Extracting Code Resource from OWL by Matching Method Signatures using UML Design Document

UML Extractor

Gopinath Ganapathy [1]
[1]Department of Computer Science
Bharathidasan University
Trichy, India.
gganapathy@gmail.com

S. Sagayaraj [2]
[2] Department of Computer Science
Sacred Heart College
Tirupattur, India
sagi_sara@yahoo.com

***Abstract*—Software companies develop projects in various domains, but hardly archive the programs for future use. The method signatures are stored in the OWL and the source code components are stored in HDFS. The OWL minimizes the software development cost considerably. The design phase generates many artifacts. One such artifact is the UML class diagram for the project that consists of classes, methods, attributes, relations etc., as metadata. Methods needed for the project can be extracted from this OWL using UML metadata. The UML class diagram is given as input and the metadata about the method is extracted. The method signature is searched in OWL for the similar method prototypes and the appropriate code components will be extracted from the HDFS and reused in a project. By doing this process the time, manpower system resources and cost will be reduced in Software development.**

Keywords- Component: Unified Modeling language, XML, XMI Metadata Interchange, Metadata, Web Ontology Language, Jena framework.

I. INTRODUCTION

The World Wide Web has changed the way people communicate with each other. The term Semantic Web comprises techniques that dramatically improve the current web and its use. Today's Web content is huge and not well-suited for human consumption. The machine processable Web is called the Semantic Web. Semantic Web will not be a new global information highway parallel to the existing World Wide Web; instead it will gradually evolve out of the existing Web [1]. Ontologies are built in order to represent generic knowledge about a target world [2]. In the semantic web, ontologies can be used to encode meaning into a web page, which will enable the intelligent agents to understand the contents of the web page. Ontologies increase the efficiency and consistency of describing resources, by enabling more sophisticated functionalities in development of knowledge management and information retrieval applications. From the knowledge management perspective, the current technology suffers in searching, extracting, maintaining and viewing information. The aim of the Semantic Web is to allow much more advanced knowledge management system.

To develop such a knowledge management system the software company's can make use of the already developed coding. That is to develop new software projects with reusable codes. The concept of reuse is not a new one. It is however relatively new to the software profession. Every Engineering discipline from Mechanical, Industrial, Hydraulic, Electrical, etc, understands the concept of reuse. However, Software Engineers often feel the need to be creative and like to design "one time use" components. The fact is they come with unique solution for every problem. Reuse is a process, an applied concept and a paradigm shift for most people. There are many definitions for reuse. In plain and simple words, reuse is*, "The process of creating new software systems from existing software assets rather then building new ones".*

Systematic reuse of previously written code is a way to increase software development productivity as well as the quality of the software [3, 4, 5]. Reuse of software has been cited as the most effective means for improvement of productivity in software development projects [6, 7]. Many artifacts can be reused including; code, documentation, standards, test cases, objects, components and design models. Few organizations argue the benefits of reuse. These benefits certainly will vary organization to organization and to a degree in economic rational. Some general reusability guidelines, which are quite often similar to general software quality guidelines, include [8] ease of understanding, functional completeness, reliability, good error and exception handling, information hiding, high cohesion and low coupling, portability and modularity. Reuse could provide improved profitability, higher productivity and quality, reduced project costs, quicker time to market and a better use of resources. The challenge is to quantify these benefits.

For every new project Software teams design new components and code by employing new developers. If the company archives the completed code and components, they can be used with no further testing unlike open source code and components. This has a recursive effect on the time of development, testing, deployment and developers. So there is a base necessity to create system that will minimize these factors.

Code re-usability is the only solution for this problem. This will reduce the development of an existing work and testing. As the developed code has undergone the rigorous software development life cycle, it will be robust and error free. There is no need to re-invent the wheel. To reuse the code, a tool can be create that can extract the metadata such as function, definition, type, arguments, brief description, author, and so on from the source code and store them in OWL. This source code can be stored in the HDFS repository. For a new project, the development can search for components in the OWL and retrieve them at ease. The OWL represents the knowledgebase of the company for the reuse code.

