

References

- [1]. Gadomski A.M., 1987"Application of System-Process-Goal Approach for description of TRIGA RC1 System", Proceedings of 9th.European TRIGA Nuclear Reactor Users Conference,Roma:GA Technologies,TOC-19 USA.
- [2] Learning ,Available : <http://en.wikipedia.org/wiki/learning>
- [3]. Navigli R., Velardi P., Gangemi A., 2008, TextOntoEx: Automatic Ontology Construction from Natural English Text IEEE Intelligent Systems, vol. 18:1,
- [4]. Maedche, A., Staab, S.: 2000, "Semi-automatic Engineering of Ontologies from Text." In: Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering.
- [5] Raghu Anantharangachar, Srinivasan Ramani and S Rajagopalan Ontology ,2013, Guided Information Extraction from Unstructured Text, International Journal of Web & Semantic Technology (IJWesT) Vol.4, No.1.
- [6] Mc Guinness, D, 2004,Question Answering on the Semantic Web. IEEE Intelligent Systems, 19 (1) pp 82-85
- [7] Vanessa Lopez, Miriam Fernández, Nico Stieler and Enrico Motta, PowerAqua: supporting users in querying and exploring the Semantic Web content
- [8] Tablan, V, Damljanovic, D., and Bontcheva, K. (2008). A Natural Language Query Interface to Structured Information., In the 5th European Semantic Web Conference.
- [9] Witte, R., N. Khamis, and J. Rilling, 2010, "Flexible Ontology Population from Text: The OwlExporter", International Conference on Language Resources and Evaluation (LREC), Valletta, Malta : ELRA, pp. 3845--3850, May 19—21.
- [10] Christian Chiarcos, Stefanie Dipper, Michael Gotze, Ulf Leser, Anke Ludeling, Julia Ritz, and Manfred Stede. " 2008. A Flexible Framework for Integrating Annotations from Different Tools and Tagsets. In Traitement Automatique des Langues.
- [11] Diana Maynard, Adam Funk, and Wim Peters , SPRAT: a tool for automatic semantic pattern-based ontology population.

- [12] Maynard, D., Li, Y., Peters, W.: 2008, NLP Techniques for Term Extraction and Ontology Population. In Buitelaar, P., Cimiano, P., eds.: Bridging the Gap between Text and Knowledge - Selected Contributions to Ontology Learning and Population from Text. IOS Press
- [13] Hoifung Poon and Pedro Domingos, Unsupervised Ontology Induction from Text, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Pages 296-305
- [14] Philipp Cimiano. 2006. Ontology learning and population from text. Springer.
- [15] FeiWu and Daniel S.Weld. 2008. Automatically refining the wikipedia info box ontology. In *Proceedings of the Seventeenth International Conference on World Wide Web*, Beijing, China.
- [16] Giannis Varelas, Epimenidis Voutsakis, Paraskevi Raftopoulou, 2006, Semantic Similarity Methods in WordNet and their Application to Information Retrieval on the Web. International journal on semantic web and Ontologies Volume 2, Issue 3.
- [17] Maedche, A. and Staab, S, 1999: Ontology Learning. In S. Staab & R. Studer (eds.), 2003, Handbook on Ontologies in Information Systems. Heidelberg: Springer.
- [18]. Faure, D. and Nédellec, C, 2007, Knowledge Acquisition of Predicate Argument Structures from Technical Texts Using Machine Learning: The System ASIUM. LNCS 1621. Springer-Verlag, Heidelberg. 329-334.
- [19] Quirk, R. 1995 Grammatical and Lexical Variance in English. Longman, London & New York
- [20] Aston, G. and Burnard, L, 1998, The BNC Handbook: Exploring the British National Corpus. Edinburgh University Press
- [21] Natural Language Processing, Available : http://en.wikipedia.org/wiki/Natural_language_processing
- [22] J. Piskorski, Yangarber R, T. Poibeau et al. (eds.), 2013, Multi-source, Multilingual Information Extraction and Summarization II, Theory and Applications of Natural Language Processing, DOI 10.1007/978-3-642-28569-1__2, © Springer-Verlag Berlin Heidelberg,
- [23] WordNet, Available : <http://wordnet.princeton.edu>, Access date : 15.05.2014

- [24] Laresgoiti, I., Anjewierden, A., Bernaras,A., Corera, J., Schreiber, A. Th., and Wielinga, B. J.: ,1996,Ontologies as Vehicles for reuse: a mini-experiment. KAW <http://ksi.cpsc.ucalgary.ca/KAW/KAW96/laresgoiti/k.html>..
- [25]W. Zhou, Z. Liu, and Y. Zhao, 2007, "Ontology learning by clustering based on fuzzy formal concept analysis," In Proceedings of the 31st Annual International Computer Software and Applications Conference, pp. 204-210.
- [26] C. S. Lee, Y. F. Kao, Y. H. Kuo, and M. H. Wang , 2007, "Automated ontology construction for unstructured text documents," Data & Knowledge Engineering, vol. 60, pp. 547-566.
- [27] D. Zhang, L. Zhang, and H. Jiang, , 2010, "Research on Semiautomatic Domain Ontology Construction," Seventh Web Information Systems and Applications Conference, pp. 115-118.
- [28] H. Li, and W. Ko, 2007 ,"Automated food ontology construction mechanism for diabetes diet care," International conference on machine learning and cybernetics, vol. 5, pp.2953-2958.
- [29] Youngia Park, Roy J .Byrd and Branimir K. Boguraev, 2004, "Towards Ontologies On Demand", Proceedings of Workshop on Semantic Web Technologies for Scientific Search and Information Retrieval.



Appendix A: Definitions, Acronyms, and Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
CNL	Control Natural Language
GATE	General Architecture for Text Engineering
GB	Giga Byte
GUI	Graphical User Interfaces
IT	Information Technology
KB	Knowledge Base
NLP	Natural Language Processing
OS	Operating System
OWL	Web Ontology Language
OWL-QL	Web Ontology Language-Query Language
PC	Personal Computer
QBS	Question Base Learning System
RAM	Random Access Memory
RRD	Research Review Document
SESLATA	Semantic Self Learning And Teaching Agent
SLIIT	Sri Lanka Institute of Information Technology
SRS	Software Requirement Specification
URL	Uniform Resource Locator

Natural Language processing

A field of computer science and linguistics concerned with the interactions between computers and human (natural) languages. Natural language generation systems convert information from computer databases into readable human language.

Controlled natural languages

All subsets of natural languages, obtained by restricting the grammar and vocabulary in order to reduce or eliminate ambiguity and complexity.

Ontology

A formal representation of the knowledge, by a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to describe the domain.

A knowledge base

A special kind of database for knowledge management, providing the means for the computerized collection, organization, and retrieval of knowledge. Also a collection of data representing related experiences, their results is related to their problems and solutions.

Artificial intelligence

The intelligence of machines and the branch of computer science that aims to create it. Textbooks define the field as "the study and design of intelligent agents," where an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success, which coined the term in 1956, defines it as "the science and engineering of making intelligent machines."

Inference engine

A computer program that tries to derive answers from a knowledge base. It is the "brain" that expert systems use to reason about the information in the knowledge base for the ultimate purpose of formulating new conclusions. Inference engines are considered to be a special case of reasoning engines, which can use more general methods of reasoning.

Parsing

The process of analyzing a text, made of a sequence of tokens (for example, words), to determine its grammatical structure with respect to a given (more or less) formal grammar.

