comment(char[] ch, int start, int length)
    throws SAXException {
    }

    public void startCDATA()
    throws SAXException {
        text(); // echo anything we've seen before now
        //emit("START CDATA SECTION");
    }

    public void endCDATA() throws SAXException {
        text(); // echo the CDATA text
    }

```

```
        //emit("END CDATA SECTION");
    }

    public void startEntity(java.lang.String name)
    throws SAXException {
    }

    public void endEntity(java.lang.String name)
    throws SAXException {
    }

    public void startDTD(String name, String publicId, String systemId) {
    }

    public void endDTD() throws SAXException {
    }

    //=====
    // Utility Methods ...
    //=====

    // Return text accumulated in the character buffer
    private String text() throws SAXException {
        if (textBuffer == null) return null;
        String s = "" + textBuffer;
        textBuffer = null;
        return s.trim();
    }
}
}
```


Appendix C

OWLType.java

```

import java.util.*;

class Ontology {
    String about;          // url for this ontology
    String comment;       // comments for this ontology
    String label;         // title of the ontology
    String fileName;      // name of the ontology file
    String filePath;      // path to the ontology file
    String versionInfo;   // ontology version
    Hashtable<String, String> nsPrefixHash = new Hashtable<String, String>(); //
nsPrefix -> url
    Hashtable<String, OwlClass> owlClassHash = new Hashtable<String, OwlClass>(); // full
name -> class object
    Hashtable<String, PropertyClass> propertyHash = new Hashtable<String, PropertyClass>();
// name -> property object
    Hashtable<String, DataType> dataTypeHash = new Hashtable<String, DataType>();
    String base;         // the base URL for the ontology
    Boolean allDifferent; // no shared objects among the classes in the ontology; default
to be false
    Vector<OrderClass> learningOrder = new Vector<OrderClass>(); // recommended learning
order lists
}

class OrderClass {
    Vector<OwlClass> list = new Vector<OwlClass>();
}

class OwlClass {        // Each object represents an OWL class
    String id;          // class tag name (no space)
    String name;       // extended natural name allowing spaces
    String namespace;  // class namespace
    Ontology ontology; // the ontology object containing this class
    Vector<OwlClass> subClassOf = new Vector<OwlClass>(); // list of super
classes
    Vector<OwlClass> superClassOf = new Vector<OwlClass>(); // list of subclasses
    Vector<Object> equivalentClass = new Vector<Object>(); // list of equivalent
classes
    Vector<OwlClass> disjointWith = new Vector<OwlClass>(); // list of disjoint
classes
    Vector<PropertyClass> properties = new Vector<PropertyClass>(); // list of
properties for this class
    Vector<PropertyRestriction> propertyRestrictions = new
Vector<PropertyRestriction>(); // list of restrictions on the properties
    Vector<OwlClass> includes = new Vector<OwlClass>(); // list of included
(reverse of part-of) classes
    OwlClass partof;    // this class is part of which class
}

class PropertyRestriction { // Each instance represents an restriction on a property
    String basePropertyName;
    PropertyClass baseProperty;
    PropertyResctrictionType type;
    String value;        // hasValue
    String valueType;   // hasValue
    OwlClass valueClass; // someValuesFrom
}

```

```

}

class PropertyClass { // Each object represents an OWL property class
    String id; // name of the property
    String namespace; // property namespace
    Ontology ontology; // the ontology object containing this property
    PropertyType propertyType; // DatatypeProperty or ObjectProperty
    OwlClass domain; // Property domain
    Vector<String> type = new Vector<String>(); // list of properties like transitive,
symmetric, ...
}

class DatatypeProperty extends PropertyClass { // A data type property: the range is a
string
    String range; // Property range
}

class ObjectProperty extends PropertyClass { // An object property: the range is a class
    OwlClass range; // Property range
    ObjectProperty inverseOf; // if it is not null, it points to its inverse property
object
}

class DataType { // Declares a data type or its restriction
    String about; // what is a data type; hash key
    String subclassOf; // id is a subclass of which DataType
}

// The following enums are to be extended as needed
// The enums are for improving code reability and execution efficiency - replacing string
comparison with integer comparison
enum PropertyType {DatatypeProperty, ObjectProperty}
//enum Property {Inverse}
//enum ObjPropType {Transitive, Symmetric}
//enum ObjPropRelation {Inverse}
enum PropertyRestrictionType {SomeValuesFrom, HasValue}
enum OwlElement {RDF, ONTOLOGY, COMMENT, LABEL, VERSIONINFO, CLASS, SUBCLASSOF,
DISJOINTWITH, RESTRICTION,
    EQUIVALENTCLASS, ONPROPERTY, SOMEVALUESFROM, HASVALUE, ALLDIFFERENT,
DATATYPEPROPERTY, DOMAIN, RANGE,
    DATATYPE, TYPE, OBJECTPROPERTY, INVERSEOF, DISTINCTMEMBERS,
    NODEF, // no definition; probably new values need be added to the enum
    NULL, // the String version is null; for debugging
    INCLUDES,
    LEARNINGORDER, LIST,
}

```

Appendix D

Web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <display-name>pace-jena</display-name>

  <!-- Spring MVC -->
  <listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
  <servlet>
    <servlet-name>SpringDispatcher</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>SpringDispatcher</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
      /WEB-INF/spring-database.xml,
      /WEB-INF/spring-security.xml
    </param-value>
  </context-param>

  <!-- Spring Security -->
  <filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

  <session-config>
    <session-timeout>30</session-timeout>
  </session-config>

</web-app

