Realm for Metamorphose Management in Software Requirements for global Software progress Environment

V PremaLatha, G Rahul, K Pragathi

Abstract: ***Recent outsourcing /off-shoring software development practices testify that any development done without a proper sharing mechanism leads to the generation of inconsistent information, which further results in an undesired, error-prone software. Further, with the business process automation, a significant way to minimize human effort involves various development, support and maintenance activities to reuse available information. Thus, reusing and sharing information in a standardized way is the key operative challenges which foster the need to identify and exploit novel knowledge-based frameworks. Those recommended research gives a tool-based answer for mechanize those programming documentation transform utilizing ontologies. This multi-phase schema need in general six stages the place every period yield contributes of the last robotized documentation.***

Should assess the degree for robotized documentation it is looked at utilizing spare and more open wellspring programming known as WCopyfind of the existing manual documentation to an after effect administration framework research endeavor. Preliminary Outcomes indicate a most noteworthy mechanization from claiming 60 percent, which will be plainly foremost.

Keywords: Software engineering, ontology driven, semantic web

I. INTRODUCTION

The Web/ Mobile Apps Agile Development Practices, Big-Data, Security, Cloud Computing, IoT, Open Source, Customer-first Design are some of the key terms that characterize the latest trends in the software development technology. It has been established across pertinent literature that a well-balancing act between the Iron Triangle or the Project Triangle representing the triple constraints of Time-Cost-Quality along with an added scope and sustainability dimensions can lead to successful software development and delivery. The upsurge in economic globalization has compelled organizations to gain a competitive advantage by cutting their costs, optimizing efficiency, and at the same time provide superlative customer service. Outsourcing /off-shore software development practices have proven to be vital, valuable and profitable for many organizations worldwide as geographically distributed teams can offer huge benefits in terms of efficiency and cost savings. Moreover, the ability to choose team members with the best skills by passing the location-based hiring pool can lead to more focused and strategic development.

Revised Manuscript Received on December 05, 2019.

V. Prema Latha, [1]Asst.Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Educational Foundation, vaddeswaram, Guntur Dt, Andhra Pradesh, India.

G. Rahul[2], K. Pragathi[3] , [2,3,] Department of Computer Science and Engineering, Koneru Lakshmaiah Educational Foundation,vaddeswaram, GunturDt, Andhra Pradesh, India.

In this 'Think Global, Code Local' approach, time-zone coordination, diverse development culture collaboration, team unity around a team charter, strong communication and knowledge sharing technologies arise as significant challenges. Pertinent studies have indicated that any development work that is done with a lack of sharing mechanism leads to the generation of inconsistent information, which further results in an undesired software (Bhatia et al., 2014; 2016b). Moreover, with the business process automation, a significant way to minimize human effort involves various development, support and maintenance activities to reuse available information. Thus, reusing and sharing information in a standardized way is the key operative challenges which foster the need to identify and exploit novel frameworks (Anunobi et al., 2008; Knight & King, 2010).

The above challenge leads to the integration between research fields of Semantic Web (SW) technologies and Software Engineering (SE) (Zhao et al., 2009; Bhatia et al., 2016a; Gašević et al., 2009; Bhatia et al., 2016c) because "Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries" ("W3C", n.d.a; "W3Cb", n.d.b). Alternatively, Software Engineering is "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software" (IEEE Standards Coordinating Committee, 1990).

This reconciliation opened new parkways for specialists to take a gander at different issues also tests that got created because of the amalgamation about SW and SE (Jenkins, 2008). "Around such ontology, particular case will be metaphysics driven programming improvement for robotized documentation (Bhatia et al., 2016a; Bhatia et al. , 2015). Documentation is a fundamental a piece of the programming likewise it is supportive to legitimate correspondence also offering of data. It comprises from sharing different manuals for example, product prerequisites. Specifications (SRS) produced throughout prerequisite building phase, software design document (SDD) generated throughout those plan phase, source book posting made throughout the usage phase, also test information also test outcome made throughout the testing stage. Great quality about documentation gives an incredible help clinched alongside support and also over reverse engineering phase.