Annotation

A summary made of information in a book, document, online record, video, software code or other information. Commonly this is used, for example, in draft documents, where another reader has written notes about the quality of a document at a certain point, "in the margin", or perhaps just underlined or highlighted passages.

System interfaces

The System interacts with several other software applications. Several API s will be implemented over the SESLATA system, in order to interact with those software systems.

SESLATA 2 GATE API

(API for GATE-5.1)

This is the API which uses to interact with the GATE-5.1 IDE. The GATE is responsible of assist the SESLATA in natural language processing activities.

SESLATA 2 WORDNET API

(API for word net 2.1)

The word Net 2.1 system assists in generating synonyms for the words broken in the early stage. The special API will be developed to interact with the word net 2.1

SESLATA 2 Protégé API

(API for Protégé 4.1)

Protégé 4.1.2 plays a major role in ontology creation of the SESLATA system. this is the API used for the purpose of interacting with the protégé 4.1.2

SESLATA 2 JENA API

(API for Jena Reasoner)

Jena reasoner is the built in inference engine for the SESLATA system. This will be the API developed for the interacting with the jena reasoner.

Software interfaces

The SESLATA interacts with different software components in order to achieve its task in different stages of component. All these software are open source and the source codes and customization options are freely available to the entire world.

Word Net-2.1

Word Net is a large lexical database of English language which groups English words into sets of synonyms (Nouns, verbs, adjectives, adverbs) called synsets. It provides short, general definitions, and records the various semantic relations between these synonym sets. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. An API which supports Word Net will be used to perform automatic text analysis throughout the research.

JENA Reasoner

Jena is an open source Semantic Web framework for Java. It provides a suitable programming environment for RDF, RDFS, OWL, SPARQL and an integrated rule-based inference mechanism. Jena can be used to import OWL ontology into suitable models, process them and export them to OWL files. Once a query has been interpreted, reasoning process should take place in finding the most accurate answer, and this part is done by jena reasoner. Due to its storage abstraction jena enables new storage subsystems to be

Protégé 4.1.2

Protégé is a free, open source ontology editor and knowledge-base framework. The Protégé platform supports two main ways of modeling ontologies via the Protégé-Frames and Protégé-OWL editors. Protégé ontologies can be exported into a variety of formats including RDF(S), OWL, and XML Schema. Protégé is based on Java, is extensible, and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development.



Protégé software is currently transitioning from the current version (Protégé 3.X) to an entirely new release (Protégé 4). Protégé 3.X is widely used and the adoption of Protégé 4 is slowing happening, because of this, you will find resources for both versions

User Operations

The system provides various operations and features depending on the user type.

Ordinary user:

The user has given different priorities depending on the status of logging or not.

The logged users:

- Operate the system by inserting the questions and getting explanations
- Powered by the knowledge updating engine features
- Ability of continue the previous logging sessions

The un-logged users will only be able to operate with the first function given in the above list.

Domain Expert:

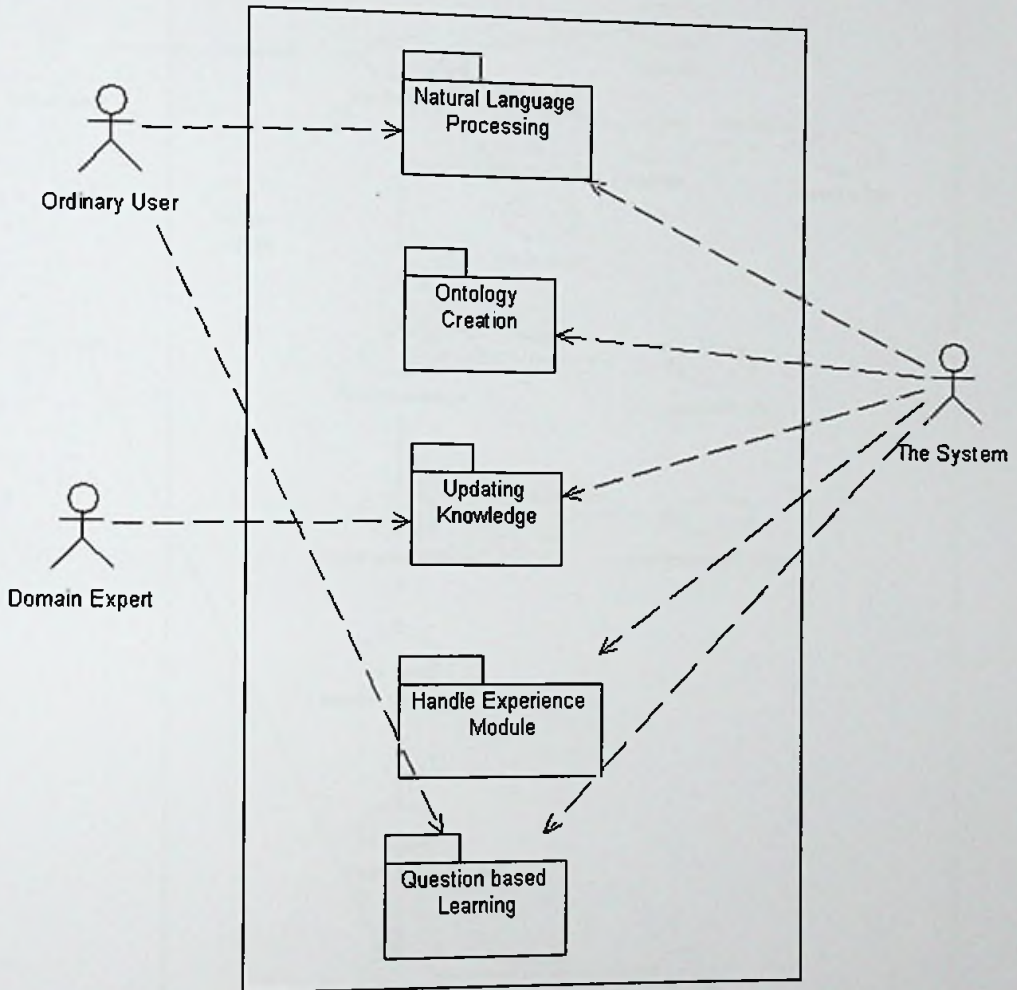
- Involving with learning activities
- Provide feed backs in both learning and teaching activities
- Refining the current processes

System Administrators:

