Anti-patterns in Open Source Software Development

Rabia Bashir
DCE, College of E & ME
National University of Science
and Technology (NUST)
Pakistan

ABSTRACT
In software engineering, an Anti-pattern (or Anti-pattern) is a pattern that may be commonly used but is ineffective and/or counterproductive in practice. Anti-patterns or bad practices are not novel; they are ordinary in software industry, and have been since software's inception. The foundation of anti-patterns initiated in 1980 and continued to nurture in software community in 1990s. Analysis of anti-patterns assists to identify the common faults in software projects. These are the practices performed by experienced people or project managers that provide guidance to avoid project failure. We feel *there is a lack of empirical knowledge about anti-patterns in open source software development which direct the practitioners about bad practices that can influence the software quality.*
This paper exposes the software project management anti-patterns and their impact on open source software development where developers work in variable locations, seldom or certainly not meet face to face, and manage their activities through emails and video conferencing. The main aim of the paper is to uncover the anti-patterns, which are present in open source software development and to provide a knowledge based solution to avoid them.

Keywords
Open Source Software, Anti-patterns, Pitfalls, Issues, Bad Practices

1. INTRODUCTION
An anti-pattern is a challenging software project situation. It can be caused by human mistakes (management anti-pattern) or by community/society issues (environmental anti-pattern) [1]. The term anti-pattern was proposed by Andrew Koenig in 1995 motivated by the book named Design Patterns, by Gang of Four which introduced the idea of software design patterns [2]. Anti-pattern paradigm can include bad practice, incorrect response to group of events, failure to forecast, understand or manage project factor etc.
The idea of anti-pattern is intimately associated to project risk, in this way that anti-patterns are prospective risks to a project if their reasons occur during project operation [1]. Anti-patterns can be generated by overview of project cases where wrong assessments have been done. Anti-patterns articulate general mistakes that are made during development of software and their solutions. Therefore, anti-patterns guide us what to avoid and how to handle the problem when we get it. [3]. Anti-patterns are knowledge items and they have to be shared with project management society in efficient way [1]. Furthermore, anti-patterns are normally interconnected and seldom emerge in isolation. Consequently, identifying which anti-patterns present in software project is difficult job which needs special/expert knowledge [4].
Anti-patterns offer real-world knowledge in identifying frequent problems in software development and present the tools to allow software developers and managers to spot these problems and find out their fundamental reasons. Another merit of using anti-pattern is these methods provide general vocabulary of representing and classifying problems and communicating their solutions [4].Open source software development is changing the strategies of classic software development [5]. It is the method by which open source software (publically available source code) is built. Open source products are offered with source code under license to understand, modify and enhance its design and functionality. We consider that there is deficiency of empirical knowledge regarding anti-pattern in open source software development which may guide the experts about bad practices that can affect the quality of software [5]. Normally, open source software is developed by internet-based society of developers [5].
Geographically isolated software developers develop trustworthy and innovative software over the internet; Knowledge is key element for dealing with complex situations and preventing bad decisions and their results. Knowledge on a specific area requires to be represented in structures that assist knowledge capturing, storing, knowledge searching, retrieving and finally reusing [1]. For anti-patterns, a knowledge-based system that can help project managers in identifying bad practices/anti-patterns is needed [4]. Knowledge-base is a repository for storing related information about any specific subject. Knowledge-based systems are one of the applications of artificial intelligence and they transmit knowledge from human being to machine (computer) [4].
The purpose of this research is to first explore the pitfalls/anti-patterns in open source software development and then provide an ontology based knowledge management solution to mitigate these problems. We will start with identifying the key issues / pitfalls in open source development practices and will aim to provide an ontology based knowledge management solution to the issues identified in the first phase.