However, on large number cases, it will be seen that documentation is not based with correct energy. The documentation may be possibly totally or incompletely ignored, which may be basically due to cosset and plan demand alongside those intricacy for programming framework included (Tang et al. , 2006; Kruchten et al. , 2009). The amount and exertion estimates for the straight sum from claiming documentation bring been elementary worries a really. In IT industry the place alarmed improvement technique may be very much popular, its centre may be that's only the tip of the iceberg ahead individuals and cooperation's the middle of them; product deliverable as opposed protracted documents, client coordinated effort for those advancement cycle, Furthermore once fast reactions on progress (Cockburn, 2002; Martin, 2002). Over such An community oriented improvement nature's domain the place various cross-functional groups need aid working, mechanization about software's documentation may be an essential require should meet those necessities about embraced alarmed standard (Foerstel, 2002; american library Association, 2013).

Hence, the worth of effort introduced in this paper gives an answer the place metaphysics determined product improvement structure to robotized documentation is modelled, implemented, and investigated with an instance study. The result caters couple of the prerequisites for alarmed paradigm-setting and helps done its advantageous useful selection.

In ontology determined software improvement to robotized documentation, an alternate methodology will be embraced for product advancement likewise contrasted with our existing accepted methodologies. Here we create those programming utilizing ontologies. "Ontology is formal and more express determination of an imparted conceptualization" (Studer, 1998). Ontologies need aid manufactured with model a Web-domain also backing thinking through the ideas. Metaphysics building previously, semantic Web is fundamentally underpinned via dialects for example, such that RDF, RDFS Also owl (Ding, 2007).

To achieve ontology determined programming improvement to robotized documentation, To begin with about all, an multi-phase skeleton may be suggested which will be then exhibited utilizing an instance study. This multi-phase skeleton need in general six successive phases, in particular Web-domain metaphysics phase, product prerequisites detail metaphysics phase, configuration phase, source book phase, source book posting metaphysics phase, Also test instances metaphysics stage the place the yield for person period turns into information should following stage.

Every of the phases need its yield which contributes towards last robotized documentation. Further, to assess Also confirm the suggested structure to automation; we look at the robotized documentation of the existing manual documentation of detailed analysis utilizing open sourball programming. This open wellspring product figures crazy those likenesses between two documents which for turns portrays that what rate of mechanization may be attained. Similarly as an after-effect about our work, we attained most extreme 60 percent mechanization when this open hotspot product may be tuned will a standout amongst the proper settings. Another point from claiming this methodology will be that it generates the documentation on mankind's and additionally clinched alongside machine-understandable manifestation (American library affiliation Council, 1939). Likewise it may be likewise accessible in the machine-understandable form, documentation will be nothing from inconsistencies and more ambiguities (Liu, 2005; Knox, 2015).

Whatever remains of the paper may be composed as takes after: segment 2 examines the related work; area 3 introduces the skeleton emulated toward area 4 which expounds those execution of the framework, and segment 5 displays those come about What's more examination. Finally, segment 6 finishes up the paper and gives bearing for future worth of effort.

II. BACKGROUND

To improve the overall description of documentation, de Graaf (2011) and Tang et al. (2011) made an annotating semantic wiki page with the help of lightweight Software Engineering ontology. This lightweight Software Engineering ontology had limited no. of classes and properties. Further, properties comprise of Dublin core data properties to allow specification of metadata. Both the authors created the wiki page with the help of ArchiMind and OntoWiki Semantic Wiki in which the annotated text can be searched. The only difference between their work is that Graaf provided the documentation software requirements and architecture design part only. So far, none of them generated any documentation file. The annotated text in ArchiMind and OntoWiki Semantic wiki is like a tooltip for particular selected text.

López et al. (2012) developed ontology based tool known as Toeska Rationale Extraction (TREx) tool for automating the software architecture documentation part only. They implemented the TREx on a case study and compared its outcome vs. plain text documents, and it was found that humans are poorer than TREx in identifying rationale, but are better at dealing with "generic" components.

de Graaf et al. (2014) also provided an exploratory study on ontology engineering approach for software architecture documentation. They described an approach that uses typical questions for elicitating and constructing an ontology for software architecture documentation. Along with that, they also depicted eight contextual factors that influence the acquisition of typical questions, which are asked from architectural knowledge users. Moreover, they applied this approach on a case study showing how it can be used for obtaining and modeling architectural knowledge needs. Koukias et al. (2015) developed an ontology-based model to represent the technical documentation content towards using it to optimize the performance of the asset. Furthermore, Koukias & Kiritsis (2015) discussed a step-by-step ontology-based approach for modelling technical documentation to optimize asset management using rule-based mechanism.