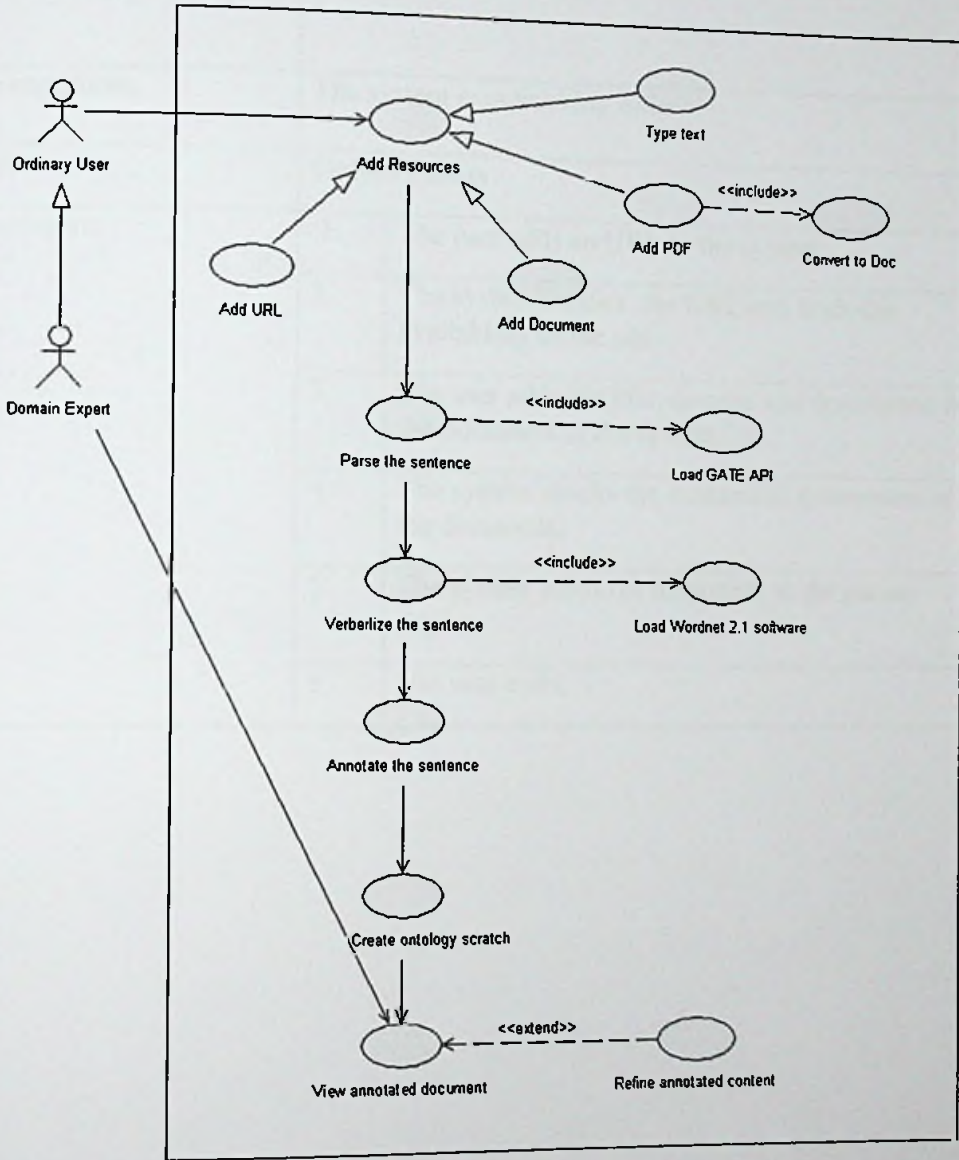
Training activities relevant to both learning and teaching activities All stored contents will back up once a week. Different backup server will be used for the backup processes.

Appendix B: Design Diagrams of the system

Main Use Case Diagram



Natural Language Processing



Use case scenarios

Use case 1.1	Add resources - URL	
Pre conditions	The system is in working order.	
Actors	Ordinary users	
Description	1.	The user adds an URL to the system
	2.	The system verifies the URL and finds the availability of the site
	3.	The user adds the title, domain and description of the document to the system.
	4.	The system checks the syntactical correctness of the document.
	5.	The system sends the document to the parser engine.
	6.	Use case ends.



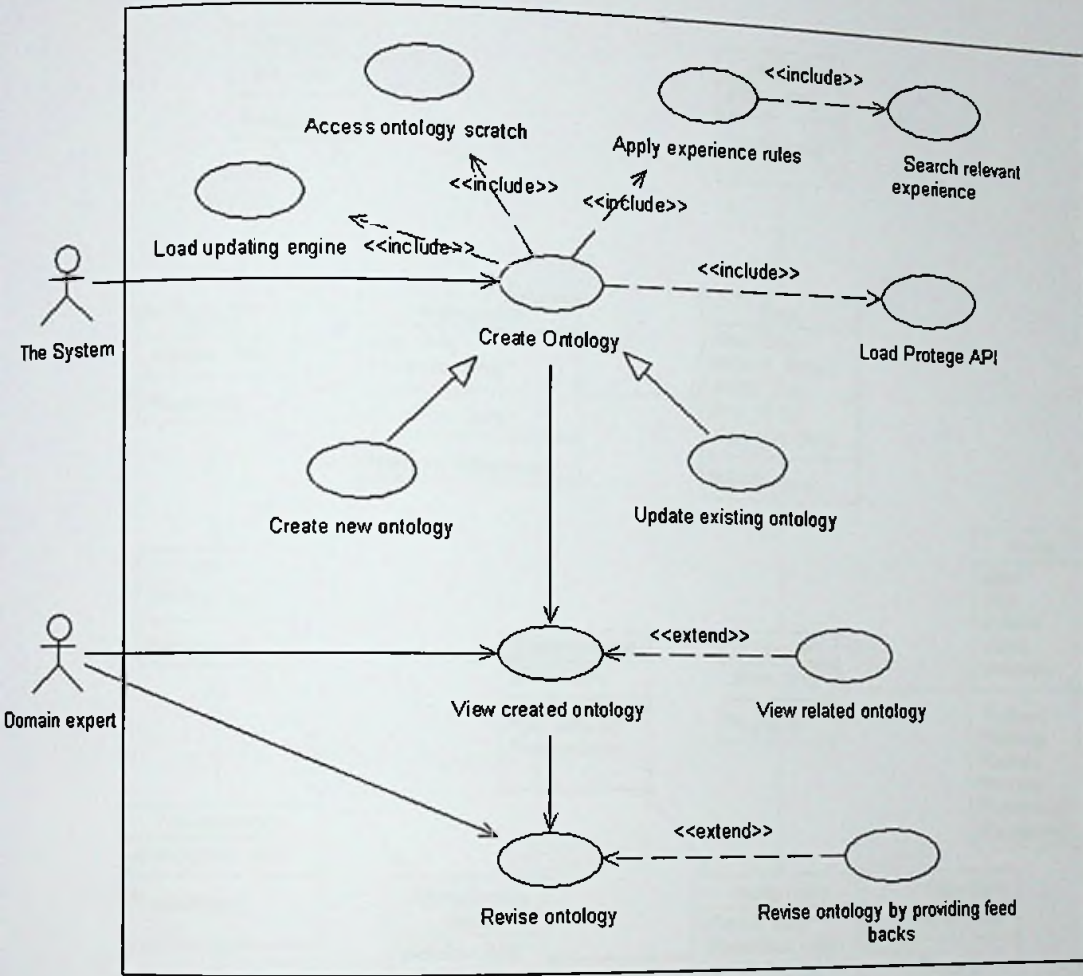
Use case 1.3	Add resources - Document	
Pre conditions	The system is in working order.	
Actors	Ordinary users	
Description	1.	The user adds the title, domain and description of the document to the system.
	2.	The system checks the syntactical correctness of the document.
	3.	The system sends the document to the parser engine.
	4.	Use case ends.

Use case 1.2	Add resources – Paragraph or Page	
Pre conditions	The system is in working order.	
Actors	Ordinary users	
Description	1.	The system reads the paragraph.
	2.	The user adds the title, domain and description of the document to the system.
	3.	The system checks the syntactical correctness of the document.
	4.	The system sends the document to the parser engine.
	5.	Use case ends.