2. RELATED WORK
In [7], the authors have discussed the issue of maintenance of dispersed developers' group awareness. In their opinion, Open source software development projects are very much shared and dispersed. Except the distance problems these projects have produced huge, multifaceted and successful software systems. However, it is still question that how the dispersed teams handle their teamwork and collaborate with each others. Their main focus is how the geographically scattered teams manage team awareness.
They conducted interviews of software developers, study project communication and come across with different project artifacts of three successful open source software projects. They explored that distributed software developers have

required keeping knowhow of each others, and have common information of whole development team and in depth knowledge of people with whom they are going to work .even if there are various ways of information, but this information is maintained mainly through textual communication (email list and chat tools). These text based channels contain many features that assist to manage the awareness.

Saleh et. al. [8] have proposed a Quality Assessment System Based on Feedback to maintain open source software (OSS) Evolution. To uphold the software evolution of huge OSS is a main challenge. Some aspects that make software difficult to manage are distributed software development teams, constant and quick turnover of volunteers, lack of proper procedures and ways, absence of project planning and project documentation. Their work is based on distant and constant analysis of open source system to observe progress using existing resources like code repository of CVS, exchange of emails and commitment log files. Progress monitoring depends on three major services. The first service evaluates and track the rise in difficulty and fall in quality; the second helps scattered software developers by conveying them report as a feedback after everyone contribution; the third permits developers to get an in depth overview of software by giving a dashboard of project evolution. For the solution of these problems their remote OSS development analysis strategy gives a wide range of services by making use of CVS as data source. It uses a Plugin-based Architecture (PAM) for maintenance of software; it provides various fundamental software modules for repair/maintenance. Moreover, this approach uses internet for communication among CVS and Software Cost-effective Change and Evolution Research (SOCCER) laboratory. SOCCER laboratory [Fig.1] supplies development and quality control mechanisms automatically. Martin et. al. [9] presented quality features of OSS projects achieved by investigatory interviews with OSS developers.

Various quality issues have been identified that OSS are facing. These findings can be utilized as beginning point for future exploration regarding quality in OSS projects, along with the implementation of quality enhancement methods and approaches.

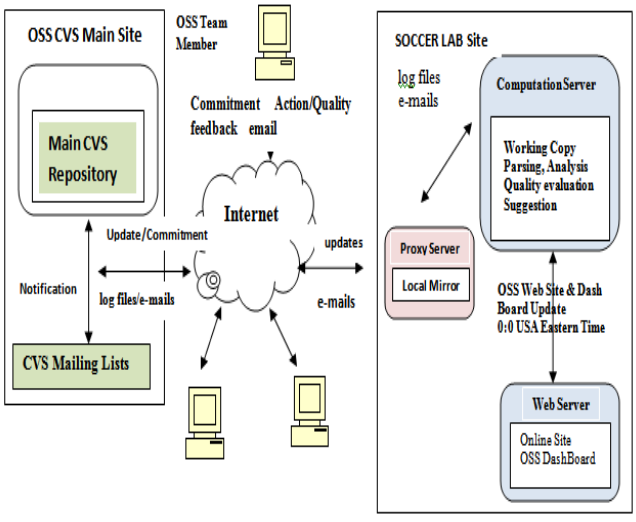

Fig 1: Infrastructure for Remote OOS evolution analysis[8]

They showed the importance of additional research on OSS quality features and recommended to discover other quality problems that are not in their research study and to estimate the effect of diverse practices on software project quality. It has been argued that tools are required to find out and measure quality, both for education to carry out advance studies and for OSS developers to perk up project quality, and those collaborations might be apparent in this field. Anupriya Ankolekar et. al. [10] investigated that semantic web technologies can add different facets of collaboration in OSS environment like maintaining cooperation and teamwork between software developers. Furthermore, they developed ontologies for cooperation tools and developed refined agents that can control the data in those tools and represent significant information. They also stated various challenges to cooperate in OSS development in terms of finding an expert person at the distant location, less communication between team members and reduced awareness in result of less contact and absentness of background or contextual work.