Bhatia et al. (2015) discussed ontology-based framework for automatic software's documentation where they have just provided the framework, however, to the best of our knowledge, the framework is not realized or implemented so far. This ontology driven software development for automated documentation is also depicted as an open issue problem by Bhatia et al. (2016a) in their literature survey article.

Therefore, as far as ontology driven automated documentation is concerned, Authors' de Graaf, (2011) and Tang et al. (2011) work does not yield any documentation file. While authors' López et al. (2012) and de Graaf et al. (2104) work is limited to architecture documentation only and that too without specifying what percentage of automation is achieved. Moreover, Authors' Koukias et al. (2015) and Koukias and Kiritsis (2015) work is also restricted for modeling purpose only. Author's Bhatia et al. (2015) work is restrained to providing framework only, missing the actual realization or implementation. However, in this paper, we are extending their work by providing ontology driven software development approach for automated documentation that contains all the requisite details of a software system and not just architecture information. Further, we are also generating the documentation file in human as well as in machine-understandable form followed by its comparison with a case study to show what percentage of automation is achieved.

III. SYSTEM WORK

Those proposed multi-phase structure to ontology determined programming improvement to computerizing those documentation transform may be isolated under six phases, to be specific Web-domain ontology phase, programming necessities detail metaphysics phase, plan phase, source book phase, source book posting ontology phase, and test instances ontology period. Figure 1 depicts this system.

3.1 Domain Ontology Phase

In this primary phase, we create a Web-domain ontology, which will be used to catch way ideas of the area under attention. This area metaphysics encapsulates Generally speaking space built information to a real-world situation or issue. Those Web-domain metaphysics may be regularly assembled starting with scratch, In spite of couple of machine learning (ML) systems might a chance to be used to computerize it. However, 100 percent mechanization will be not conceivable due to a few motivations. You quit offering on that one reason might a chance to be those mind boggling way of the real-world issue. For e. g. In the event that of an unpredictable real-world problem, we might unequivocally oblige space masters who might help us for seeing Different space related terminologies. Without Web-domain experts, understanding separate terminologies for example, synonym, homonym, context-dependent interpretation, and so forth throughout this way, observing and stock arrangement of all instrumentation may be enha. Might prompt vagueness. An alternate reason will be that ontology advancement may be an iterative methodology which dives through large portions cycles of revisions and more refinement in front of it may be At last molded and could be utilized. Moreover, following every cycle, a portion mankind's intercession will be needed for specialized foul assessment for produced ontology should assess its quality. In those conclusion for every last one of cycles, the fabricated space metaphysics may be accessible previously, an machine-understandable type which may be then changed over under Web-domain html. This may be fundamentally done to store that space learning under a mankind's readable/ justifiable type for future use. Thus, this period makes and more backs see all the capability previously, both manifestations viz. Machine-understandable and additionally done human justifiable structure.

3.2 Software Requirements Specification Phase

The second phase requires an extension of domain ontology to create Software Requirements Specification (SRS) ontology. The domain ontology serves as the genesis for SRS ontology. IEEE has defined guidelines and standard to organize an SRS document (IEEE Computer Society, 1998a; IEEE Computer Society, 1998b). Based on it, the domain ontology is reconstructed for SRS ontology keeping in mind that it adheres to all the guidelines specified by IEEE. Once this SRS ontology is developed, it is also converted to SRS HTML which can be shared among different geographically and virtually located development teams.

The second phase requires a development from Web-domain ontology with make software requirement specifications (SRS) ontology. The area ontology serves as that genesis to SRS metaphysics. IEEE needs characterized rules What's more standard should sort out an SRS report (IEEE workstation Society, 1998a; IEEE workstation Society, 1998b). In view of it, the space metaphysics is recreated to SRS ontology keeping clinched alongside brain that it adheres with every last one of rules specified by IEEE. Once this SRS ontology will be developed, it will be also changed over should SRS html which could make imparted "around diverse geographically and more essentially spotted improvement groups.

3.3 Design Phase

In the third phase, we develop the design document known as Software Design Document (SDD) which is same as used in conventional development methodology.

3.4 Source Code Phase

This phase could additionally make termed as usage phase and require those advancement also usage from implementation source code alternately the real system satisfying the sum necessities laid down by SRS ontology. That is, in this phase, we perform every last bit frontend What's more backend related exercises to satisfy each Also each enrolled prerequisite.