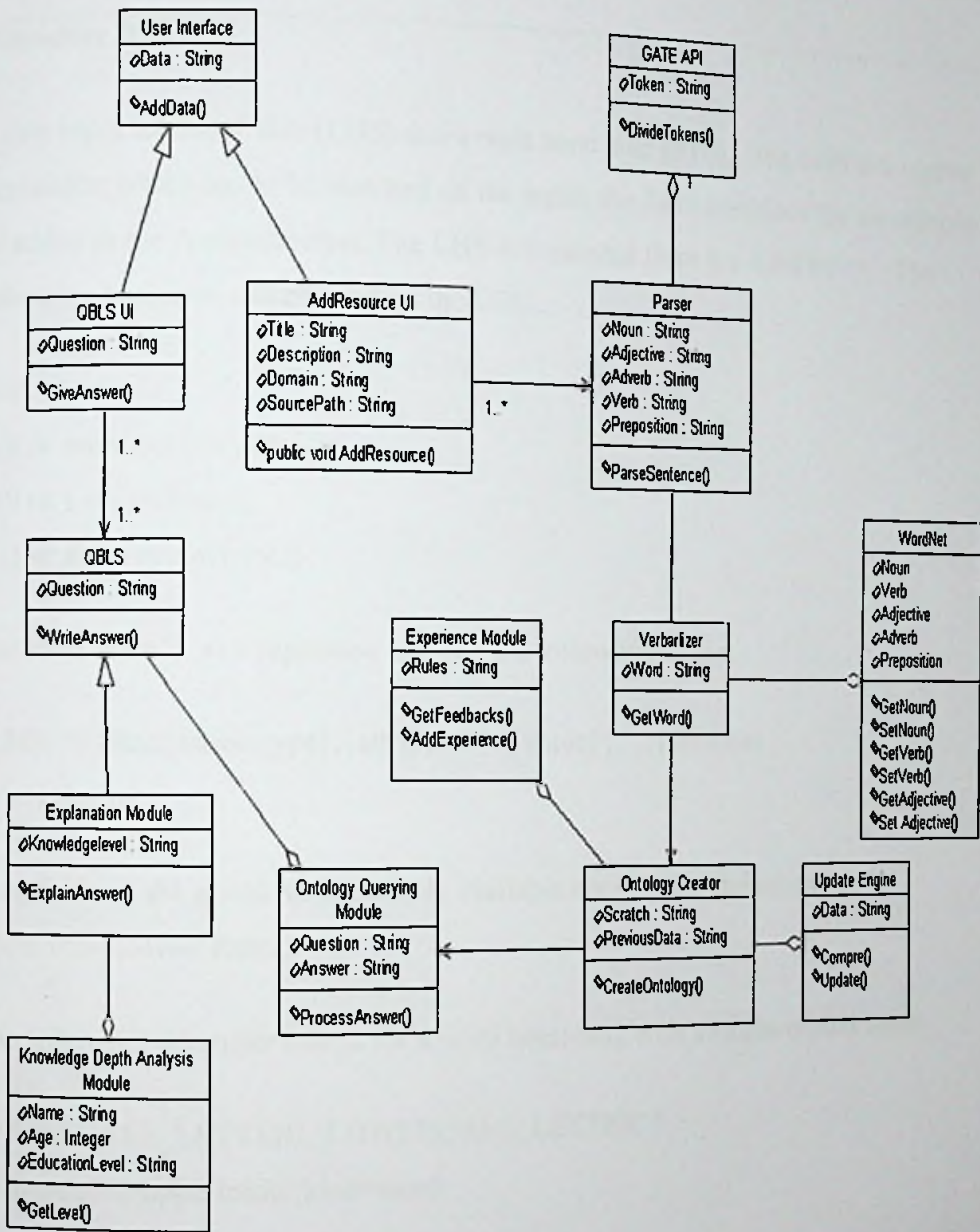
Use case 1.4	Parse the sentence	
Pre conditions	The system is in working order.	
Actors	The system	
Description	1.	The system loads the GATE API.
	2.	GATE API divides the document to sentences and tokens.
	3.	Parser breaks the sentence to verbs, nouns, adverbs, adjectives etc.
	4.	Parser sends the parts to the verbalizer.
	5.	Use case ends.

Use case 1.5	Verbalize the sentence	
Pre conditions	The system is in working order. Parser should separate the sentence into parts.	
Actors	The system	
Description	1.	The system loads the Word Net 2.1 software.
	2.	Verbalizer gives meaning to the sentence by integrating the Word Net software.
	3.	Use case ends.

Ontology creation



Main Class Diagram of the system



Appendix C Algorithms, Flowcharts and Pseudo Codes

Tokeniser Rules.

A rule has a left hand side (LHS) and a right hand side (RHS). The LHS is a regular expression which has to be matched on the input; the RHS describes the annotations to be added to the AnnotationSet. The LHS is separated from the RHS by '>'. The following operators can be used on the LHS:

| (or)

* (0 or more occurrences)

? (0 or 1 occurrences)

+ (1 or more occurrences)

The RHS uses ';' as a separator, and has the following format:

```
{LHS} > {Annotation type};{attribute1}={value1};...;{attribute n}={value n}
```

Details about the primitive constructs available are given in the tokeniser file (DefaultTokeniser.Rules).

The following tokeniser rule is for a word beginning with a single capital letter:

```
'UPPERCASE_LETTER' 'LOWERCASE_LETTER'* >
```

```
Token;orth=upperInitial;kind=word;
```

It states that the sequence must begin with an uppercase letter, followed by zero or more lowercase letters. This sequence will then be annotated as type 'Token'. The attribute 'orth' (orthography) has the value 'upperInitial'; the attribute 'kind' has the value 'word'.

POS Tagging Parameters



CC - coordinating conjunction: 'and', 'but', 'nor', 'or', 'yet', plus, minus, less, times (multiplication), over (division). Also 'for' (because) and 'so' (i.e., 'so that').

CD - cardinal number

DT - determiner: Articles including 'a', 'an', 'every', 'no', 'the', 'another', 'any', 'some', 'those'.

EX - existential 'there': Unstressed 'there' that triggers inversion of the inflected verb and the logical subject; 'There was a party in progress'.

FW - foreign word

IN - preposition or subordinating conjunction

JJ - adjective: Hyphenated compounds that are used as modifiers; happy-go-lucky.

JJR - adjective - comparative: Adjectives with the comparative ending '-er' and a comparative meaning. Sometimes 'more' and 'less'.

JJS - adjective - superlative: Adjectives with the superlative ending '-est' (and 'worst'). Sometimes 'most' and 'least'.

JJSS - -unknown-, but probably a variant of JJS

-LRB- - -unknown-

LS - list item marker: Numbers and letters used as identifiers of items in a list.

MD - modal: All verbs that don't take an '-s' ending in the third person singular present: 'can', 'could', 'dare', 'may', 'might', 'must', 'ought', 'shall', 'should', 'will', 'would'.

NN - noun - singular or mass

NNP - proper noun - singular: All words in names usually are capitalized but titles might not be.

NNPS - proper noun - plural: All words in names usually are capitalized but titles might not be.

NNS - noun - plural

NP - proper noun - singular

NPS - proper noun - plural

PDT - predeterminer: Determiner like elements preceding an article or possessive pronoun; 'all/PDT his marbles', 'quite/PDT a mess'.

POS - possessive ending: Nouns ending in 's' or ''.

PP - personal pronoun

PRPR\$ - unknown-, but probably possessive pronoun

PRP - unknown-, but probably possessive pronoun

PRP\$ - unknown, but probably possessive pronoun, such as 'my', 'your', 'his', 'his', 'its', 'one's', 'our', and 'their'.