```

Appendix E

Spring-database.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

  <bean id="dataSource"

  class="org.springframework.jdbc.datasource.DriverManagerDataSource">

    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://localhost:3306/pacejena"
  />

    <property name="username" value="root" />
    <property name="password" value="123456" />
  </bean>

</beans>
```

SpringDispatcher-servlet.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:util="http://www.springframework.org/schema/util"
  xsi:schemaLocation="
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
  http://www.springframework.org/schema/mvc
  http://www.springframework.org/schema/mvc/spring-mvc.xsd
  http://www.springframework.org/schema/context
  http://www.springframework.org/schema/context/spring-context-3.0.xsd
  http://www.springframework.org/schema/util
  http://www.springframework.org/schema/util/spring-util.xsd">

  <context:component-scan base-package="edu.pace.pacejena.*" />

  <bean

  class="org.springframework.web.servlet.view.InternalResourceViewResolver"
  >

    <property name="prefix">
      <value>/WEB-INF/views/</value>
    </property>
    <property name="suffix">
      <value>.jsp</value>
  </bean>
```

```
        </property>
    </bean>

    <mvc:resources mapping="/resources/**" location="/resources/" />
    <mvc:resources mapping="/webjars/**" location="classpath:/META-INF/resources/webjars/" />

    <bean id="jacksonMessageChanger"

        class="org.springframework.http.converter.json.MappingJacksonHttpMessageC
onverter">
        <property name="supportedMediaTypes" value="application/json" />
    </bean>

    <bean

        class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHan
dlerAdapter">
        <property name="messageConverters">
            <util:list id="beanList">
                <ref bean="jacksonMessageChanger" />
            </util:list>
        </property>
    </bean>
</beans>
```

Appendix F

PaceJenaAuthenticationSuccessHandler.java

```

package edu.pace.pacejena.auth;

import java.io.IOException;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.web.DefaultRedirectStrategy;
import org.springframework.security.web.RedirectStrategy;
import org.springframework.security.web.WebAttributes;
import org.springframework.security.web.authentication.AuthenticationSuccessHandler;

import edu.pace.pacejena.constants.PaceJenaConstants;
import edu.pace.pacejena.model.User;
import edu.pace.semweb.OntologyStructureBuilder;
import edu.pace.semweb.PaceJena;
import edu.pace.semweb.tree.OntologyNode;

public class PaceJenaAuthenticationSuccessHandler implements
AuthenticationSuccessHandler {
    protected Log logger = LogFactory.getLog(this.getClass());

    private RedirectStrategy redirectStrategy = new DefaultRedirectStrategy();

    @Override
    public void onAuthenticationSuccess(HttpServletRequest request,
        HttpServletResponse response, Authentication authentication) throws
IOException {
        clearAuthenticationAttributes(request);
        HttpSession session = request.getSession();
        User authUser = SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        session.setAttribute("username", authUser.getUsername());
        session.setAttribute("authorities", authentication.getAuthorities());

        PaceJena paceJena = (PaceJena) session.getAttribute("PaceJena");
        if (paceJena == null) {
            //ServletContext sc = session.getServletContext();
            //String fullPath = sc.getRealPath("/WEB-INF/owl/webTutorial.owl");
            //System.out.println(fullPath);

```

```

        paceJena = new PaceJena("/webTutorial.owl");
        session.setAttribute(PaceJenaConstants.PACE_JENA_SESSION_KEY,
paceJena);
    }
    loadOntologyTreeIntoSession(session,                paceJena,
authUser.getLearningLevel());

    //set our response to OK status
    response.setStatus(ServletResponse.SC_OK);

    //since we have created our custom success handler, its up to us to
where
    //we will redirect the user after successfully login
    response.sendRedirect("/pace-jena/home");
}

protected void clearAuthenticationAttributes(HttpServletRequest request) {
    HttpSession session = request.getSession(false);
    if (session == null) {
        return;
    }
    session.removeAttribute(WebAttributes.AUTHENTICATION_EXCEPTION);
}

public static void loadOntologyTreeIntoSession(HttpSession session,
PaceJena parser, int learningOrder) {
    final List<String> rootClassNames = parser.getRootClassNames();

    final String parentName = rootClassNames.get(0);
    final OntologyNode rootNode = new OntologyNode(null, parentName);
    OntologyStructureBuilder.loadCurrentNode(parser,        rootNode,
parser.getLearningOrderNames(PaceJenaConstants.LEARNING_LEVELS[learningOrder]))
;

    session.setAttribute(PaceJenaConstants.ONTOLOGY_ROOT_NODE_SESSION_KEY,
rootNode);
}
}