The projects are stored in OWL and the source code is stored in the Hadoop Distributed File System (HDFS) [9]. The client and the developer decide and approve the design document. For the paper the UML class diagram is one such design document considered as the input for the system. The method metadata is extracted from the UML and passed to the SPARQL to extract the available methods from the OWL. Selecting appropriate method from the list the code component is retrieved from the HDFS. The purpose of using an UML diagram as input is before developing software this tool can be used to estimate how many methods is to be developed by extraction. The UML diagram is a powerful tool that acts between the developer and the user. So it is like a contract where both parties agree for software development using UML diagram. After extracting the methods from the UML diagram these methods are matched in the OWL. From the retrieved methods the developer can account for how many are already available in the repository and how many to be developed. If the retrieved methods are more the development time will be shorter. To have more method matches the corporate should store more projects. The uploading of projects in the OWL and HDFS the corporate knowledge grows and the developers will use more of reuse code than developing themselves. Using the reuse code the development cost will come down, development time will become shorter, resource utilization will be less and quality will go up.

The paper begins with a note on the related technology required in Section 2. The detailed features and framework for source code retriever is found in Section 3. The Keyword Extractor for UML is in section 4. The Method Retriever by Jena framework is in section 5. The Source Retriever from the HDFS is in section 6. The implementation scenario is in Section 7. Section 8 deals with the findings and future work of the paper.

II. Related Work

A. Metadata

Metadata is defined as "data about data" or descriptions of stored data. Metadata definition is about defining, creating, updating, transforming, and migrating all types of metadata that are relevant and important to a user's objectives. Some metadata can be seen easily by users, such as file dates and file sizes, while other metadata can be hidden. Metadata standards include not only those for modeling and exchanging metadata, but also the vocabulary and knowledge for ontology [10]. A lot of efforts have been made to standardize the metadata but all these efforts belong to some specific group or class. The Dublin Core Metadata Initiative (DCMI) [11] is perhaps the largest candidate in defining the Metadata. It is simple yet effective element set for describing a wide range of networked resources and comprises 15 elements. Dublin Core is more suitable for document-like objects. IEEE LOM [12], is a metadata standard for Learning Objects. It has approximately 100 fields to define any learning object. Medical Core Metadata (MCM) [13] is a Standard Metadata Scheme for Health Resources. MPEG-7 [14] multimedia description schemes provide metadata structures for describing and annotating multimedia content. Standard knowledge ontology is also needed to organize such types of metadata as content metadata and data usage metadata.

B. Hadoop & HDFS

The Hadoop project promotes the development of open source software and it supplies a framework for the development of highly scalable distributed computing applications [15]. Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment and it also supports data intensive distributed application. Hadoop is designed to efficiently process large volumes of information[16]. It connects many commodity computers so that they could work in parallel. Hadoop ties smaller and low-priced machines into a compute cluster. It is a simplified programming model which allows the user to write and test distributed systems quickly. It is an efficient, automatic distribution of data and it works across machines and in turn it utilizes the underlying parallelism of the CPU cores. The monitoring system then re-replicates the data in response to system failures which can result in partial storage. Even though the file parts are replicated and distributed across several machines, they form a single namespace, so their contents are universally accessible. Map Reduce [17] is a functional abstraction which provides an easy-to-understand model for designing scalable, distributed algorithms.

C. Ontology

The key component of the Semantic Web is the collections of information called ontologies. Ontology is a term borrowed from philosophy that refers to the science of describing the kinds of entities in the world and how they are related. Gruber defined ontology as a specification of a conceptualization [18].Ontology defines the basic terms and their relationships comprising the vocabulary of an application domain and the axioms for constraining the relationships among terms [19]. This definition explains what an ontology looks like [20].The most typical kind of ontology for the Web has taxonomy and a set of inference rules. The taxonomy defines classes of objects and relations among them. Classes, subclasses and relations among entities are a very powerful tool for Web use.