RB - adverb: most words ending in '-ly'. Also 'quite', 'too', 'very', 'enough', 'indeed', 'not', '-n't', and 'never'.

RBR - adverb - comparative: adverbs ending with '-er' with a comparative meaning.

RBS - adverb - superlative

RP - particle: Mostly monosyllabic words that also double as directional adverbs.

JAPE Rule : Object Oriented Language

Phase: OBJECTORIENTEDLANGUAGE

Input: Lookup Token

Options: control = appelt

Rule: OBJECTORIENTEDLANGUAGE1

(

{Lookup.majorType == objectorientedlanguage}

(
{Lookup.majorType == objectorientedlanguage}

)?)

)

:objectorientedlanguage

-->

:objectorientedlanguage.OBJECTORIENTEDLANGUAGE = {rule =

"OBJECTORIENTEDLANGUAGE1"}

JAPE Rule: Declarative Language

Phase: DECLARATIVELANGUAGE

Input: Lookup Token

Options: control = appelt

Rule: DECLARATIVELANGUAGE1

```
(
  {Lookup.majorType == declarativelanguage}
  (
    {Lookup.majorType == declarativelanguage}
  )?
)
:declarativelanguage
-->
:declarativelanguage.DECLARATIVELANGUAGE = {rule =
"DECLARATIVELANGUAGE1"}
```

OWL Rules

- If the word is a verb it is a property.
- If the word is an adjective it is a property.
- If the word is a general noun it is a class. (Use word net)
E.g.: Programming Languages, object oriented Languages
- If the word is a specific noun (proper noun) it can be an instance of particular class
E.g.: java , c++ (Most probably they are not dictionary words)
- If the word is a pronoun (it , he, she, they) navigate to the previous sentence' subject noun.

- If sentence contains the numeric values it can be a value a property so navigate to the same and the previous sentence and check for property identification.
- If the POS tags contains adjacent noun and a pronoun it can be a class/sub class and the instance of that particular class/sub class.

Sample Code work through with the rules

E.g.: **Java** is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture.

Word (Token)	Ontology Component
Programming language	Class
Java	Instance of Programming language Class
Developed by	Property
Person	Class
James Gosling -	Instance of Person Class
Organization	Class
Developed in	Property
Sun Microsystems	Instance of Organization Class
Released	Property
Year	Class
1995	Instance of Year Class

E.g.: A **programming language** is an artificial language designed to express computations that can be performed by a machine, particularly a computer. Programming languages can be used to create programs that control the behavior of a machine, to express algorithms precisely, or as a mode of human communication.



Artificial language –class

Programming language – sub class of Artificial language

Express - Property

Computations – class

Machine Computations – sub class of Computations

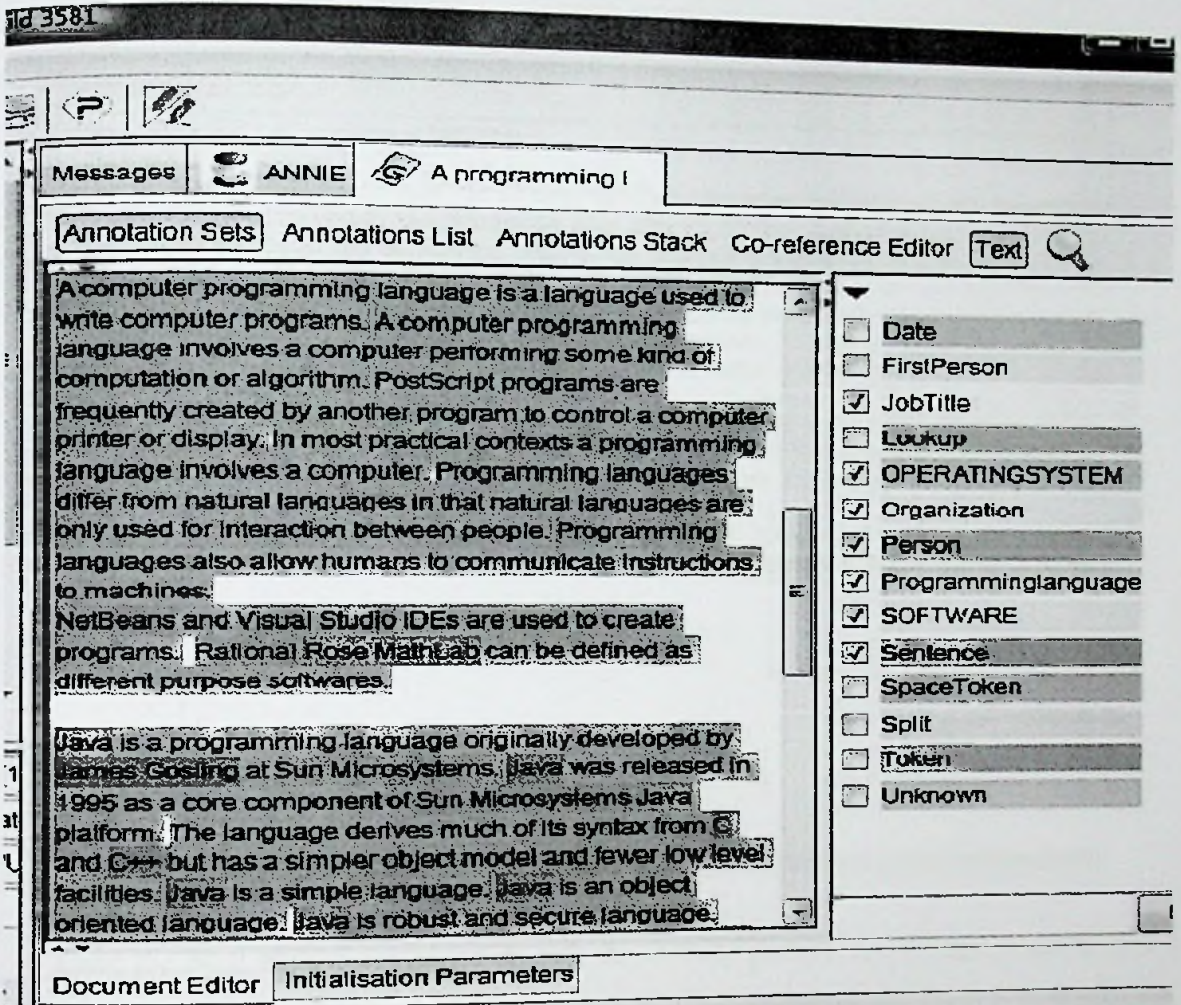
Create – Property // Programming languages create programs