Kevin et. al. [11] have shared case study knowledge of developing small software whose structural model depends on OSS. This study has described the benefits and pitfalls coming together the people, culture and software development practices crosswise an OSS and vendors. Open source products are less documented or sparse or not on hand at all, inadequate testing and may have incomplete requirements.

Klaas et.al .[12] have conducted a literature review and indentified the challenges that emerge during OSS development. Their study is more focused towards the challenges of using OSS in developing the system.

They have demonstrated the results of their study by finding the challenges that exist in developing OSS and classified 21 challenges that can assist the developers to have awareness about them in case of using OSS for the development of product. They are assertive that these findings are worth some for researchers and people from the software industry. Developers will become more alert about the issues and challenges of making use of OSS in developing software and will also assist researchers to think and propose the solution for these challenges.

In [13] author has reviewed the usability related case studies in eight projects of Free/Libre/Open Source Software. Most essential issue that was reported in case studies of reviewed projects was deficiency of user research. Author says dealing with three major challenges of culture, participation and leadership will assist to make it simple to accomplish the user research in these software projects. These projects will be capable to attain good usability without any difficulty and to enhance the usability status of OSS.

In our work we have provided knowledge based solution for anti-patterns in open source software development by identifying two ontologies: Bug ontology and CVS ontology and also provided their implementation.

3. PROPOSED MODEL

Ontologies are defined as a formal specification of a shared conceptualization. [Borst and Akkermans (1997)].They present an opportunity to reuse domain knowledge and coordinate information between persons or software agents. Many procedures and methodologies for developing, maintaining and analyzing ontologies are provided. e.g. Methodology of Uschold and King, Gruninger and Fox's methodology etc. Since there are many anti-patterns of open source projects involved in our case, and it is very hard to cover every aspect of anti-patterns, we are currently focusing on the lack of communication and coordination which is one of the core issues in open source software development.

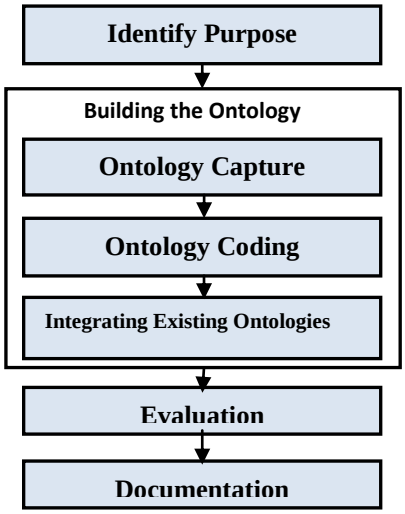

Fig 2. Methodology Structure, Uschold and King's Methodology [14]

To analyze the communication and coordination knowledge and provide a base for sharing of common understanding, we have identified two ontologies:

- Bug Ontology
- CVS Ontology

Many software developing companies use bug tracking. We can use Bug ontology for tracking bugs.
Below are few considerations for a scenario:
Consider the following scenario:
1. Specify a bug and few checks/limitations (bug fixing, bug fixing cost), what is the suitable approach for solution?
1.1 Specify a bug and its time limitations, what is the suitable approach for solution?
1.2 Specify a bug and its cost limitations, what is the suitable approach for solution?
2. Which bugs are with costly solution?
Example:
Bug Ontology:
Concept/Idea: bug (bg), solution(sol),costly-solution(cs),time(tm),cost(co)
Attributes: bug-identifier (bg, id).
Relations/Associations: bug-solution (bg, sol), solution-time (sol, tm), Solution-cost(sol, co).
The basic thought is that it should wind up as powerful and strong *Bug Ontology* that can be used by bugs tracking tool i.e. bugzilla and other data storages to allow the people to query regarding bugs [12-14].

