MSD Tool: An Automatically Produced Ontology and UML Diagram for Multi-site Software Development

Hemant H. Patel[a], Dr. Nitesh M. Sureja[b]

[a] Phd Scholar – Rai University-Ahmedabad
[b] Professor-GTU-Ahmedabad
[a]patelhemant.dstc@gmail.com,[b]nmsureja@gmail.com

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 20 April 2021

Abstract: One of the core objectives of the software engineering (SE) industry is to find an advanced level of concept and ways of recycling software to upsurge output and quality. Typically, the object is measured to be one or more technologies or artifacts used in the software lifecycle stage, which can be used to accomplish this area. This article provides a document overview, discussion and analysis, and implements a new solution to the automatic generate ontology, UML diagram, Team Management, Document Management, Team Discussion within a tool. We have selected several software development examples (including software product lines, component development, synthetic programming and model-oriented engineering) to classify and discuss different methods and present ontology tool for comparison with our tool which is proposed in the document. The ontology has been establish to be suitable for providing a common terminology

Keywords: Software Engineering, Ontology, Multi-site Software Development

1. Introduction

Information system assistance is the ability to share, associate and / or discussion information amid multiple agents (personalities, organizations and governments) and / or between multiple agents information sources and widespread information convenience of the end recipient clearly. The main concerns obstructing cooperation in the information system are: independence, distribution, heterogeneity and uncertainty of information causes. Especially we Take care of heterogeneous issues, which can be identified from several levels: system, heterogeneity of syntax, structure and semantics. Ontology provides a knowledge that is expected to solve the problem of semantic heterogeneity. Terms Ontology was proposed by Gruber [1] as a "explicit specification of conceptualization". In this definition, conceptualization refers to an intellectual model of how people usually think about the authenticities of the world; explicit specification mean ideas and associations the abstract model received an explicit name. In general, an ontology is an association consisting of a set of terms related to a information system. Unlike a dictionary, which requires the term and provides a definition for it, ontology is organized intangible knowledge. The ontology accurately expresses a concept, can be interpreted and used by computer systems, but cannot handle dictionary definitions through the computer system. Another difference is through the notion of dependency and assignment speak correctly and formally, they are independent of language either duration. By consenting spread software task groups to right to use public software work facts and appeal semantic allied task evidence, consensus and consistent announcement can be fostered amongst spread software task teams. In the Figure 1.1we have shown the exemplification of software trade ontology by the analyses of awareness and depiction.

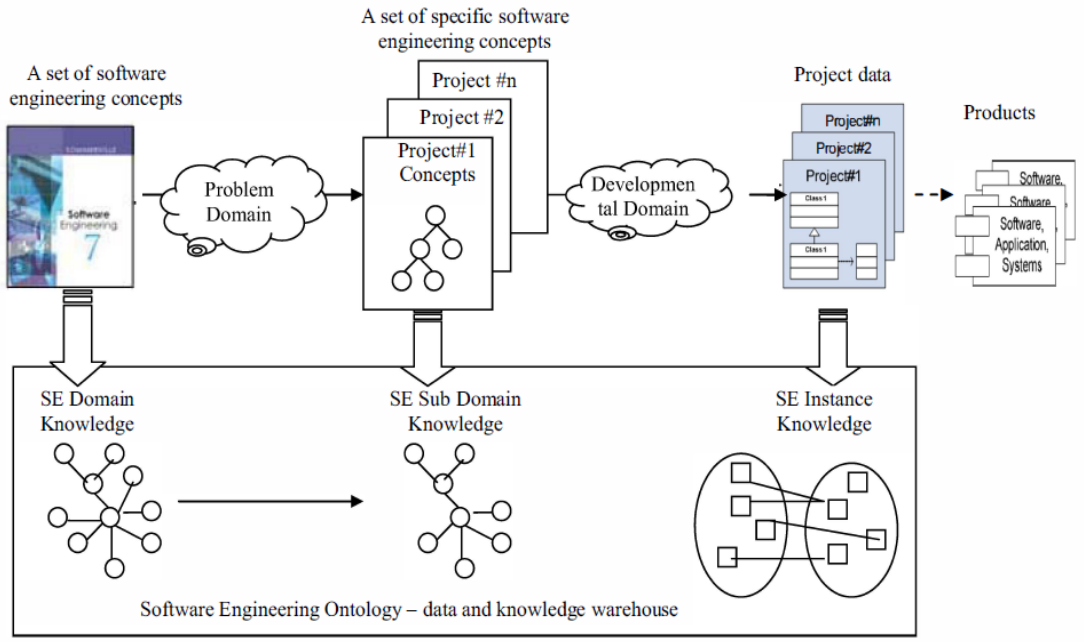

Figure 1.1:- Exemplification of software trade ontology by the analyses of awareness and depiction (Wongthongtham et al. 2009) [2]

Nevertheless, we all know the organization of the ontology is an inactive similar like the other software trade in ontology. When conventional in the ontology and accomplishment at that time it's become an inactive arrangement, its poses two main challenges for knowledge absorption and knowledge dissemination. Absorption is disturbed with the development of catching and it's recognized intangible representations and effort in domain-set awareness, while in altered forms of requests or impart acquaintance to other it turn out to be circulating facts and procedure of in case awareness (Forbes 2013) [3].

2. Literature Survey

In this section, an inclusive survey of numerous existing approaches related to research will be performed to provide a broad enough background and abstracts of relevant literature. They are divided into two categories: ontological-based semantic annotation and ontological-based multi-agent system. The complete precipitate of this fragment is as follows...

- First, we will refer to an altered way for detention of statistics in semantic mark ontology.
- Second, we will publicize ontology-based multi-agent systems and then evaluate these systems.
- Third, which closely evaluates existing methods from an integration opinion of view, including mark ontology as well as more than one agent system in software trends.