Programs- class

Express- Property // Programming languages express algorithms

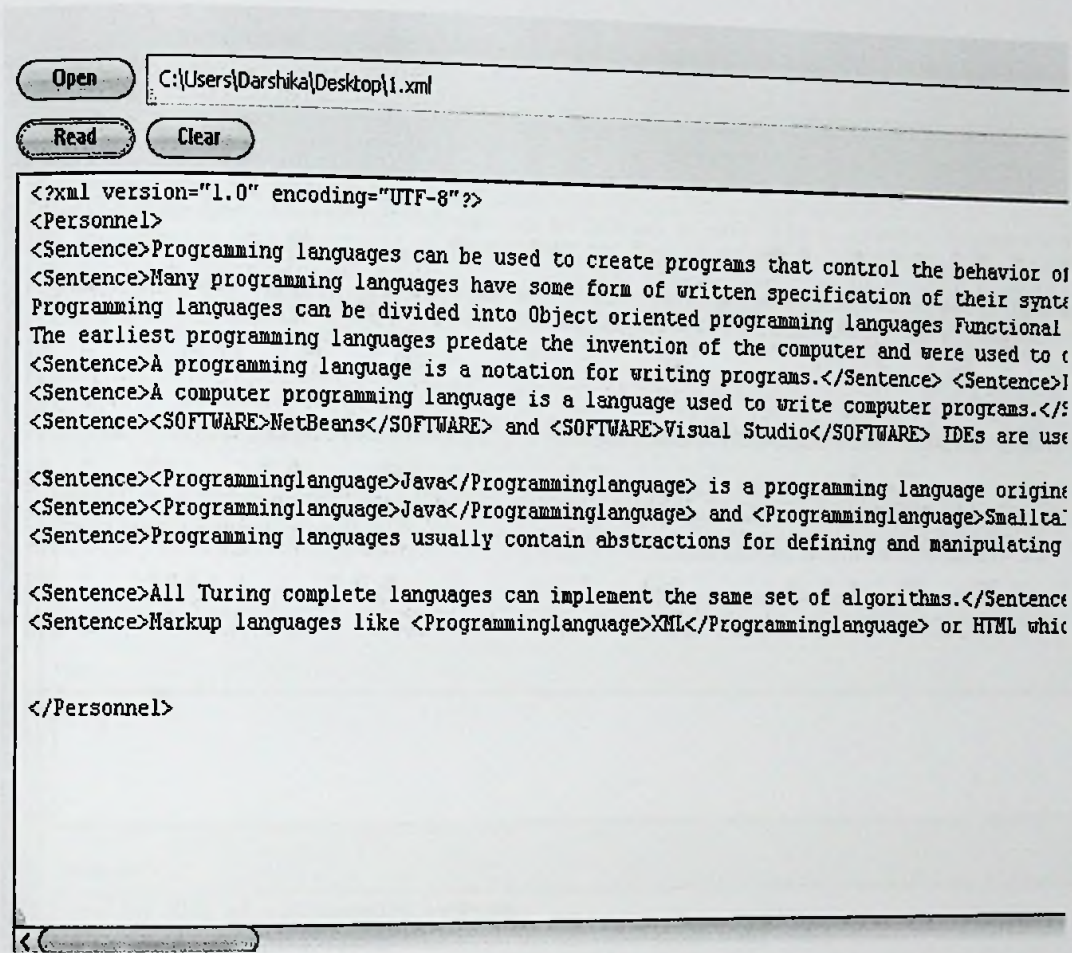
Algorithms – class.

Appendix D User Interfaces of the system



The system is capable of understanding the elements and individuals in a given document separately. The Figure 3.1 shows the annotated output for the uploaded document which contains information about programming languages. The system has special ability to identify the elements such as software, programming languages, Object oriented languages etc...

Ontology scratch of the System



Once the NLP builder completes its task the initially gathered knowledge is stored as the ontology scratch. The Figure describes the ontology scratch of the given Document. The annotated content is displayed with the meaningful semantics in the ontology scratch.

Ontology Scratch Refining screen for Domain Experts

Open C:\Users\Darshika\Desktop\1.xml

Refine

language>Smalltalk</Programminglanguage> can be defined as pure object oriented languages.</Sen
& manipulating data structures or controlling the flow of execution.</Sentence> <Sentence>Pro
lms.</Sentence> <Sentence>ANSI ISO SQL and <Person>Charity</Person> are examples of language
> or HTML which define structured data are not generally considered programming languages.<

Category
Programming Languages

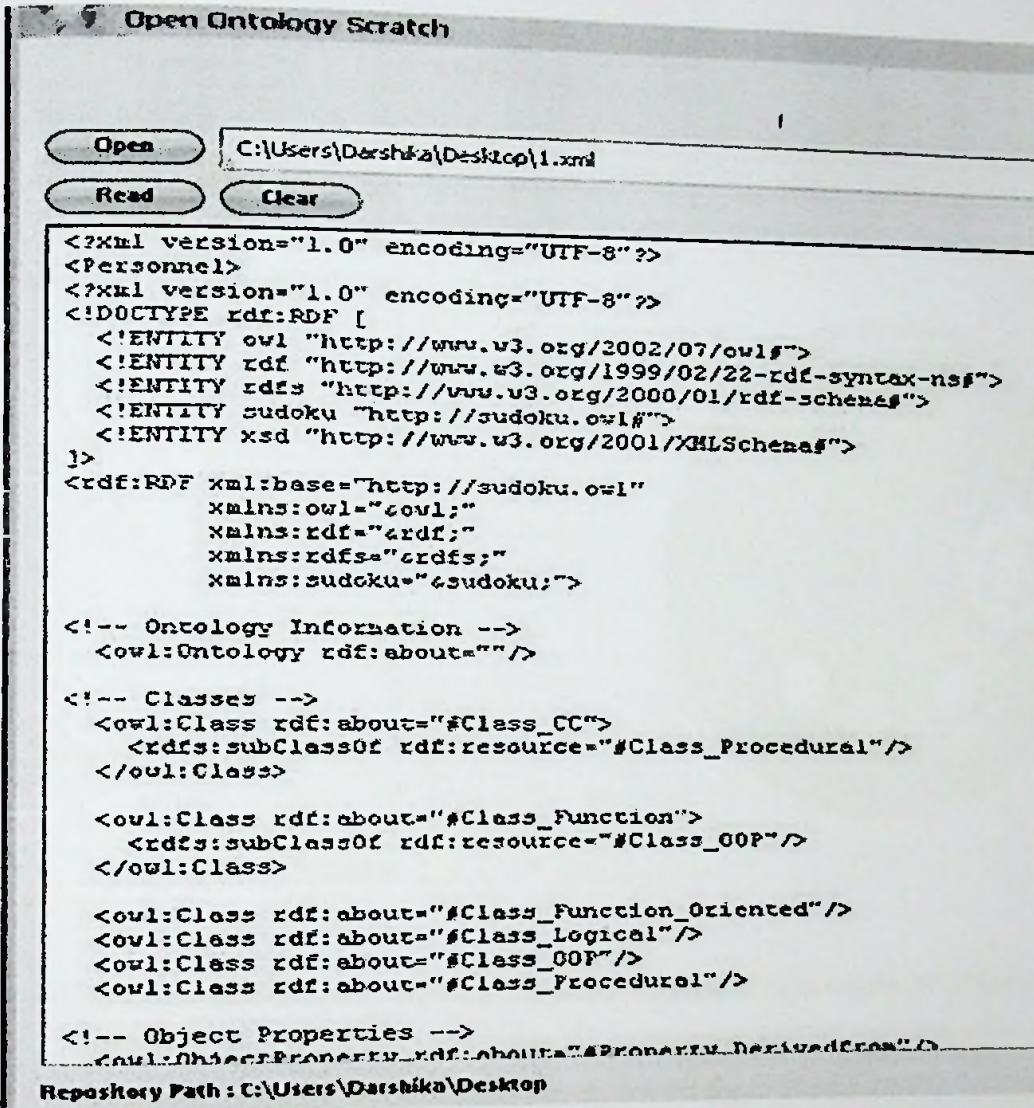
Value
HTML

Comments
Consider HTML as a programming language

Clear Relax Save

The ontology scratch is refined by the domain expert, when there is any inconsistencies occurred. The domain experts are facilitated to correct those mistakes or incorrect annotations by providing a feed back to the initial knowledge.