III. Source Code Retriever Framework

The Source Code Retriever makes use of OWL is constructed for the project and the source code of the project is stored in the HDFS [21]. All the project information of a software company is stored in the OWL. The size of the project source will be of terabytes and the corporate branches are

spread over in various geographical locations so, it is stored in Hadoop repository to ensure distributed computing environment. Source Code Retriever is a frame work that takes UML class diagram or XMI (XML Metadata Interchange) file as an input from the user and suggests the reusable methods for the given Class Diagram. The Source Code Retriever consists of three components: Keyword Extractor for UML, Method Retriever and Source Retriever. The process of the Source Code Retriever Framework is presented in the "Fig. 1" The Keyword Extractor for UML extracts the metadata from the UML class diagram. The class diagram created by Umberllo tool is passed as input to the Keyword Extractor for UML.

The input for the frame work can be an existing UML class diagram or created by the tool. Both types of input are loaded in to Umberllo and the file type for storing UML class diagram is XMI format. The file is parsed for metadata extraction. The parser extracts method signatures from the XMI file and passes it to the Method Retriever component. Method Retriever component

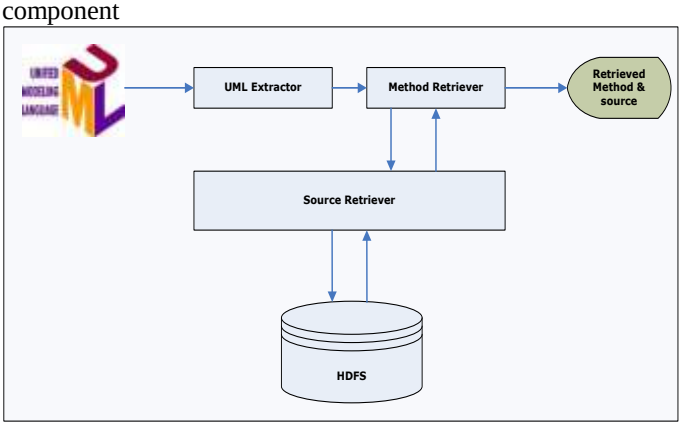


Figure 1. Process of Source Retriever

retrieves the matched methods from the repository. Method Retriever constructs SPARQL query to retrieve the matched results. The user should select the appropriate method from the list of methods and retrieve the source code by Source Retriever component which interacts with HDFS and displays the source code.

IV. KEYWORD EXTRACTOR FOR UML

Unified Modeling Language (UML) is a visual language for specifying, constructing, and documenting the artifacts of systems. It is a standardized general-purpose modeling language in the field of software engineering. To create UML class diagram Umberllo UML Modular open source tool is used. The diagram is stored in XMI format. Umbrello UML Modeller is a Unified Modeling Language diagram program for KDE. UML allows the user to create diagrams of software and other systems in a standard format. Umbrello It can support in the software development process especially during the analysis and design phases of this process. UML is the diagramming language used to describing such models. Software ideas can be represented in UML using different types of diagrams. Umbrello UML Modeller 1.2 supports Class Diagram, Sequence Diagram, Collaboration Diagram, Use Case Diagram, State Diagram, Activity Diagram, Component Diagram and Deployment Diagram.

The XMI is an Object Management Group (OMG) standard for exchanging metadata information using XML. The initial proposal of XMI "specifies an open information interchange model that is intended to give developers working with object technology the ability to exchange programming data over the Internet in a standardized way, thus bringing consistency and compatibility to applications created in collaborative environments. "The main purpose of XMI is to enable easy interchange of metadata between modeling tools and between tools and metadata repositories in distributed heterogeneous environments. XMI integrates three key industry standards: (a) XML - a W3C standard (b) UML - an OMG (c) MOF - Meta Object Facility and OMG modeling and metadata repository standard. The integration of these three standards into XMI marries the best of OMG and W3C metadata and modeling technologies allowing developers of distributed systems share object models and other Meta data over the Internet.