Figure 3 shows the main concepts involved in Bug ontology. One of the major concepts is issue i.e. BugTracker that deals with BugIssue to track the bugs in the source code and the probability of these issues can be major, minor and critical. Other concepts are Project and SoftwarePackage, and issue is related to any project or SoftwarePackage. Project concept is showing the developer, documenter of the project and when the project is released. A repository concept illustrates that project is stored in some repository, which explains the location of that repository and when it is browsed, repositories can be SVN (subversion repository), CVS (concurrent version control) and BK. Issue also represents the status, precedence and the date when it was created or updated. One concept is Description, which presents the details of author, date of the issue and the issue contents. Another major concept is Version, giving an idea of version of the document in terms of version number, previous or next versions. The figure also shows a concept DocumentVersion which uses version concept properties.
Committing concept demonstrates that issue is solved by committing which involves summary of commits, when it was committed and which author has committed.
In addition, many companies have adopted CVS (concurrent version system) to handle and maintain the internal source code.CVS is a tool that is used to keep track the changes done by different software developers in source file; by this way it allows them to remain in sync with one another:

- It present a single method to use for the whole development team and every member works under the one 'ground rule'.

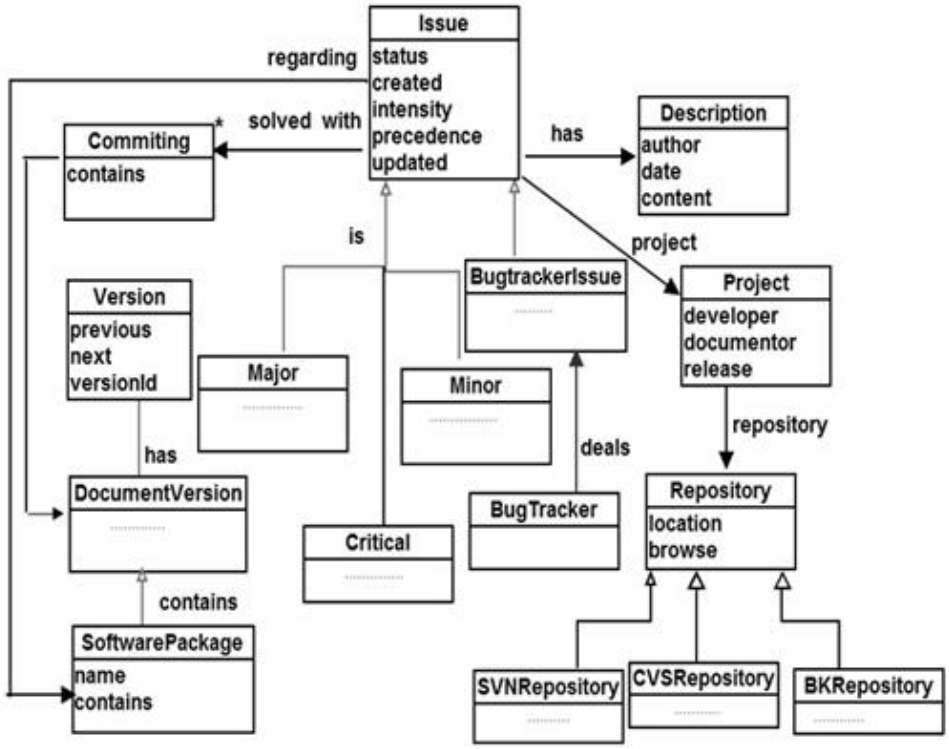

Fig 3: Proposed Bug Ontology

- It has ability to track the alterations, encourage accountability and make it easy to get the right person to resolve the issues in the documents.
- List of accurate changes, can be produced easily and fastly, and makes easy to help and recommend the users that how information is changed from one version to another.
- Changes are in order rather than chaotic, its saves development time.
- If any critical mistake is done during the change then it's easier to "roll back" to the previous version [15].