2.1 Sementic Mark In Ontology

Knowledge absorption is the process of capturing and demonstrating domain-specific knowledge in a proper theoretical model (Forbes, 2013) [2]. Schwotzer and Berlin (2008) [4] reflect it a process in which new knowledge is apprehended and combined into the knowledge base. When it is essential to capture a large amount of knowledge, it is very significant to acquire knowledge that has been mined through a systematic method without much human resource. The ontological model can be used to theoretically represent acquired knowledge. In the document, a number of studies have projected the use of ontological-based semantic annotations (or semantic annotation for short) to represent the official demonstration of the resource content by connecting associates the content of the resource with ideas defined in the ontology. Ontology is the main portion of application domain explanation and is used as a method for defining annotations related to semantics. Semantic annotation is deployed to create content intelligence and provides many benefits to content-oriented intelligent applications (Yang 2006) [5]. Kiyavitskaya (2006) [6] pointed out that semantic annotation has been extensively used in many altered applications and fields, such as personalization, text summarization, question answering, information filtering, and intelligent knowledge management. In the Semantic Web, semantic annotation tools are used to annotate Web documents so that both humans and machines can understand the Web content. Amardeilh (2009) [7] pointed out that another advantage of semantic annotation is that it can be used to complement ontological filling tasks. In this work, the semantic annotation process takes the following steps: extract the relevant field-

related information from a set of documents, then map it to the concepts identified in the document can learn the domain. The goal is to obtain new instances before filling them in the ontology-bound knowledge base.

Tichy, Koerner and Landhauber (2010) [8] suggested a method to automatically create software models from natural language text with semantic annotations. In (Graubmann and Roshchin, 2006) [9], the author introduced a concept that the semantic model can support automated software components and use a semantic annotation and extension process through technology based on knowledge.

2.1.1 Statistics Valuation in Software Task Base on Semantic Annotation

In the field of software engineering, a large amount of records contains suggestions for the semantic annotation of information about software projects. Much work has donated to the semantic annotation of the source code. Most review methods, however, are based on manual and semi-automatic annotations. Manual methods (such as Qiang, Ming and Zhiguang 2008 [10]; Zygkostiotis, Dranidis, and Kourtesis 2009 [11]) are considered unsuitable because they are bulky, time-consuming, and error-prone, especially when a quantity of large amount of software artifacts are created within. The semi-automatic annotation method (such as Arantes and Falbo 2010 [12]; Panagiotou and Mentzas 2011 [13]) can be a decent solution; however, they still require manual intervention at certain levels of annotation.

Automated methods have been proposed by several works (e.g. Graubmann and Roshchin 2006 [9]; Damljanovic, Amardeilh and Bontcheva 2009 [14]; Tichy, Koerner and Landhauber 2010 [15]; Taglialatela and Taglino 2012 [16]). However, most of them are based on text analysis techniques, so they are only suitable for text artifacts (e.g. software requirements specification, software documentation); they do not fit the semantic annotation of certain types of artifacts (e.g. source code). Also, in the field of software engineering, most of the work considered involves new semantic annotation methods focusing only on semantic annotation to create semantic descriptions of section resources of software. Few pay attention to the filling of the ontology, which is the task of adding new instances of the concept to the ontology (Petasis et al., 2011 [17]). New versions can be derived from semantic annotations.

2.2 Classification of Multi-Agent Task In Ontology

It can be seen from the literature that the combination of ontologies and multi-agent systems (also known as the "ontological-based multi-agent methodology") made many attempts to propagate to consciousness obtained in ontology. In addition, some researchers have mentioned that they are a means to promote knowledge assimilation by capturing knowledge and integrating it into the ontology knowledge base. These jobs cover many fields, including software engineering, medical, and education.

MAEST (Maamri and Sahnoun, 2007) [18] is a multi-agent system designed to assist testers in the assessment process. Software testing ontology is developed to model various aspects related to testing software systems, such as testing operations, testing methods, software artifacts, information on test environment, available resources and requirements for test results. Agents use this information as a way to share knowledge and foster consistent statement.

Lee and Wang (2009) [19] announced an ontological-based computational intelligent multi-agent to conduct the Capacity Maturity Model Integration (CMMI) assessment. The system consists of three main actors interacting with each other to achieve the goal of effectively synthesizing software engineering process evaluation reports related to CMMI evaluation. The CMMI ontology is specially established based on basic knowledge of CMMI process and invention quality assurance (PPQA). The software agent uses the concepts identified in the ontology to extract key statements from the assessment report, so that the team members involved can understand it easily and quickly.

The OntoDiSEN ontology (Chaves et al., 2011) [20] has been developed to represent contextual information in a global software development environment. Software agents use this ontology to recover and infer information in context. In addition, the author claims that the planned ontological agent could deploy ontological knowledge, such as updating contextual information or inserting new inference actions and events. However, no particulars information's are provided to indicate how ontology agents perform these tasks.

Dolia (2010) [21] proposes an ontological-based multi-agent system to provide useful information about educational institutions, such as course statistics, course process and plans. Ontology Academy Academic developed to identify ideas and associations that exist in the university education environment. Agents use this ontology to enhance their understanding of reliable communication and provide response for different types of queries.

In (Parhi, Pattanayak and Patra 2015) [22] the authors improve an ontology-based multi-agent system to determine suitable cloud services as requested by consumers. The system involves of three agents collaboratively

working to provide active searching for a cloud service. The Cloud Service Ontology is established to signify cloud service description. The agents procedure this ontology for intellectual about the services and for information retrieval.

2.3 Marge Vision And Valuation For Ontology Assessment

In this segment, existing systems and methods implemented in the document are evaluated and key issues to be addressed in order to design a framework that allows the working software engineering ontology to be identified for concentration. This section outlines all the issues.

The foremost limitations of existing systems and methods relate to three areas.

- Absence of effective methods to dynamically capture knowledge about software project information.
- Absence of effective management of the knowledge gained in the ontology.
- Absence of active platform can be used in multi-site software development environment.