The process flow of Keyword Extractor for UML is given in the "Fig. 2". The XMI or UML file is parsed with the help of the SAX (Simple API for XML) Parser. SAX is a sequential access parser API for XML. SAX provides a mechanism for reading data from an XML document. SAX loads the XMI or UML file and get the list of tags by passing name. It gets the attribute value of the tags by attributes.getValue(<Name of the attributes>) method. The methods used to retrieve the attributes are Parse, Attributes and getValue(nameOfAttibute). The Parse() method will parse the XMI file. The Attribute is to hold the attribute value. GetValue(nameOfAttibute) method returns class information, method information and parameter information of the attribute.

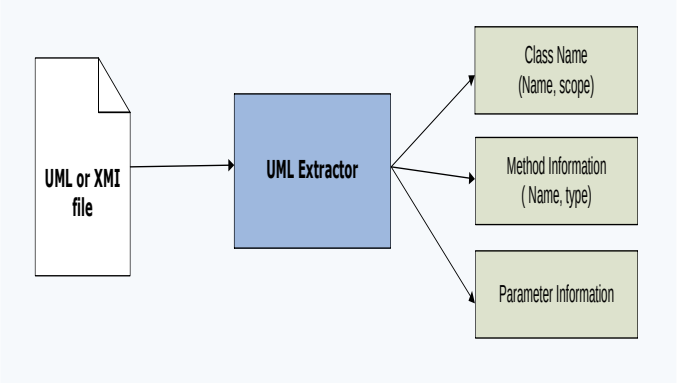


Figure 2. Process of Keyword Extractor for UML

The XMI file consists of XML tags. To extract class information, method information and parameter information are identified with the appropriate tag as given in the Table I.

TABLE I. TAGS USED TO EXTRACT METADATA FROM XMI FILE

Tag	Purpose
UML:DataType	It holds the data type information
UML:Class	It holds the class informations like name of the class, visibility of the class ,etc.,

UML:Attribute	Attribute is a sub tag of class. It holds the informations of the class attributes like name of the attributes, type of the attribute, and visibility of the attribute etc.,
UML:Operation	It holds the methods information of the class like name of the method, return type of the methods, visibility of the method.
UML:BehavioralFeature.parameter	It holds the information of the methods parameters like name of the parameter, data type of the parameter.

Using the tags the metadata of the UML or the XMI is extracted. The extracted metadata are class, methods, and attributes etc., which are passed to the Method Retriever component.

V. METHOD RETRIEVER

Method Retriever component interact with the OWL and returns the available methods from the OWL for the given class diagram is represented diagrammatically in "Fig. 3". The extracted information from the UML file by the Keyword Extractor for UML is passed to the Method Retriever component. It interacts with OWL and retrieves matched method information using SPARQL query. SPARQL is a Query language for RDF. The SPARQL Query is executed on OWL file. Jena is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. Jena is a Java framework for manipulating ontologies defined in RDFS and OWL Lite [22]. Jena is a leading Semantic Web toolkit [23] for Java programmers. Jena1 and Jena2 are released in 2000 and August 2003 respectively. The main contribution of Jena1 was the rich *Model* API. Around this API, Jena1 provided various tools, including I/O modules for: RDF/XML [24], [25], N3 [26], and N-triple [27]; and the query language RDQL [28]. In response to these issues, Jena2 has a more decoupled architecture than Jena1. Jena2 provides inference support for both the RDF semantics [29] and the OWL semantics [30].

SPARQL is an RDF query language; its name is a recursive acronym that stands for ***S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage* used to retrieve the information from the OWL. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs.

A. Query processor

A query processor executes the SPARQL Query and retrieves the matched results. The SPARQL Query Language for RDF[31] and the SPARQL Protocol for RDF[32] are increasingly used as a standardized query API for providing access to datasets on the public Web and within enterprise settings. The SPARQL query takes method parameters and the returns the results. The retrieved results contains project details like name of the project, version of the project and method details like name of the package, name of the class, method name , method return type, method parameter. Query processer takes the extracted method name and the method parameter as an input and retrieves the methods and project information from the OWL.