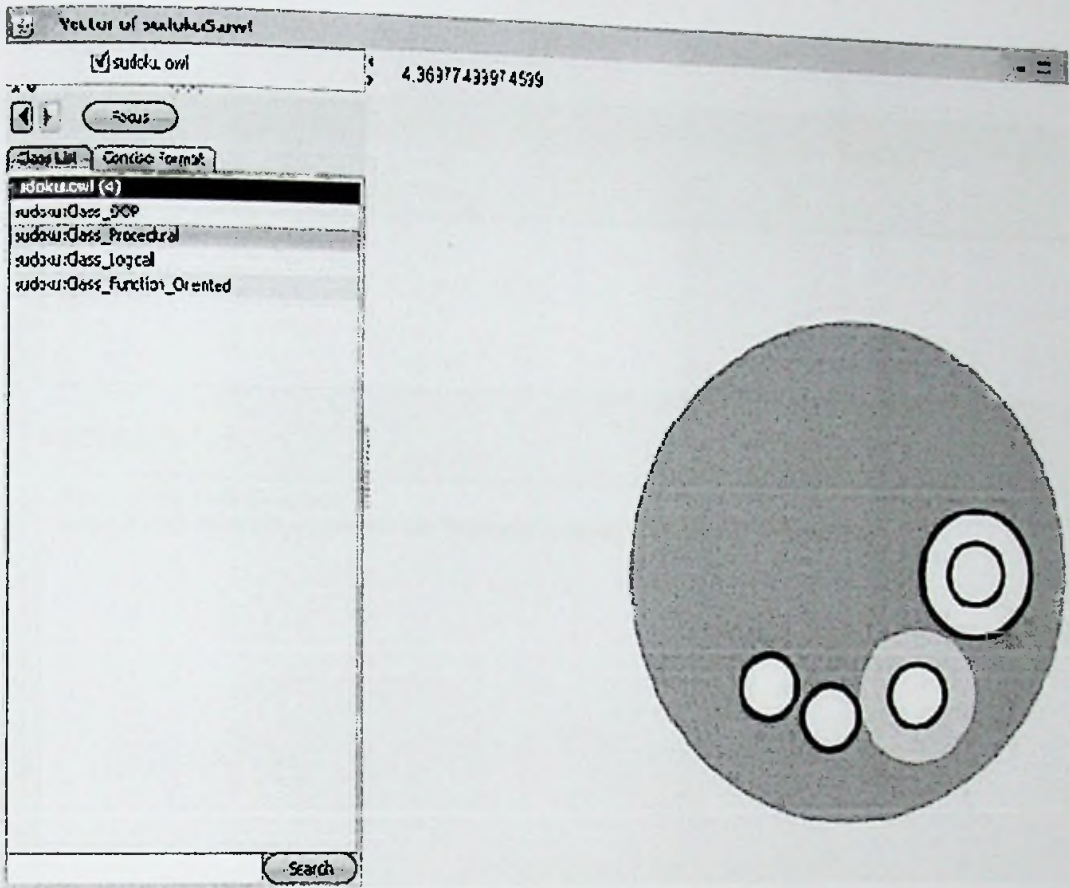
Newly created Ontology



The refined content is parsed in to the ontology editor in order to create the ontology. Once the ontology editor finishes its task, it displays the newly constructed ontology. The Figure illustrates the content of the newly constructed ontology in advance. It has identified the classes, objects and individuals of the given document. In addition to that the editor is capable enough to identify the object properties, data properties etc...



Ontology visualization

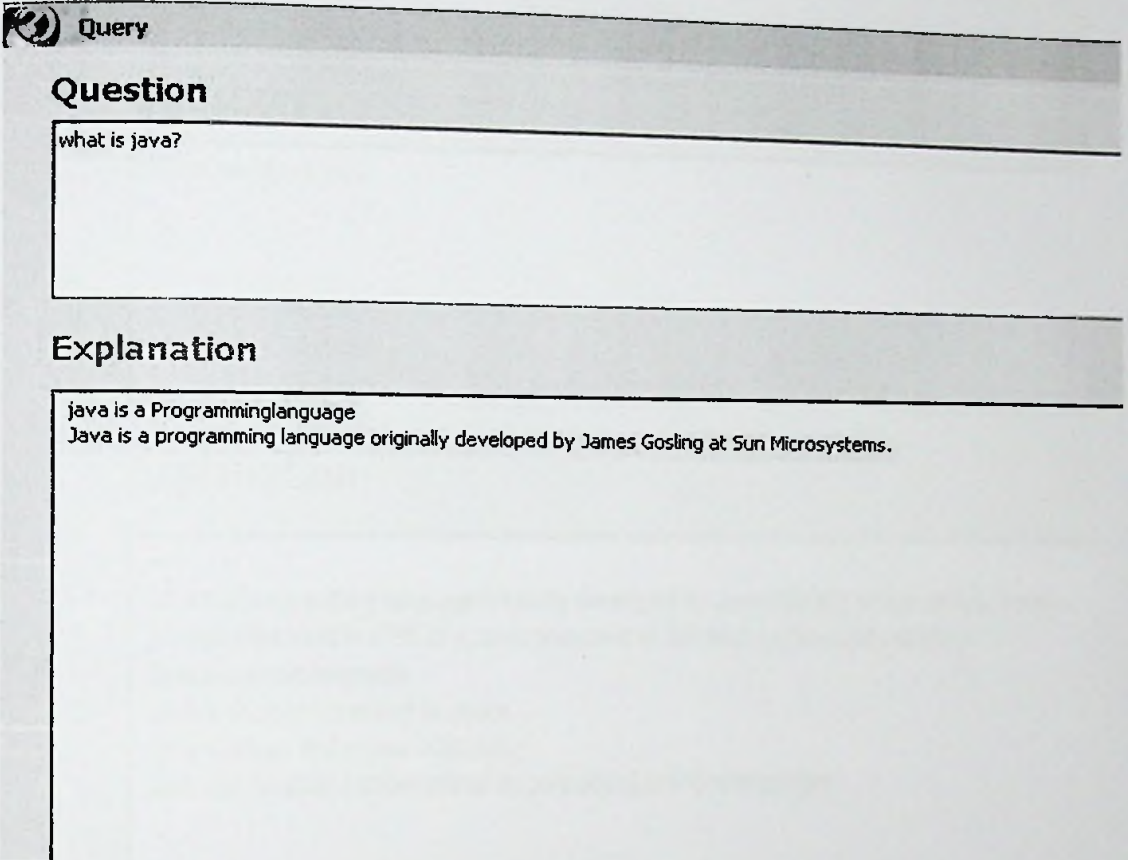


The editor contains an "ontology visualizer" which is responsible of illustrating the constructed ontology. Figure illustrates the constructed ontology visualization in a ven diagram which represents classes sub classes and the depth of each class is included.

With the ontology construction process the system completes the learning process of the system. The user can start the teaching session by asking a natural language question from the system.

Next Figure illustrates the simple question raised by the user and the SESLATA's answer for the given question. The system is focused on providing a direct and simple short answer to the given question.

Answering a simple question



The screenshot shows a user interface for a query system. At the top left, there is a circular icon with a question mark and the word "Query" next to it. Below this, the interface is divided into two main sections: "Question" and "Explanation".