2.3.1 Absence of Effective Methods

As mentioned in above Section, in the software engineering documentation, the semantic annotations were used to capture software project information to perform some knowledge assimilation work. Certain of them are still created on manual semantic annotation approaches (eg Qiang, Ming and Zhiguang 2008 [10]; Zygkostiotis, Dranidis and Kourtesis 2009 [11). The manual technique has many disadvantages like: they are cumbersome, time consuming, lying to errors and labor intensive. A number of works have attempted to overawed these complications by applying semi-automatic semantic annotation approaches (eg, Arantes and Falbo 2010 [12]; Panagiotou and Mentzas 2011 [23]). On the other hand, they still need additional guide involvement. A moment ago, the focus has turned to an automated technique to capture the semantics of software project information, which is well-organized and involves nominal or no effort from software team members. Nevertheless, most of the reviewed approaches (eg Graubmann and Roshchin 2006 [9]; Damljanovic, Amardeilh and Bontcheva 2009 [14]; Taglialatela and Taglino 2012 [16]) are created on textual analysis. They are appropriate for software relics that enclose textual explanations (such as software documentation, software requirements specification). Though, they are not appropriate for taking knowledge of certain types of artifacts (such as source code).

In addition, the capture output of these approaches is mostly in RDF (Resource Description Framework) and RDFS (Resource Description Framework Schema). RDF is specifically designed at relating the semantics of information in a machine-understandable and machine-computable form. RDFS encompasses RDF with the schema terminology (such as Class, subclass Of, Property, domain, range). Less research has been done to populate the ontology library in the OWL (Web ontology Language), which is an allowance of RDF and RDFS. Although RDF and RDFS are very valuable for relating resources using simple semantics that covering objects and their relationships, they still have certain boundaries. For example, it doesn't provide the bridging, inverse, or symmetry properties that OWL can. Because OWL is so communicative and can formally determine relationships between classes based on description logic, it is more beneficial to capture and store knowledge in OWL. It allows to term and conclude the properties of software resources from the knowledge base.

2.3.2 Absence of Effective Management of the Knowledge Gained In the Ontology

In the document, some researchers have suggested using agent-based technology along with ontology for the resolve of absorbing knowledge and disseminating knowledge. Most tested ontology-based multi-agent systems use the ontology to assist the software agent in the following tasks:

- Application representation and domain knowledge
- Find and retrieve information
- Knowledge of reasoning
- Promote communication and interoperability of agents
- Promote semantic annotation

Although some works (Monte-Alto et al., 2012 [24]; Teixeira and Huzita, 2014 [20]) privilege to control the development of knowledge acquired by software agents in ontology, as far as they are concerned. I know, they don't explicitly mention how agents can manipulate this knowledge. The clarification is just abstract. As point out earlier, knowledge in software development projects is continuously changing. Therefore, there is an essential for a technique that can successfully manage the development of knowledge assimilated in software engineering ontology.

2.3.3 Absence of Active Platform Can Be Used In Multi-Site Software Development

Software development is measured to be a difficult, shared, knowledge-intensive movement. The assets of a software product be determined by mostly on the excellence of the software process, which is the outcome of the accomplishments performed throughout the complete software development process. An active software process is tied to persons, outfits, and operating measures in common (Paulk 2002) [25]. Consequently, a lot of research has been paid attention to evolving software applications or tools that can support project team members capably carry out their tasks. Even if the project team can assistance from tools exactly designed for a software development action, supplementary combination tools that can be used for associated activities can be benefit more. Ossher, Harrison and Tarr (2000) [26] show that important desires that support combination of software development activities in each phase of the software development life cycle denotes the origin of the software engineering environment. . They describe Software Engineering Environment (SEEs) as "an assimilated set of software applications that ambition software engineering activities throughout the software lifecycle."

As mentioned in earlier Section, in the existing works, all qualified backend systems are designed to deliver knowledge at the end of development teams for support activities only in specific software development dynamics or specific stages in the software development lifecycle. In certain, they concentration on software deployment and conservation tasks. On the other hand, software development actions and their artifacts are correlated. Alterations in one job or movement can disturb work in other actions. As a result, it is essential to care software development team members in their several events. Especially in a multi-site software development environment where team members are geographically circulated, insufficient communication and synchronization are key factors delaying the success of projects or software. It is very significant to have a backend platform for a multi-site software development environment that can help faraway team member's work organized during altered segments of the software development lifecycle. (Sengupta, Chandra and Sinha 2006) [27].

In addition, several of the systems measured were unable to function actively. This means that they mainly rely on certain team members' determinations to obtain knowledge. Often, team members may not understand the actuality of useful knowledge due to the huge volume of information or since they are new members of the project. The absence of an operational platform that can successfully achieve knowledge and share knowledge to convey the correct information to the exact people at the accurate time is a concern that this research still desires to talk.

3. Methodology

In this research, the focus in on the automatic evolution of ontology, UML diagram, document control, team management and team discussion. In Figure 4.1 we have describe the flow of how to generate an ontology and UML diagram with help of CoreNLP fundamental and Graphical open source library.

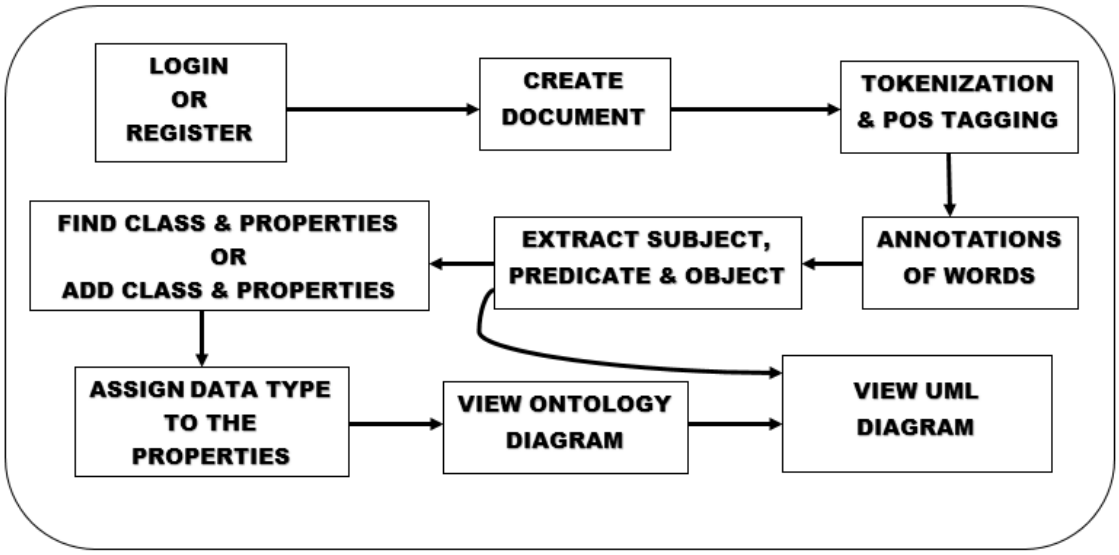

Figure 3.1: - MSD Flow Diagram

- **Login**: - In the login phase user has create their account. User also maintain and share their document, discuss with other user, see their past record, give rights to another user if required.