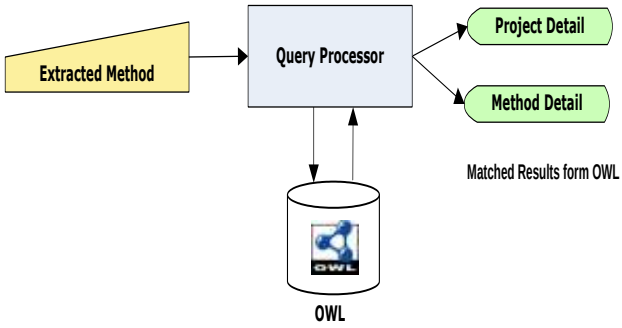


Figure 3. Method Retriever Process

B. SPARQL query

The SPARQL query is constructed to extracting project name, version of the project, package name, class name, method name, return type, and return identifier name, method parameter name and type. The sample query is as follows

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT  ?pname ?version ?packname ?cname ?mname
?rType ?identifier ?paramName ?parmDT ?paramT
WHERE {
?project rdf:type base:Project .
?project base:Name ?pname .
?project base:Project_Version ?version .
?project base:hasPackage ?pack .
?pack base:Name ?packname .
?pack base:hasClass ?class .
?class base:Name ?cname .
?class base:hasMethod ?subject .
?subject base:Name ?mname .
?subject base:Returns ?rType.
?subject base:Identifier ?identifier.
?subject base:hasParameter ?parameter.
?parameter base:Name ?paramName.
?parameter base:DataType ?parmDT.
?parameter base:DataType ?parmT.
FILTER  regex ( ?mname , "add" , "i" ) .
FILTER  regex ( ?parmT  , "java.lang.String" , "i" ) .
}
```

VI. SOURCE RETRIEVER

Source Retriever component retrieves the appropriate source code of the user selected method from the HDFS. It is the primary storage system used by Hadoop applications.

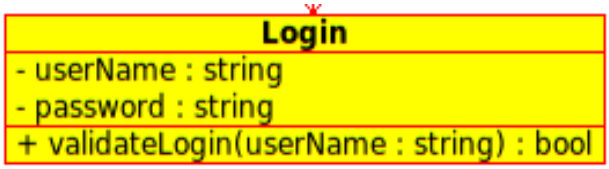
HDFS creates multiple replicas of data blocks and distributes them on compute nodes throughout a cluster to enable reliable, extremely rapid computations. The source code file location of the Hadoop repository path is obtained from the OWL and retrieved from the HDFS by the copyToLocal(FromFilepath,localFilePath) method.

QDox is a high speed, small footprint parser for extracting class/interface/method definitions from source files. When the java source file or folder that consists java source file loaded to QDox; it automatically performs the iteration. The loaded information is stored in the JavaBuilder object. From the java builder object the list of packages as an array of string are returned. This package list has to be looped to get the class information. From the class information the method information is extracted. It returns the array of JavaMethod. From this java method the information like scope of the method, name of method, return type of the method and parameter informations are extracted from the JavaMethod.

QDox finds the methods from the source code. The file that is retrieved from the HDFS is stored in the local temporary file. This file is passed to the Qdox addSource() method for parsing. Through Qdox each method is retrieved one by one. The retrieved methods are compared with methods that the user requested for source code retrieval method. If it matches the source code is retrieved by getSourceCode() method. Then the temporary file is deleted after the process. In Hadoop repository files are organized in the same hierarchy of java folder. So it gets the source location from the OWL and retrieve the java source file to a temp file. The temporary file is loaded into QDox to identify methods. Each method is compared with method to be searched. If it matches; the source code of the method is retrieved by getMethodSourceCode() method.

VII. Case Study

The input for the frame work is a UML class diagram. The sample class diagram is given below



The entire process of the framework is given in the Table II. The Keyword Extractor for UML uses the class diagram and retrieves the method validateLogin(username:string). The output is given to the Method Extractor and generates the SPAQL query and extracts the matched methods which are listed in the Table III. From the list the appropriate method will be selected and the QDox retrieves the source code from the HDFS and displays the method definition of the selected methods as shown in the output of the Source Retriever in Table II.