```

References

- [1] Carl Albing, and Michael Schwartz, “*Java Application Development on Linux*”, Retrieved Nov 26, 2016
http://everythingcomputerscience.com/books/013143697X_book.pdf
- [2] A. Altowayan, and L. Tao, “*Simplified Approach for Representing Part-Whole Relations in OWL-DL Ontologies.*” In The IEEE International Symposium on Big Data Security on Cloud, pages 1399–1405, New York, NY, USA, 2015. IEEE.
- [3] L. Aroyo, and D. Dicheva, “*AIMS: Learning and Teaching Support for WWW-Based Education,*” International Journal of Continuing Engineering Education and Life-Long Learning, vol. 11, pp. 152-164, 2001.
- [4] Franz Baader, LuFg, “*Logic Based Knowledge Representation*”, Theoretical Computer Science, RWTH, Aachen, AhonstraBe, 55, 52074 Aachen Germany
- [5] M. Buchheit, Nutt, W.M. Jeusfeld, and M. Staudt, “*Subsumption of Queries to Object Oriented Databases. Information Systems*”, 19(1):33-54, 1994.
- [6] P. Devanbu, R. J. Brachman, P.G. Selfridge, and B.W. Ballard, “*LASSIE: A Knowledge-Based Software Information System*”. Communications of the ACM, 34(5):34-49, 199
- [7] Sandra Dominiek, Jan-Ola Östman, and Jef Verschueren, eds. “*Cognition and Pragmatics*”, Vol. 3. John Benjamins Publishing, 2009.
- [8] J. Falkner, K. Jones, “*Servlets and JSP: The J2EE Web Tier*”, Addison-Wesley Pub Co., (2003)
- [9] K. Gai, “*A Report About Suggestions on Developing e-Learning in China.*” In 2010 International Conference on E-Business and E-Government, pages 609–613, Guangzhou, China, 2010. IEEE.
- [10] K. Gai, M. Thuraisingham, B. Qiu, and L. Tao, “*Proactive Attribute Based Secure Data Schema for Mobile Cloud in Financial Industry.*” In The IEEE International Symposium on Big Data Security on Cloud; 17th IEEE International Conference on High Performance Computing and Communications, pages 1332–1337, New York, USA, 2015.
- [11] James Garson, “*Modal Logic*”, The Stanford Encyclopedia of Philosophy (Spring 2016 Edition), Edward N. Zalta (ed.), URL = <http://plato.stanford.edu/archives/spr2016/entries/logic-modal/>.
- [12] James Gosling, and Bill Joy, “*The Java Language Specification Language, Second Edition*”,
<https://jcp.org/aboutJava/communityprocess/maintenance/JLS/jls2draft.pdf>
- [13] Sabine Grunwald, and K. Ramesh Reddy, “*Concept Guide on Reusable Learning Objects with Application to Soil, Water and Environmental Sciences.*” Retrieved on July 28 (2007): 2011.