- **Create Document**: - With the help of login credential user also create a new project / document or open their existing project / document.

- **Tokenization & POS Tagging**: - Divide phrases, sentences, paragraphs, or whole text documents into smaller units, such as separate words or terms. Each of these smaller units is acknowledged as a token. The process of converting meaningful data (such as sentences) into random strings is known as tokens. These tags are used as references to the original data. Then we have identifying correct part-of-speech of a word. A Part-Of-Speech Tagging assign a Tag to each words in the document. POS Tagging reads and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc. With the help of POS Tagging process we have collect a list of nouns from the documents or paragraph.

- **Annotation of Words**: - It refer that, we have given annotation to each word which is collected in POS Tagging phase. In this phase we have collect a list of nouns base on that we have select a key words or annotation which can help us to identify Class, Subclass and Properties.

- **Extract Subject, Predicate and Object**: - In this phase, we take a help of NLP to find out Subject, Predicate and Object from each sentences using Dependency Parsing. As we know a sentences are made up of different parts of grammar. There is a grammatical dependencies between each words in a sentence. For Example "Teacher is a subtype of the Person." The Dependencies can be said as follows: 1) "Teacher" is the subject of this sentence (The one we are talking about). 2) "Subtype" is the predicate (This the is the relation we are talking about). 3) "Person" is the object (To whom something is pointed to be). NLP provides us a grammatical Tress representation of the sentence, based on the categories according to the TreebankLanguagePack. Using this Tree we can extract the subject, Predicate and Object from the sentences.

- **Find Class & Properties Or Add Class & Properties**: - Base on the Annotation of Words phase and Extract Subject, Predicate and Object phase we have automatically find out Class and its Properties from the documents / paragraphs. If we have required to add some class and its properties at run time so in this phase we can do it manually.

- **Assign Data Type To The Properties: -** In this phase we have manually apply a data type (Integer, Float, Double etc.) to the properties.

- **View Ontology Diagram**: - We have find the Class, Subclass and Properties and their data type in above phase. In this phase we have seen the graphical view of ontology diagram base on the Class and its properties.

- **View UML Diagram**: - Base on the Extract Subject, Predicate and Object phase and Find Class and Properties phase we have find the dependency to each class and its properties. So it's help to draw the Class Diagram and ER-Diagram in this phase**.**

4. Results and discussion

We will endeavour to provide basic informative or overview for our Multisite software development tool. We will also provide a brief description of each phase of the tool. Below figure 4.1 show the complete dependency list of Subject, Predicate and Object for the created document.

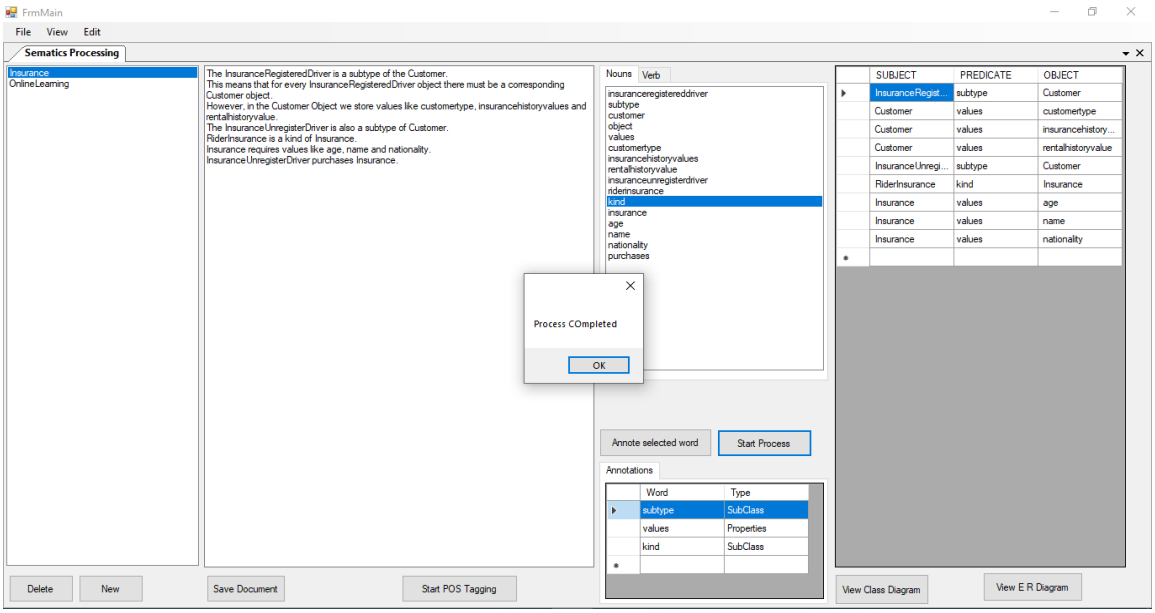

Figure 4.1: - List of Subject, Predicate and Object

In below Figure 4.2 show the class hierarchy base on the annotation of word and their relationship.