TABLE II. PROCESS FLOW OF THE FRAMEWORK

Proces	Input	Output
UML Extraction	Given Class Diagram	**Method Information** Name : validateLogin Return : Boolean visibility : public Parameters : User Name DataType : username
Method Retriever	validateLogin(String userName)	Refer Table 2

Source Retriever	validateLogin(String userName)	boolean returnStatus = false;
		DatabaseOperation databaseOperation = new DatabaseOperation();
		String strQuery = "SELECT * FROM login WHERE uname='"+userId+"'";
		ResultSet resultSet = databaseOperation.selectFromDatabase(strQuery);
		try {
		while(resultSet.next()){
		returnStatus = true;
		}
		} catch (SQLException e) {
		e.printStackTrace();
		}
		return returnStatus;

To test the performance of this framework the reusable OWL files are created by uploading the completed projects. The first OWL file is uploaded with first java project. The second OWL file is uploaded with first and the second java projects. The third OWL file is uploaded with first, second and third java projects. Similarly five OWL files are constructed. The purpose of creating OWL is to show how reusability increases when the knowledgebase grows. A sample new project is considered and it contains ten methods to be developed. The OWL files are listed with the number of packages, number of classes, number of methods and number of parameters. These methods are matches with the OWL files and the number of matches is listed in the Table IV.

TABLE III. METHOD RETRIEVER OUTPUT

Sl. No.	Information
1	Project Name : CBR_1.0 Package : com.cbr.my.engine Class Name : Login Method Name : ValidateLogin Parameters : UserName Return Type : boolean
2	Project Name : RBR_1.0 Package : com.my.rbr.utils.engine Class Name : LoginManger Method Name : LoginLog Parameters : UserName,ActivityCode

	Return Type : Boolean Method Name : LoginContol Parameters : UserName,password Return Type : Boolean
3	Project Name : BHR_1.0 Package : com.boscoits.BHR.utils.Action Class Name : ControlManager Method Name : ManageLogin Parameters : UserName,password,memberId,ActionId Return Type : Boolean Method Name : ValidateLogin Parameters : UserName,password Return Type : Boolean

These data in the row of the Table IV shows that the number of matched methods. The reusability graph shown in the "Fig. 4" shows that how the matches increases when the number of projects in the OWL grows. For the graph only five new method names are used instead of ten listed in the Table IV. The X-axis represents the OWL file numbers and the Y-axis represents the number of method matched for the new method legends. This progress shows that by uploading more projects in the knowledgebase can able to provide nearly hundred percent of the methods for reuse during software development.

TABLE IV. NEW METHOD MATCHES WITH VARIOUS KNOWLEDGEBASE

	1 OWL	**2 OWL**	**3 OWL**	**4 OWL**	**5 OWL**
Classes	86	116	129	297	321
Methods	50	1088	1130	3405	3697
Packages	12	15	22	27	31
Parameters	765	1119	1174	4552	4802
Method Name					
ValidateLogin	5	26	27	46	40
getUserType	0	0	0	0	2
addStudent	4	6	6	18	18
ManageRole	6	14	0	28	29
connect	4	5	8	11	16
InsertQuery	2	3	3	5	6
deleteQuery	2	3	3	5	6
updateQuery	2	3	3	5	6
selectQuery	2	3	3	5	6
connect	2	3	3	5	6

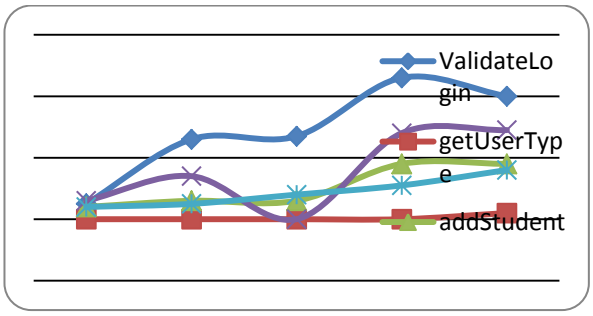


Figure 4. The number of matches for methods to the Projects

VIII. CONCLUSION