In figure 4, the main concepts involved in CVS ontology are presented. One of the important concepts in CVS is document versioning to keep track of changes and Version concept indicates the number, date, time and status of the file and documents that are revised .CVS contains many repositories having different directories. Inside the directories there are many documents and files representing their type, the markup language and names respectively. Different opera-

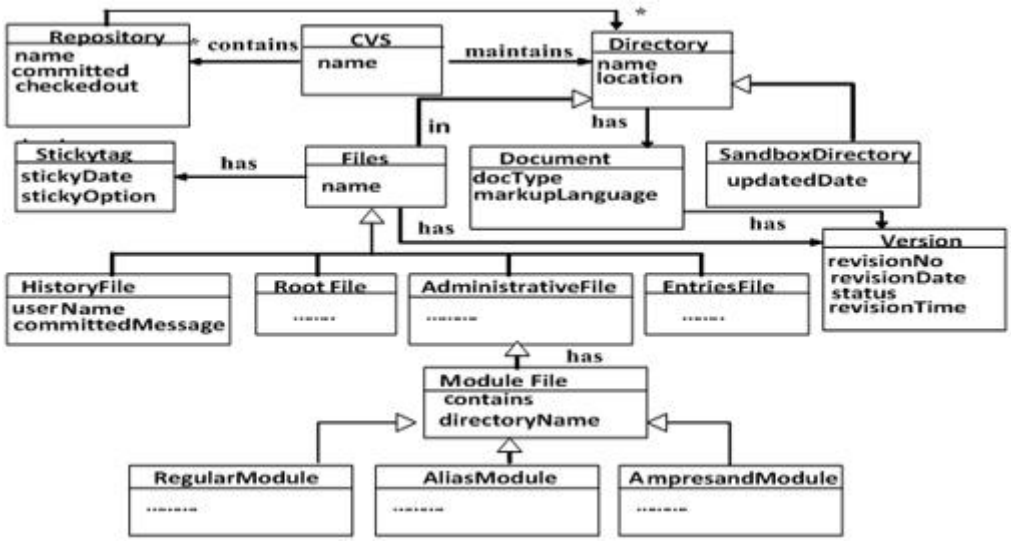

Fig 4: Proposed CV Ontology

tions can be performed on directories like checkedout : which creates working copy in a repository from ongoing project, commit: to copy the alterations to the files in the working copy and backup: to take the backup/copy of the files. One of the important concepts is SandboxDirectory concept demonstrating all those files that are in editable form are present in it.

Another concept is StickyTag that is related to file concept and it illustrates: if we commit any changes then stick tag will be on our working file unless we remove it showing the date when the changes are committed along with any options. There are different types of file concepts like history file concept which contains log of all commit messages and the name of users who have committed the revisions and EnteriesFile concept specifies the timestamp of the local filesin the directory. There are basically three types of module concepts: Alias module concept, Regular module concept, and Ampersand modules concept. AliasModule concept shows the way of defining the *aliases* of the module, AmpresandModule concept refers to other modules and RegularModule concept defines all the files that are present in directory.

4. IMPLEMENTATION

In our work, CVS and Bug ontologies are implemented by using Protégé 4.1. Protégé is open source software to develop knowledge based applications and domain models by using ontologies. To implement ontologies , we have first specified