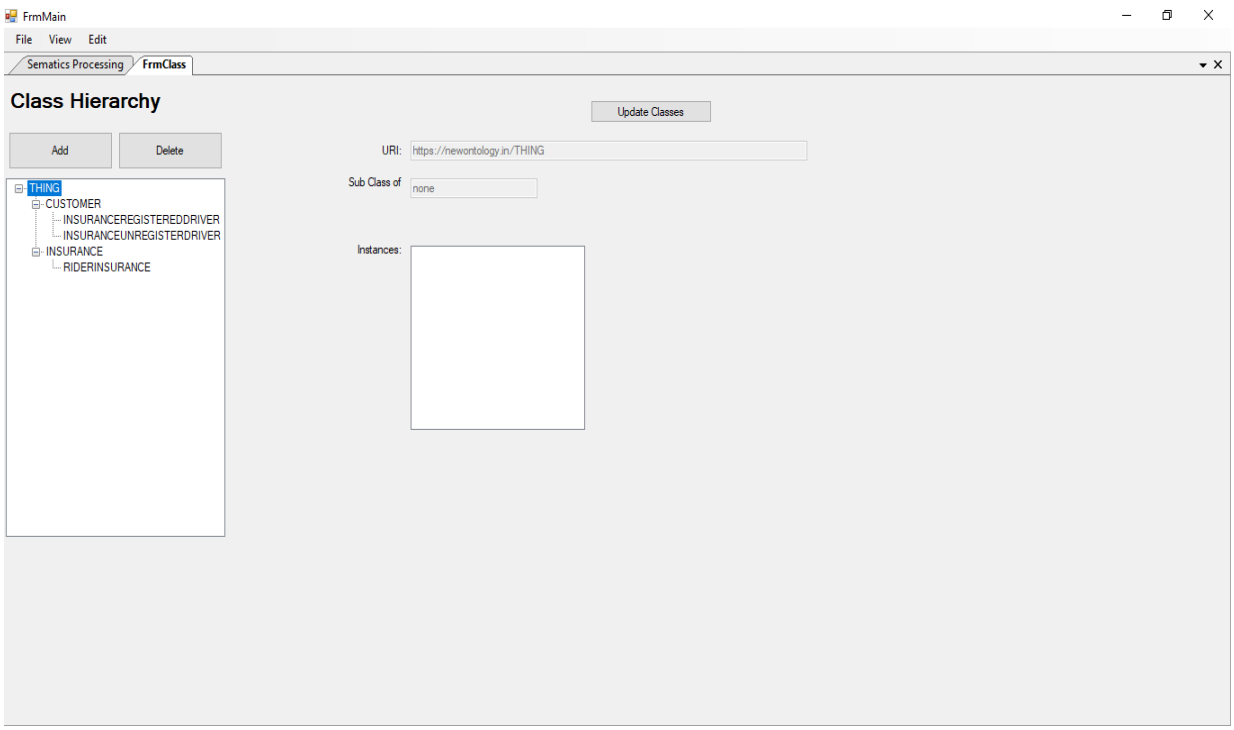

Figure 4.2: - Class Hierarchy

In below Figure 4.3 show the property hierarchy and data type of that property. We have fetch property base on the annotation of word and their relationship.

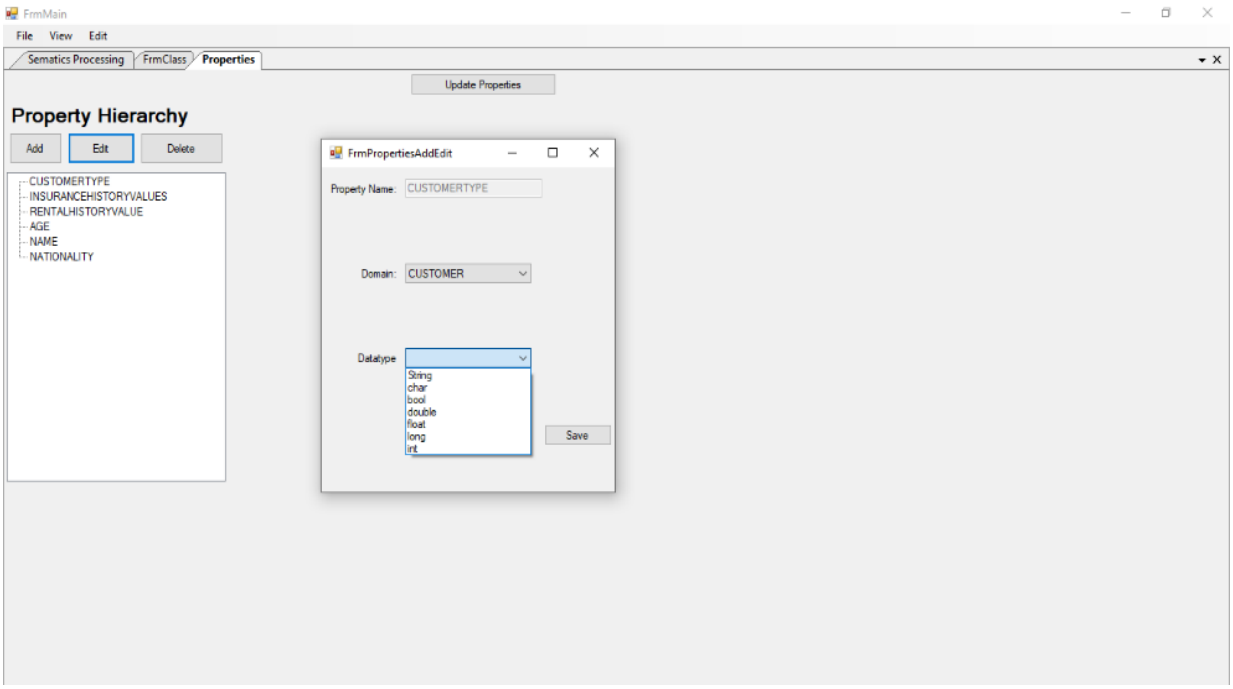

Figure 4.3: - Property Hierarchy and their Data type

In below Figure 4.4 show the ontology diagram base on the class hierarchy, property hierarchy and data type of the property.