The paper presents a framework to extract the method code components from the OWL using the UML design document. OWL is semantically much more expressive than needed for the results of our searching. With these sample tests the paper argues that it is indeed possible to extract code from OWL using the UML class diagram. The purpose of the paper is to achieve the code reusability for the software development. The OWL for the source code has already been created and this paper searches and extracts the code and components and reuses to shorten the software development life cycle. Before starting the coding phase of the development the framework helps the software development team to access the possibilities of how much code can be reused and how much code need to be developed. This assessment can help project manager to allot resources to the project and reduce cost, time and resource. The software companies can make use of this framework and develop the project quickly and grab the project at the lower cost among the competitors.

After developing OWL Ontology and storing the source code in the HDFS, the code components can be reused. This paper has taken design document from the user as input, then extracted the method signature and try to search and match in the OWL. The knowledgebase gets uploaded with more and more projects the reuse rate is also higher. The future work can take the SRS as input; text mining can be performed to extract the keywords as classes and the process as methods. The SRS artifact is much earlier phase than the UML. So considerable amount of time can be reduced than using UML as input. The method prototype can be used to search and match with the OWL and the required method definition can be retrieved from the HDFS. The purpose of storing the metadata in OWL is to minimize the factors like time of development, time of testing, time of deployment and developers. By creating OWL using this framework can reduce these factors.

REFERENCES

[1]. Grigoris Antoniou and Frank van Harmelen, "A Semantic Web Primer", PHI Learning Private Limited, New Delhi, 2010, pp 1-3.

[2]. Bung. M, "Treatise on Basic Philosophy. Ontology I". The Furniture of the World. Vol. 3, Boston: Reidel.

[3]. Gaffney Jr., J. E,, Durek, T. A., "Software reuse - key to enhanced Productivity: Some quantitative models", Information and Software Technology 31(5): 258-267.

[4]. Banker, R. D., Kauffman, R. J., " Reuse and Productivity in Integrated Computer-Aided Software Engineering: An Empirical Study", MIS

Quarterly 15(3): 374-401.

[5]. Basili, V. R.,Briand L. C., Melo, W. L., " How Reuse Influences Productivity in Object-Oriented Systems", Communications of the ACM 39(10): 104-116.

[6]. Boehm B.W., Pendo M., Pyster A., Stuckle E.D., and William R.D.," An Environment for Improving Software Productivity", In IEEE Computer, June 1984.

[7]. Paul R.A., "Metric-Guided Reuse", In proceedings of 7th International Conference on tools with artificial Intelligence (TAI'95), 5-8 November, 1995, pp. 120-127.

[8]. Poulin Jeffrey S., "Measuring Software Reusability", In proceedings of 3rd International Conference on Software Reuse, Brazil, 1-4 November 1994, pp. 126-138.

[9]. Gopinath Ganapathy and S. Sagayaraj, "Automatic Ontology Creation by Extracting Metadata from the Source code ", in Global Journal of Computer Science and Technology,Vol.10, Issue 14(Ver.1.0) Nov. 2010. pp.310-314.

[10]. Won Kim: "On Metadata Management Technology Status and Issues", In Journal of Object Technology, vol. 4, no. 2, 2005, pp. 41-47.

[11]. Dublin Core Metadata Initiative. < http://dublincore.org/documents/>,2002.

[12]. IEEE Learning Technology Standards Committee, http://ltsc.ieee.org/wg12, IEEE Standards for Learning Object Metadata.

[13]. Darmoni, Thirion, "Metadata Scheme for Health Resources" American Medical Informatics Association, 2000 Jan–Feb; 7(1): 108–109.