3.5 Source Code Listing Ontology Phase

In this fifth phase, a source code posting ontology will be built, which empowers us should record every last one of from classes/ variables/ functions utilized within those source code.

Generally, those names about every last one of parts alongside their design previously, source code posting record need aid composed. This documentation permits a better Comprehension of the source code for upkeep design. When this ontology may be constructed, we change over it under its HTML form

3.6 Test Cases Ontology Phase

In this phase, the test situations ontologies need aid created and changed over under html structure. This period ensures that the formed programming may be checked furthermore approved concerning illustration for every SRS and more client desires. Thus, as a yield from these phases, we need area HTML, SRS HTML, source book posting HTML, Furthermore test situations html which unite should mechanize those programming documentation transform. The next area illustrates that usage of this recommended system. The ontology-based approach provides a systematic and structuralized description of knowledge in the development process. We adapt the work in by (A. Maedche and S. Staab. 2001) to define an ontology as O=(l,f,g,c,root,h,s) which consists of lexicon, reference function, concepts, taxonomy, template slots. In other definition, ontology is defined as O=(c,r) where c is a set of ontology concepts and r is relationship between knowledge concepts (G. Zhang, et all. 2010). C is a class which includes class concept Cc, attribute concept Cp, hence C can be shown as a set of C=<Cc, Cp>. R is a finite set of assertions that can be expressed as r={r┤|r ∈C}. After defining classes (c) and class hierarchy (r), we use a combination of Top-Down and Bottom-Up approaches to develop the ontology in (A. Heredia, et all, 2013). In Topdown approach, development we starts by finding the general concept class and we perform subsequent specialization of those concepts. For example, we start building the ontology with two general concepts of "implementation" and "business modeling". We specialize the implementation class by generating appropriate subclasses such as: source code, log history and error tracking. This relationship can be expressed as S={C_x |C_x is an argument of C}. In Bottom-Up development we start with more specific concepts and grows upward to the most general concepts that are super classes of concepts. For example, we start with subclasses of "Java" and "Git" and "Trac" and then we generalize to super classes of "source code", "log history" and "error tracking", respectively. This relationship can be demonstrated as G={C_x |C_x is a child of C} (M.cSensoy and P. Yolum. 2009). Presenting knowledge concepts based on their concept classes, helps us to build the knowledge flow framework. This happens by implementing two methods of specialization and generalization of knowledge concepts.

Figure 1 shows the domain ontology of an enterprise software development processes. It starts with enterprise software development processes as the basis and we expand to general knowledge concepts of specifications, Design and implementation, Integration and testing, Operation and maintenance. Then we use specialization to identify subclass concepts. For example, a super class of "Specifications" can be expanded to subclasses of "Enterprise business modeling", "Portfolio management", "People management", "Enterprise administration", "Business analysis" and finally "Requirement analysis".