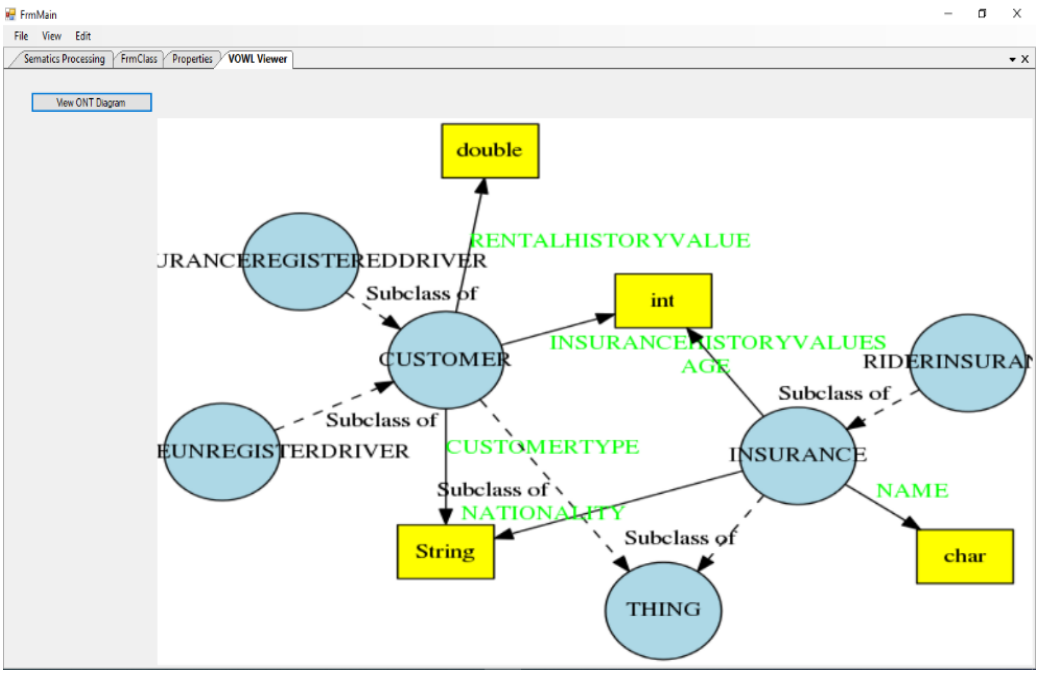

Figure 4.4: - Ontology Diagram

In below Figure 4.5 show a class diagram base on the selected annotation word and their relationship.

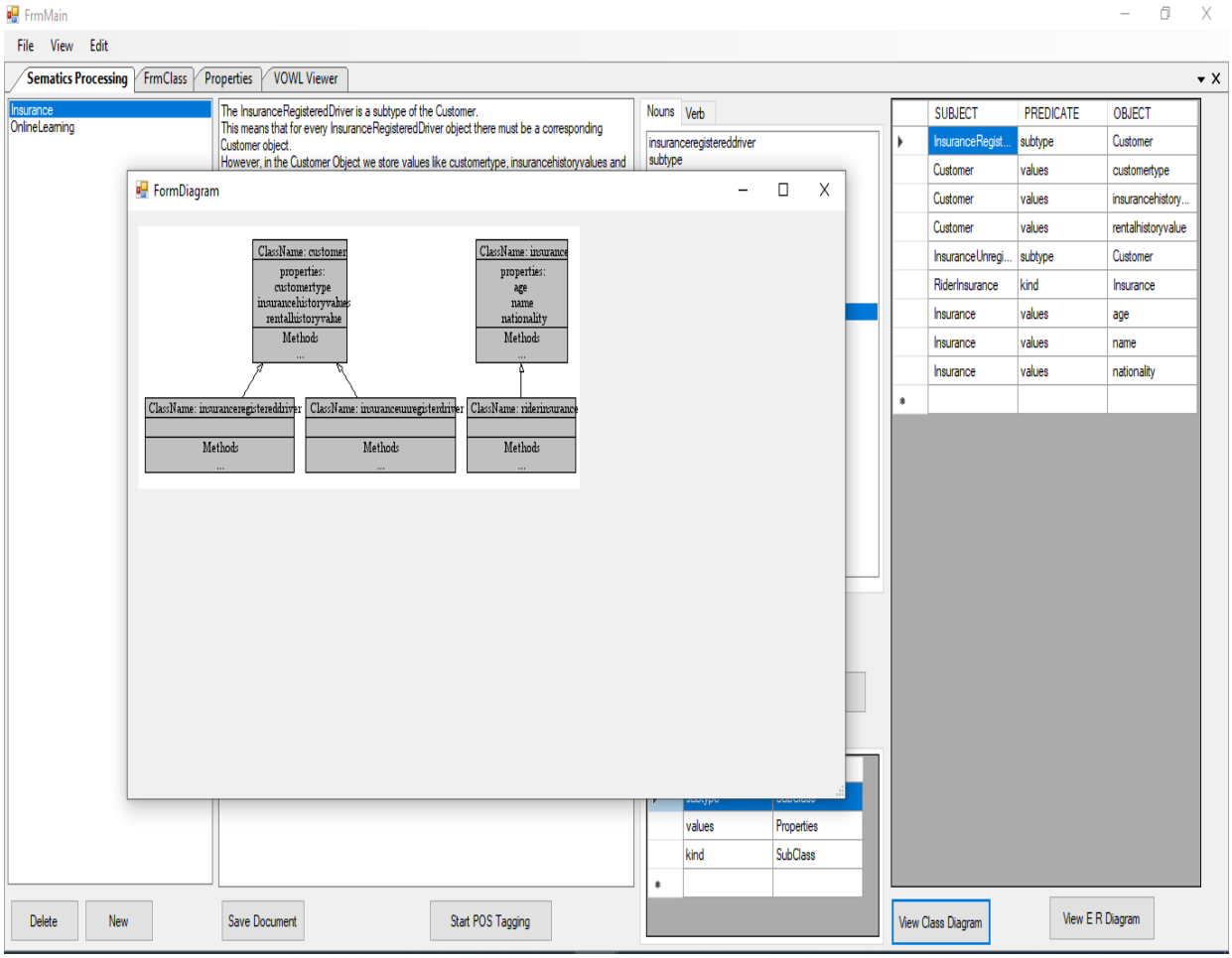

Figure 4.5: - Class Diagram

In below Figure 4.6 show an ER diagram base on the selected annotation word and their relationship.