- [14] Praveen Gupta, Bari Pacheri, and M. C. Govil. "*Spring Web MVC Framework for Rapid Open Source J2EE Application Development: A Case Study.*" *Interface* 2.6 (2010): 1684-1689.
- [15] Eliotte Rusty Harold, "*Processing XML with Java*", Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [16] John, Hebel, "*Semantic Web Programming.*" John Wiley & Sons, 2011.
- [17] D. Hu, J. Yan, J. Zhao, and Z. Hua, "*Ontology-Based Scenario Modeling and Analysis for Bank Stress Testing.*" *Decision Support Systems*," 63:81–94, 2014.
- [18] R. Hubscher, and S. Puntambekar, "*Adaptive Navigation for Learners in Hypermedia is Scaffolded Navigation,*" in Proc. of Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems , May 29-31, 2002, Malaga, Spain, pp. 184-192.
- [19] David Hunter, Jon Rafter, Andrew Watt, and Linda McKinnon. *Beginning XML.* John Wiley & Sons, 2011.
- [20] Inderjeet, Singh and Sterns, Beth, "Design Enterprise Application", http://download.oracle.com/otn-pub/java/designing_enterprise_apps/2.0/designing_enterprise_apps-2_0-book.pdf?AuthParam=1483472529_ccd371a21a50885fa273f3f8fa7c7adc, Retrieved on Nov 28, 2016
- [21] L. Johnson, (2003). "*Elusive Vision: Challenges Impeding the Learning Object Economy*", San Francisco: Macromedia Inc. Retrieved May 22, 2011 from http://www.nmc.org/pdf/Elusive_Vision.pdf
- [22] J. Koehler, "*An Application of Terminological Logics to Case-Based Reasoning*", In Proceeding of the fourth International Conference on Principles of Knowledge Representation and Reasoning, KR'94, pages 351 362, Bonn, Germany, 1994. Morgan Kaufmann.
- [23] F. Liu, H. Chen, L. Lo, and W. Lee, "*Comprehensive Security Integrated Model and Ontology within Cloud Computing.*" *J. of Internet Technology*, 14(6):935–946, 2013.
- [24] Marvin Minsky, "*Framework for Representing Knowledge*", MIT-AI Laboratory Memo 306, June, 1974
- [25] N.J. Nilsson, "*Logic and Artificial Intelligence*", *Artificial Intelligence* 47 (1990) 31-56
- [26] J. Quantz, and B. Schmitz, "*Knowledge-Based Disambiguation for Machine Translation*", *Minds and Machines*, 4:39-57, 1994
- [27] M. Qiu, K. Gai, B. Thuraisingham, L. Tao, and H. Zhao. Proactive "*User-Centric Secure Data Scheme Using Attribute-Based Semantic Access Controls for Mobile Clouds in Financial Industry.*" *Future Generation Computer Systems*, PP:1, 2016.
- [28] H. Song, L. Zhong, Wang, H., Li, R. and Xia, H., "*Constructing An Ontology for Web-Based Learning Resource Repository,*" in Proc. Of Workshop on Applications of Semantic Web Technologies for e-Learning, October 2-5, 2005, Banff, Canada.
- [29] Barbara Starr, "*A Layman's Visual Guide to Google's Knowledge Graph Search API*", from <http://searchengineland.com/laymans-visual-guide-googles-knowledge-graph-search-api-241935>

- [30] M. Tan, and A. Goh, “*The Use of Ontologies in Web-Based Learning*,” in Proc. Of Workshop on Applications of Semantic Web Technologies for e-Learning, November 8, 2004, Hiroshima, Japan.
- [31] L. Tao, S. Golikov, K. Gai, and M. Qiu, “*A Reusable Software Component for Integrated Syntax and Semantic Validation for Services Computing*.” In 9th Int’l IEEE Symposium on Service-Oriented System Engineering, pages 127–132, San Francisco Bay, USA, 2015.
- [32] J.R. Wright, E.S. Brown, K. Weixelbaum, G.T. Vesonder, S.R. Palmer, J.T. Berman, and H.H. Moor, “*A Knowledge-Based Configurator that Supports Sales, Engineering, and Manufacturing at AT&T Network Systems*”, AI Magazine, 14(3):69-80, 1993
- [33] M V Uttam Tej, Dhanaraj Cheelu, Babu M.Rajasekhara, and P Venkata Krishna, “*Analyzing XML Parsers Performance for Android Platform*”. VIT University, Tamil Nadu, 2011.
- [34] Aleksa Vukotic, and James Goodwill, “*Apache Tomcat 7*”, Chapter 7, Apress, Sept 5, 2011
- [35] Fangju Wang, Jing Li, and Hooman Homayounfar, “*A Space Efficient XML DOM Parser. Data & Knowledge Engineering*,” 60(1):185{207, 2007.
- [36] J.T. Yao, and Y.Y. Yao, “*Web-Based Support Systems*”, Proceedings of the WI/IAT Workshop on Applications, Products and Services of Web-based Support Systems, pp. 1-5, 2003.
- [37] Dublin Core Metadata Initiative. “*DCMI Metadata Basics*.” Dublin Core Metadata Initiative (2014).
- [38] IMS Global Learning Consortium. “*IMS Meta-data Best Practice Guide for IEEE 1484.12. 1-2002 Standard for Learning Object Metadata*.” (2006).
- [39] IMS Global Learning Consortium. “*IMS Learning Resource Meta-data Specification*.” (2001).
- [40] “*Java Server-Side Programming*”
<https://www.ntu.edu.sg/home/ehchua/programming/java/JavaServlets.html>, Retrieved on Nov 30, 2016
- [41] Mozilla Development Network, <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction>, Retrieved on Nov 26, 2016
- [42] Oracle “*Asynchronous JavaScript Technology and XML (Ajax) With the Java Platform*”, <http://www.oracle.com/technetwork/articles/java/ajax-135201.html>, Retrieved on Nov 29, 2016
- [43] SCORM, ADL. “*Advanced Distributed Learning*.” SCORM Overview (2004).
- [44] Summernote, license under MIT, <http://summernote.org/>, Retrieved on Sept 6, 2016.