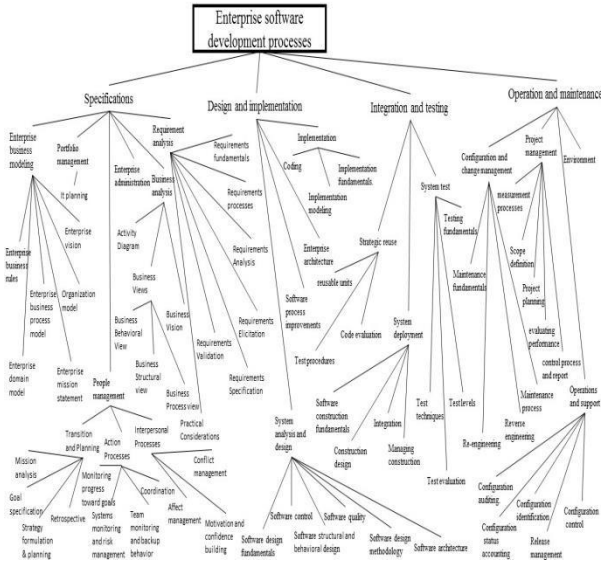


Figure 1. Domain Ontology Of ESD

Similarly, the subclass of Enterprise business modelling can be extended to "Enterprise business rules", "Enterprise domain model", "Enterprise business process model", "Enterprise mission statement", "Organization model", and "Enterprise vision" in (H.-E. Eriksson and M. Penker. 2000). In other words, S Enterprise business modelling={ Enterprise business rules, Enterprise domain model, Enterprise business process model, Enterprise mission statement, Organization model, Enterprise vision } and G Enterprise business modelling=(Specifications). In the domain ontology (Y. Ding and S. Foo. 2002), a subclass of "specification" named people management can be expanded to transition and planning, action processes and interpersonal processes which can be expanded further to S Transition and Planning ={Mission analysis, Goal specification, Strategy formulation & planning, Retrospective}, S Action Processes ={Monitoring progress toward goals, Systems monitoring and risk Management, Coordination, Team monitoring and backup behaviour}, and S Interpersonal Processes ={Conflict management, Motivation and confidence building, Affect management} (S. Bourdeau and H. Barki. 2013). The work (H.-E. Eriksson and M. Penker. 2000), suggested that Gbusiness analysis=(Specification) can be specialized to Sbusiness Analysis={Activity diagram, Business view, Business vision}. Similarly the rest of relationships have been gathered from (A. Abran, et all 2001

Using the provided domain ontology of enterprise software development processes, we are able to infer knowledge requirements of each knowledge concept at any level. For example, project managers have prior knowledge about operation and management and the involved process in a project or manage project planning activities. In this case, knowledge requirements can be presented by enterprise employee in configuration, project management or environment management.

IV. DOMAIN ONTOLOGY FOR SDP

Domain ontology term can be explained in many different ways such as a categorical specification of a group of entities that are objects, concepts and relationship (Jain, V., and Singh, M., 2013). In the field of software development, it provides a common ground for software developers to understand the concepts as well as the systems to interpret definitions of entities. Thus in general every ontology contains important concepts in a specific domain along with relationships and constraints of each concept Protégé in used as the Ontology development tool as it is customizable and effective to create ontologies using an extensible knowledge model. It has an extensible architecture that enables integration with other applications such as graphical representation of domain ontology. Figure 2 shows a view of our proposed domain ontology of enterprise software development processes. Some of the most popular ontology languages are DARPA Agent Mark-up Language (DAML) which is based on the semantic web and incorporates XML mark-ups that enhances interoperability of the language Hendler, J., &McGuiness, D. L. (2000). Semantic web rule language syntax SWRL which includes semantics and abstract syntax (Horrocks, I., et all, 2004), and Web Ontology Language OWL which presents entities and their relationship using formal XML vocabulary (McGuinness, D. L., and Van Harmelen, F., 2004). Protégé is a tool use in our study for the creation and modification of domain ontologies (Jain, V., and Singh, M., 2013). The following figures described the ontology implementation of one part of activity in a software development process selected from Figure 2, as an example of how we implement ontology to describe the knowledge flow among the people responsible in this activity. Figure 3, depicted the Design and Implementation classes, and it's relationship described using protégé tools in SDP domain ontology. Figure 4 shows the roles of individual involved during the Design and Implementation phase that includes developer, analyst, designer and project manager. While Figure 5 shows the individuals relation between the roles during the design and implementation phase in enterprise domain ontology using protégé tools.