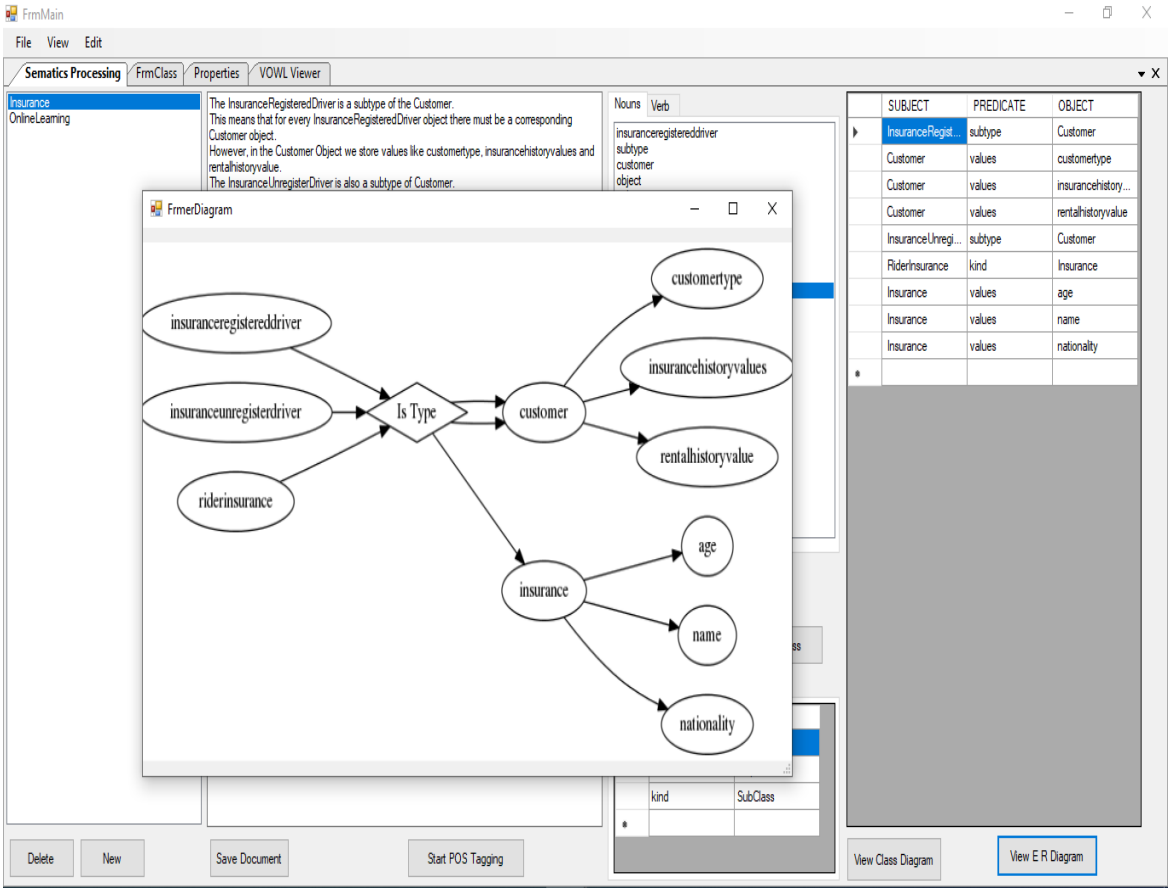

Figure 4.6: - ER Diagram

In above section we have seen in which way our Multisite Software Development tool is work. Now we have seen the comparison of MSD tool to different available tools. The analysis is done on the basis of...

- Architecture
- Storage
- GUI Design
- Import / Export Files
- Team Management
- Document Management etc…

Features	**Protege 5.2.0**	**Swoop 2.3 Beta 4**	**Odase**	**OwlGred 1.6.1**	**Apollo**	**MSD**
Semantic web Architecture	Web based, Client/server	Client/server , Web based	Web based	Desktop & Web based	Standalone	**.Net Client-server and**
Implemented in	Java	Java	SWRL	Java	Java	**.Net**
Import / Export format	XML, RDF, Owl, HTML, UML	OWL, XML, RDF, text formats, OIL+DAML	OWL, SWRL, RDF	OWL, OWL2, UML, RDF/XML	OCML, CLOS, META, RDF, XML	**RDF**
Ontology Storage	Files, DBMS	HTML models	Files	Files	Files	**DBMS**
Graphical taxonomy	Yes	Yes	Yes	Yes	Yes	**Yes**
Graphical prunes (views)	Yes	Yes	Yes	Yes	Yes	**Yes**

Zooms	Yes	No	No	No	No	**Yes**
Multi-Team Management	No	No	No	No	No	**Yes**
Document Management	No	No	No	No	No	**Yes**
Automatically Fetch Class From the Document	No	No	No	No	No	**Yes**
Automatically Fetch Property From the Document	No	No	No	No	No	**Yes**
Support UML Diagram base on document	No	No	No	No	No	**Yes**

Table 4.1:- Comparison of MSD tool with other tool

5. Conclusion

Ontologies is an identical significant in numerous scientific fields such as: knowledge engineering and representation, information retrieval and extraction, knowledge management, mediator systems, etc. For that we can say ontology is the strength of the Semantic Web. The ability to create an ontology for any natural language gives us a prospect to use information that can be processed by humans and computers in a natural way. For the improvement of the ontology, operative tools are the essential requirement. Providentially, software tools at present be used to complete most of the processes necessary for ontological development, allowing us to focus on the advanced needs of ontological development.

In MSD tool user can automatically fetch the class and property hierarchy base on the annotation word and their relationship. As well as user can manually enter a class and their property if it's required. Base on the class and property hierarchy user can see the ontology diagram. User can also see an UML diagram base on the annotation word and their relationship. User can also create a multiple document in one project and give their rights to the multiple user and that particular project that team can discuss on our tool.