[14]. MPEG-7 Overview: ISO/IEC JTC1/SC29/WG11 N4980, Kla-genfurt, July 2002.

[15]. Jason Venner, "Pro Hadoop : Build Scalable, Distributed Applications", in The Cloud, Apress, 2009.

[16]. Gopinath Ganapathy and S. Sagayaraj, "Circumventing Picture Archiving and Communication Systems Server with Hadoop Framework in Health Care Services", *in Science Publication* 6 (3) : 2010: pp.310-314.

[17]. Tom White," Hadoop: The Definitive Guide", O'Reilly Media, Inc., 2009.

[18]. Gruber, T. "What is an Ontology?" (September, 2010): http://www.ksl-stanford.edu/kst/what-is-an-ontology.html.

[19]. Yang, X. "Ontologies and How to Build Them. (January, 2011): http://www.ics.uci.edu/~xwy/publications/area-exam.ps.

[20]. Bugaite, D., O. Vasilecas," Ontology-Based Elicitation of Business Rules". In A. G. Nilsson, R. Gustas, W. Wojtkowski, W. G. Wojtkowski, S. Wrycza, J. Zupancic, Information Systems Development: Proc. of the ISD'2004. Springer- Verlag, Sweden, 2006, pp. 795-806.

[21]. Gopinath Ganapathy and S. Sagayaraj, "To Generate the Ontology from Java Source Code", *in* International Journal of Advanced Computer Science and Applications(IJACSA), Volume 2 No 2 February 2011.

[22]. McCarthy, P." Introduction to Jena", www-106.ibm.com/developerworks/java/library/j-jena/, , 22.01.2011.

[23]. B. McBride, "*Jena* IEEE Internet Computing", July2002.

[24]. J.J. Carroll,"CoParsing of RDF & XML*",* HP Labs Technical Report, HPL-2001-292, 2001.

[25] J.J. Carroll, "Unparsing RDF/XML,WWW2002", http://www.hpl.hp.com/techreports/2001/HPL-2001-292.html.

[26]. T. Berners-Lee et al.," Primer: Getting into RDF & Semantic Web using N3", http://www.w3.org/2000/10/swap/Primer.html.

[27]. J. Grant, D. Beckett, "RDF Test Cases*"*, 2004, W3C6.

[28]. L. Miller, A. Seaborne, and A. Reggiori," Three Implementations of SquishQL, a Simple RDF Query Language**,** 2002, p 423.

[29]. P. Hayes, *RDF Semantics*, 2004, W3C.

[30]. P.F. Patel-Schneider, P. Hayes, I. Horrocks, "OWL Semantics & Abstract Syntax", 2004, W3C.

[31]. Prud'hommeax, E., Seaborne, A., "SPARQL Query Language for RDF", W3C Recommendation, Retrieved November 20, 2010, http://www.w3.org/TR/rdf-sparql-query/

[32]. Kendall, G.C., Feigenbaum, L., Torres, E.(2008)," SPARQL Protocol for RDF. W3C Recommendation", Retrieved November 20, 2009, http://www.w3.org/TR/rdf-sparql-protocol/

AUTHORS PROFILE

Gopinath Ganapathy is the Professor & Head, Department of Computer Science and Engineering in Bharathidasan University, India. He obtained his under graduation and post-graduation from Bharathidhasan University, India in 1986 and 1988 respectively. He submitted his Ph.D in 1996 in Maduari Kamaraj University, India. Received Young Scientist Fellow Award for the year 1994 and eventually did the research work at IIT Madras. He published around 20 research papers. He is a member of IEEE, ACM, CSI, and ISTE. He was a Consultant for a 8.5 years in the international firms in the USA and the UK, including IBM, Lucent Technologies (Bell Labs) and Toyota. His research interests include Semantic Web, NLP, Ontology, and Text Mining.

S. Sagayaraj is the Associate professor in the Department of Computer Science, Sacred Heart College, Tirupattur, India. He did his Bachelor Degree in Mathematics in Madras University, India in 1985. He completed his Master of Computer Applications in Bharadhidhasan University, India in 1988. Received Master of Philosophy in Computer Science from Bharathiar University, India in 2001. Registered for Ph.D. programme in Bharathidhasan University, India in 2008. His Research interests include Data Mining, Ontologies and Semantic Web.