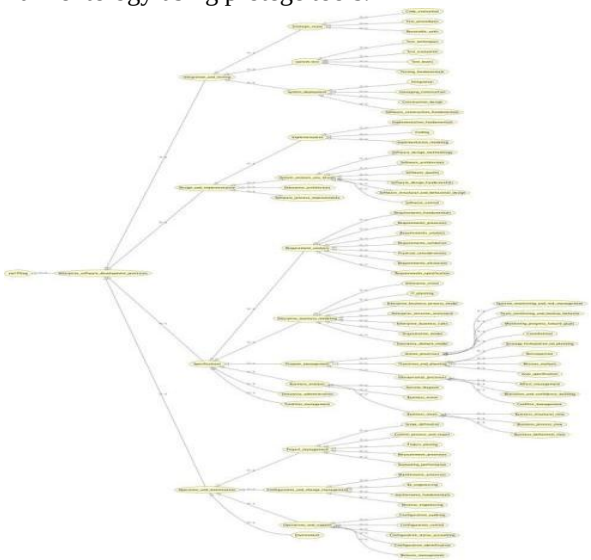


Figure 2. Model View of Complete Domain Ontology

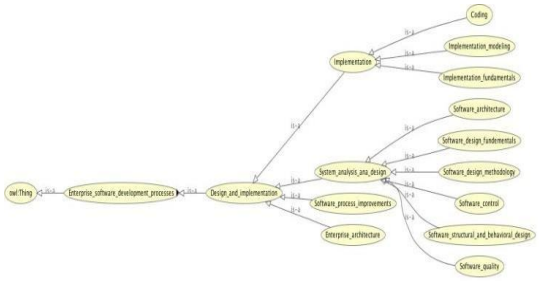


Figure 3.Design and Implementation Classes

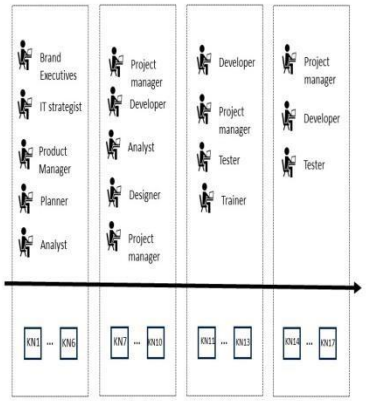
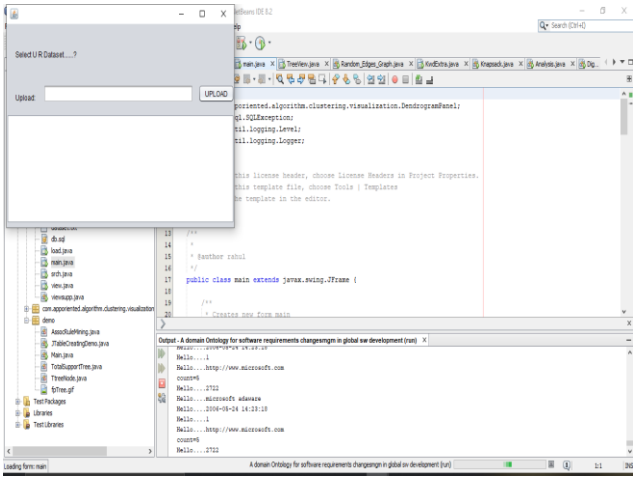
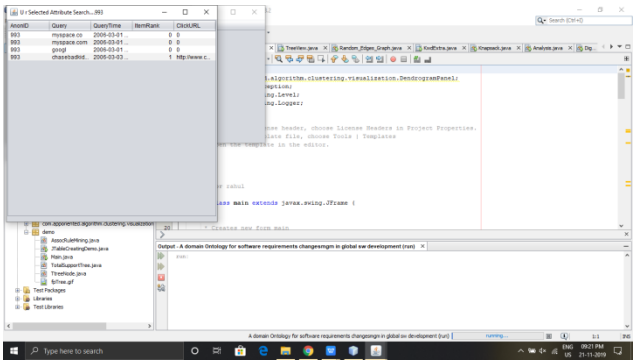
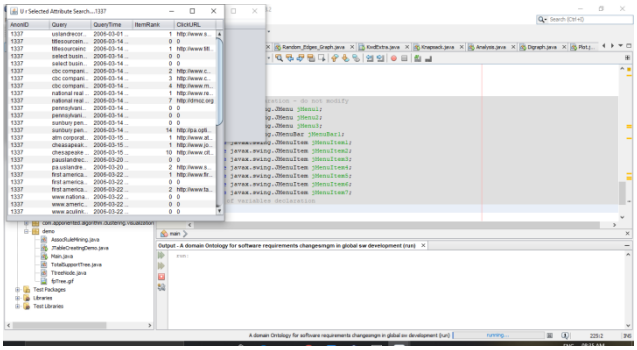


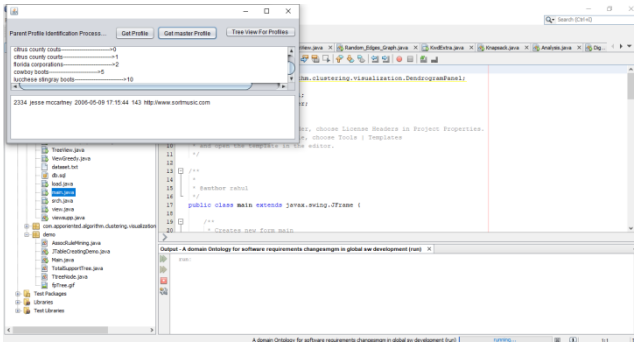
Figure 4: Design and Implementation Role