Question

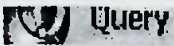
what is java?

Explanation

java is a Programming language
Java is a programming language originally developed by James Gosling at Sun Microsystems.

The system is capable of providing more detailed answer for a questions like "Explain me about java". In this scenario the system provides the detailed explanation for that type of question. Next Figure illustrates the sample question and the answer.

Further explanations



Question

Explain me about java

Explanation

Java is a programming language originally developed by James Gosling at Sun Microsystems. Java was released in 1995 as a core component of Sun Microsystems Java platform. Java is a simple language. Java is an object oriented language. Java is robust and secure language. Java and Smalltalk can be defined as pure object oriented languages.

The Domain experts are given facility to provide their feedbacks to the system's answers in advanced. The feedback can be provided in natural language when the system made a mistake. Figure 3.8 illustrates such scenario where for a question "What is ADA?" and the system's answer was "ADA is a person". Since then the domain expert can provide the correct answer the as ADA is a Programming Language. When second time user raised the same question the system provides the correct answer.

Providing feedback

The image displays two side-by-side screenshots of a query interface. The left screenshot, titled "Query", shows a question "what is ADA?" and an incorrect explanation: "ADA is a Person. Ada FORTRAN ALGOL Python are belongs to different generations of program". The right screenshot, titled "Feedback", shows the same question and a corrected explanation: "ADA is a Programming Language". Below the feedback window are buttons for "Clear Comment" and "Reload". At the bottom right of the entire interface are buttons for "Clear" and "Detailed Expt".

Query

Question

what is ADA?

Explanation

ADA is a Person
Ada FORTRAN ALGOL Python are belongs to different generations of program

Feedback

what is ADA?

ADA is a Programming Language

Clear Comment Reload

Clear Detailed Expt

Figure illustrates the system's respond to the same question with the incorrect answer and the correct answer refined by the domain expert.

Appendix E Results of Evaluation

Question	Human Answer	System Answer	Status
What is Java	A programming language	Java is a program language	Correct
Who developed java?	James Goslin	James Goslin has developed java in sun micro systems	Correct
What is Sun Microsystems	A Company which developed java	Organization	Partially correct
What is ADA	A programming language	Ada is a program language	Correct
Explain me about java	Java is a programming language. It is pure object oriented. Used in the industry.	Java is a programming language developed by James Gosling at sun micro systems.	Insufficient Information.
List Programming languages in the document	Java, Ada, Pascal, C++, C	Java, Ada, Pascal, C++, C	Correct
What is an OO language	A language with OO features.	A programming language	Answer is irrelevant
Who is a developer in programming languages?	One who codes the program?	My knowledge is not sufficient	Incorrect/No answer

Sample ontology file before cross reference with the WorldNet.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY ProgrammingLanguages
"http://ProgrammingLanguages.owl#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF xml:base="http://ProgrammingLanguages.owl"
  xmlns:ProgrammingLanguages="&ProgrammingLanguages;"
  xmlns:owl="&owl;"
  xmlns:rdf="&rdf;">

<!-- Ontology Information -->
  <owl:Ontology rdf:about=""/>

<!-- Classes -->
  <owl:Class rdf:about="#Class_a"/>
  <owl:Class rdf:about="#Class_abstractions"/>
  <owl:Class rdf:about="#Class_ada"/>
  <owl:Class rdf:about="#Class_algol"/>
  <owl:Class rdf:about="#Class_algorithm"/>
  <owl:Class rdf:about="#Class_algorithms"/>
  <owl:Class rdf:about="#Class_an"/>
  <owl:Class rdf:about="#Class_are"/>
  <owl:Class rdf:about="#Class_array"/>
  <owl:Class rdf:about="#Class_as"/>
  <owl:Class rdf:about="#Class_at"/>
  <owl:Class rdf:about="#Class_basic"/>
  <owl:Class rdf:about="#Class_be"/>
  <owl:Class rdf:about="#Class_behavior"/>
  <owl:Class rdf:about="#Class_building"/>
  <owl:Class rdf:about="#Class_c"/>
  <owl:Class rdf:about="#Class_c++"/>
  <owl:Class rdf:about="#Class_can"/>
  <owl:Class rdf:about="#Class_charity"/>
  <owl:Class rdf:about="#Class_code"/>
  <owl:Class rdf:about="#Class_communication"/>
  <owl:Class rdf:about="#Class_compiler"/>
  <owl:Class rdf:about="#Class_component"/>
  <owl:Class rdf:about="#Class_computation"/>
  <owl:Class rdf:about="#Class_computer"/>
  <owl:Class rdf:about="#Class_computers"/>
  <owl:Class rdf:about="#Class_computing"/>
  <owl:Class rdf:about="#Class_contexts"/>
  <owl:Class rdf:about="#Class_control"/>
  <owl:Class rdf:about="#Class_core"/>
  <owl:Class rdf:about="#Class_data"/>
  <owl:Class rdf:about="#Class_database"/>
  <owl:Class rdf:about="#Class_databases"/>
```

```

<owl:
<!-- Object Properties -->
  <owl:ObjectProperty rdf:about="#Property_allow"/>
  <owl:ObjectProperty rdf:about="#Property_been"/>
  <owl:ObjectProperty rdf:about="#Property_belongs"/>
  <owl:ObjectProperty rdf:about="#Property_called"/>
  <owl:ObjectProperty rdf:about="#Property_communicate"/>
  <owl:ObjectProperty rdf:about="#Property_compile"/>
  <owl:ObjectProperty rdf:about="#Property_complete"/>
  <owl:ObjectProperty rdf:about="#Property_connected"/>
  <owl:ObjectProperty rdf:about="#Property_connects"/>
  <owl:ObjectProperty rdf:about="#Property_developed"/>

<!-- Instances -->
  <ProgrammingLanguages:Class_null rdf:about="#Instances_a"/>
  <ProgrammingLanguages:Class_null
rdf:about="#Instances_abstractions"/>
rdf:about="#Instances_behavior"/>
  <ProgrammingLanguages:Class_null
rdf:about="#Instances_building"/>
  <ProgrammingLanguages:Class_null rdf:about="#Instances_c"/>
  <ProgrammingLanguages:Class_null rdf:about="#Instances_c++"/>
  <ProgrammingLanguages:Class_null rdf:about="#Instances_can"/>
  <ProgrammingLanguages:Class_null
rdf:about="#Instances_charity"/>
  <ProgrammingLanguages:Class_null
rdf:about="#Instances_code"/>

  <ProgrammingLanguages:Class_null
rdf:about="#Instances_gosling"/>
  <ProgrammingLanguages:Class_null rdf:about="#Instances_has"/>
  <ProgrammingLanguages:Class_null
rdf:about="#Instances_have"/>
  <ProgrammingLanguages:Class_null
rdf:about="#Instances_html"/>
  <ProgrammingLanguages:Class_null
rdf:about="#Instances_human"/>
  <ProgrammingLanguages:Class_null
rdf:about="#Instances_humans"/>
  <ProgrammingLanguages:Class_null
rdf:about="#Instances_ids"/>
  <ProgrammingLanguages:Class_null
rdf:about="#Instances_imperative"/>
  <ProgrammingLanguages:Class_null
rdf:about="#Instances_implement"/>
</rdf:RDF>

```


LIBRARY / UOM		
20	18	K
20		
20		
20		
20		