References

1. Gruber, T. R. A Translation Approach to Portable Ontology Specications. Knowledge Acquisition 5, 2 (1993), 199-220.
2. Wongthongtham, P., E. Chang, T.S. Dillon, and I. Sommerville. 2009. "Development of a software engineering ontology for multi-site software development." IEEE Transactions on Knowledge and Data Engineering 21 (8): 1205-1217. doi: 10.1109/TKDE.2008.209.
3. Forbes, David E. 2013. "A Framework for Assistive Communications Technology in Cross-Cultural Healthcare." School of Information Systems, Curtin Business School, Curtin University.
4. Schwotzer, Thomas, and FHTW Berlin. 2008. "Building context aware P2P systems with the shark framework." In Fourth International Conference on Topic Maps Research and Applications, Leipzig, Germany, 157-168.
5. Yang, Kun. 2006. "A Conceptual Framework for Semantic Web-based E-Commerce." Department d'informatique et de genie logiciel, University of Laval.
6. Kiyavitskaya, Nadzeya. 2006. "Tool Support for Semantic Annotation." International Doctorate School in Information and Communication Technologies DIT - University of Trento
7. Amardeilh, Florence. 2009. "Semantic annotation and ontology population." In Semantic Web engineering in the Knowledge Society, 135-160. IGI Global.
8. Tichy, Walter F., Sven J. Koerner, and Mathias Landhaußer. 2010. "Creating software models with semantic annotation." In Proceedings of the Third Workshop on Exploiting Semantic Annotations in Information Retrieval, Toronto, ON, Canada, 17-18. 1871973: ACM. doi: 10.1145/1871962.1871973.
9. Graubmann, P., and M. Roshchin. 2006. "Semantic annotation of software components." In Software Engineering and Advanced Applications, 2006. SEAA '06. 32nd EUROMICRO Conference on, Aug. 29 2006-Sept. 1 2006. 46-53. doi: 10.1109/euromicro.2006.54.
10. Qiang, Lu, Chen Ming, and Wang Zhiguang. 2008. "A semantic annotation based software knowledges sharing space" Network and Parallel Computing, 2008. NPC 2008. IFIP International Conference on, doi: 10.1109/npc.2008.55.

11. Zygkostiotis, Zinon, Dimitris Dranidis, and Dimitrios Kourtesis. 2009. "Semantic annotation, publication, and discovery of Java software components: an integrated approach" The 2nd Workshop on Artificial Intelligence Techniques in Software Engineering (AISEW 2009), Thessaloniki, Greece: CEUR-WS. org.
12. Arantes, L. d. O., and R. d. A. Falbo. 2010. "An infrastructure for managing semantic documents" 2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops, Vitoria, Brazil, doi: 10.1109/EDOCW.2010.17.
13. Panagiotou, Dimitris, and Gregoris Mentzas. 2011. "Leveraging software reuse with knowledge management in software development." International Journal of Software Engineering and Knowledge Engineering 21 (05): 693-723.
14. Damljanovic, Danica, Florence Amardeilh, and Kalina Bontcheva. 2009. "CA manager framework: creating customised workflows for ontology population and semantic annotation." In Proceedings of the Fifth International Conference on Knowledge Capture, Redondo Beach, California, USA, 177-178. 1597770: ACM. doi: 10.1145/1597735.1597770.
15. Taglialatela, Andrea, and Francesco Taglino. 2012. "A semantics-based approach to software reuse" The Fifth Interop-Vlab.It Workshop on Complexity of Systems, Complexity of Interoperability in conjunction with it AIS 2012., Rome, Italy.
16. Petasis, Georgios, Vangelis Karkaletsis, Georgios Paliouras, Anastasia Krithara, and Elias Zavitsanos. 2011. "Ontology population and enrichment: State of the art." In Knowledge-Driven Multimedia Information Extraction and Ontology Evolution: Bridging the Semantic Gap, eds Georgios Paliouras, Constantine D. Spyropoulos and George Tsatsaronis, 134-166. Berlin, Heidelberg: Springer Berlin Heidelberg.
17. Maamri, Ramdane, and Zaidi Sahnoun. 2007. "MAEST: multi-agent environment for software testing." Journal of Computer Science 3 (4): 249-258.
18. Lee, Chang-Shing, and Mei-Hui Wang. 2009. "Ontology-based computational intelligent multi-agent and its application to CMMI assessment." Applied Intelligence 30 (3): 203-219. DOI: 10.1007/s10489-007-0071-1.
19. Chaves, A.P., I. Steinmacher, L. Leal, G. Camila, E.H.M. Huzita, and A.B. Biasão. 2011. "OntoDiSENv1: An ontology to support global software development." CLEI Electronic Journal 14 (2): 2-2.
20. Dolia, Prashant M. 2010. "Integrating ontologies into multi-agent systems engineering (MaSE) for university teaching environment." Journal of Emerging Technologies in Web Intelligence 2 (1): 42-47.
21. Parhi, Manoranjan, Binod Kumar Pattanayak, and Manas Ranjan Patra. 2015. "A multi-agent-based framework for cloud service description and discovery using ontology." In Intelligent Computing, Communication and Devices: Proceedings of ICCD 2014, Volume 1, eds C. Lakhmi Jain, Srikanta Patnaik and Nikhil Ichalkaranje, 337-348. New Delhi: Springer India.
22. Panagiotou, Dimitris, Fotis Paraskevopoulos, and Gregoris Mentzas. 2011. "Knowledge-Based Interaction in Software Development." Intelligent Decision Technologies 5 (2): 163-175.
23. Teixeira, Lucas O., and Elisa H. M. Huzita. 2014. "DiSEN-AlocaHR: A multi-agent mechanism for human resources allocation in a distributed software development environment." In Distributed Computing and Artificial Intelligence, 11th International Conference, eds Sigeru Omatu, Hugues Bersini, M. Juan Corchado, Sara Rodríguez, Paweł Pawlewski and Edgardo Bucciarelli, 227-234. Cham: Springer International Publishing.
24. Paulk, Mark. 2002. "Capability maturity model for software." In Encyclopedia of Software Engineering. John Wiley & Sons, Inc.
25. Ossher, Harold, William Harrison, and Peri Tarr. 2000. "Software engineering tools and environments: A roadmap." In Proceedings of the Conference on The Future of Software Engineering, Limerick, Ireland, 261-277. 336569: ACM. doi: 10.1145/336512.336569.
26. Sengupta, Bikram, Satish Chandra, and Vibha Sinha. 2006. "A research agenda for distributed software development." In Proceedings of the 28th International Conference on Software Engineering, Shanghai, China, 731-740. 1134402: ACM. doi: 10.1145/1134285.1134402