V. Results









VI. CONCLUSION AND FUTURE SCOPE

Here in this paper, we presented ontology driven software development approach for automated documentation. We displayed the framework for our approach and exhibited its implementation. Moreover, comparison of our approach of automated documentation with the manual one shows 60 percent automation with 01word of shortest phrase to match and 32 percent in case of 06 words of shortest phrase to match. Here we would like to mention that percentage of automation may vary from one case study to another. Also, Wcopyfind software provides similarity index based on similar text and phrases found between two documents. It does not show any similarity when two statements are written differently, yet, conveying the same message. Therefore, the percentage of similarity index is just an indicator of the proportion of automation achieved, nevertheless, actual automation may be even more. Therefore, our effort for automated documentation provides very promising results that the proposed approach is definitely better when it is compared to the manual one. Future work in this area can include further optimization of automated documentation to achieve a higher percentage of automation.

REFERENCES

1. Bhatia, M. P. S., Kumar, A., &Beniwal, R. (2015). Ontology based framework for automatic software's documentation. In *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on* (pp. 421-424). IEEE.
2. Bhatia, M. P. S., Kumar, A., &Beniwal, R. (2016a). Ontologies for software engineering: Past, present and future. *Indian Journal of Science and Technology*, 9(9).
3. Bhatia, M. P. S., Kumar, A., &Beniwal, R. (2016b). Ontology based framework for reverse engineering of conventional softwares. In *Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on* (pp. 3645-3648). IEEE.
4. Bloomfield, L. (n.d.). *WCopyfind*. Retrieved September 15, 2018,
5. Cockburn, A. (2002). *Agile software development*. Vol. 177. Boston: Addison Wesley.
6. de Graaf, K. A. (2011). Annotating software documentation in semantic wikis. In *Proceedings of the fourth workshop on Exploiting semantic annotations in information retrieval* (pp. 5-6). ACM. de Graaf, K. A., Liang, P., Tang, A., van Hage, W. R., & van Vliet, H. (2014). An exploratory study on ontology engineering for software architecture documentation. *Computers in Industry*, 65(7), 1053-1064.
7. Ding, L., Kolari, P., Ding, Z., &Avancha, S. (2007). Using ontologies in the semantic web: A survey. In *Ontologies* (pp. 79-113). Springer US.
8. Drummond, N., Horridge, M., & Redmond, T. (2012). *OWLDoc*. Retrieved September 15, 2018,
9. Gašević, D., Kaviani, N., &Milanović, M. (2009). Ontologies and software engineering. In *Handbook on Ontologies* (pp. 593-615). Springer, Berlin, Heidelberg.
10. IEEE Computer Society. (1998a). IEEE Standard for Software Project Management Plans. *IEEE STD 1058-1998*.
11. IEEE Computer Society. Software Engineering Standards Committee, & IEEE-SA Standards Board. (1998b). IEEE recommended practice for software requirements specifications. Institute of Electrical and Electronics Engineers.
12. IEEE Standards Coordinating Committee. (1990). IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990). Los Alamitos. *CA: IEEE Computer Society*, 169.
13. Koukias, A., Nadoveza, D., &Kiritsis, D. (2015). An ontology–based approach for modelling technical documentation towards ensuring asset optimisation. International *Journal of Product Lifecycle Management*, 8(1), 24-45.
14. Koukias, A., &Kiritsis, D. (2015). Rule-based mechanism to optimize asset management using a technical documentation ontology. *IFAC-Papers OnLine*, 48(3), 1001-1006.
15. Kruchten, P., Capilla, R., &Dueñas, J. C. (2009). The decision view's role in software architecture practice. *IEEE Software*, 26(2), 36-42.
16. Kumar, A., & Gupta, D. (2018). Paradigm shift from conventional software quality models to web based quality models. *International Journal of Hybrid Intelligent Systems*, (Preprint), 1-13.
17. López, C., Codocedo, V., Astudillo, H., &Cysneiros, L. M. (2012). Bridging the gap between software architecture rationale formalisms and actual architecture documents: An ontology-driven approach. *Science of Computer Programming*, 77(1), 66-80.
18. Zhao, Y., Dong, J., & Peng, T. (2009). Ontology classification for semantic-web-based software engineering. *IEEE Transactions on Services Computing*, 2(4), 303-317.