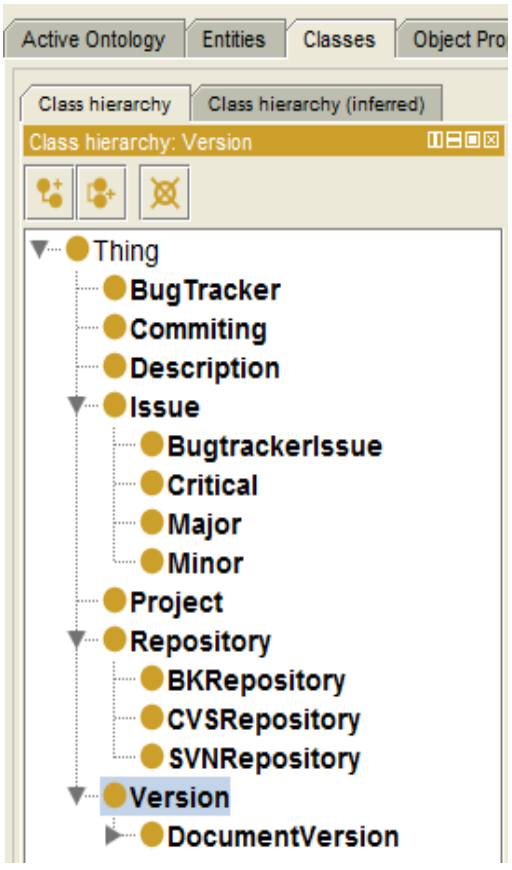

Fig 5: Bug Ontology Representation in Protégé

being implemented using Protégé 4.1 and derived in OWL (Web Ontology Language) with the help of using specific Plug-In of Protégé. We have not presented whole the "Bug" ontology, which is used by many software developing companies for tracking bugs. Bug ontology is specification of our ontology but figure 5 shows Protégé screenshot of it and figure 6 presents its partial Web Ontology Language specification.

Figure 5 shows the hierarchy of Bug ontology in terms of classes in protégé and all these classes (concepts) have been discussed in detail in section 3. OWL code generated by Protégé for the hierarchy of classes is showed in Figure 6. In Protégé OWL Properties are used to represent the relationship or link between two objects. In our work the link between classes are shown by using the property.

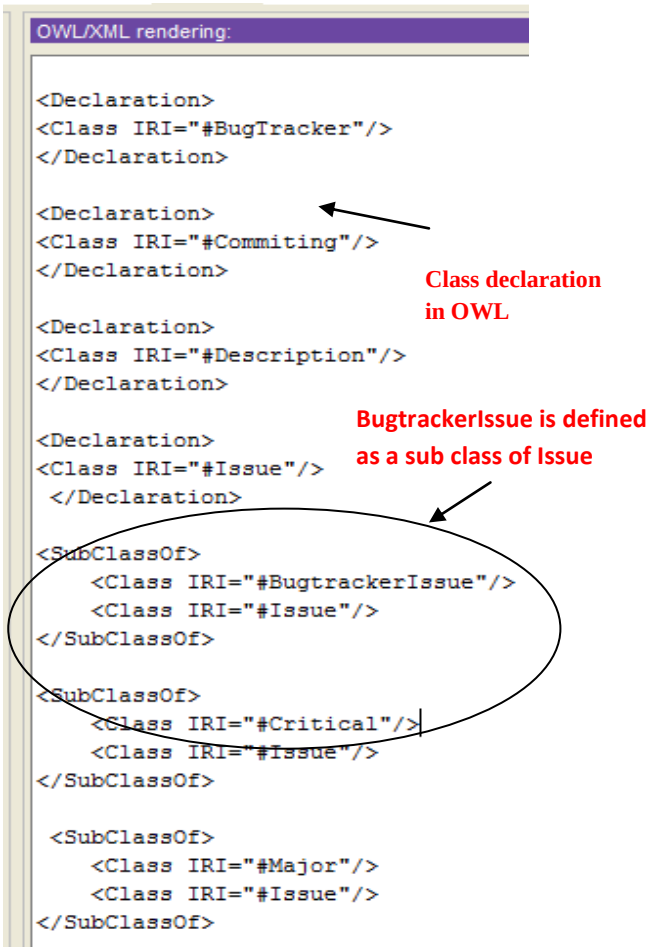

Fig 6: OWL Code for Bug Ontology

Figure 7 demonstrates the object properties along with OWL code. OWL uses domain and range axioms to link the individuals. For example, in our Bug ontology, the property **deals** would link/connect the individuals belonging to the class **BugTracker** to the individuals belonging to the class **BugtrackerIssue** as depicted in figure 8.

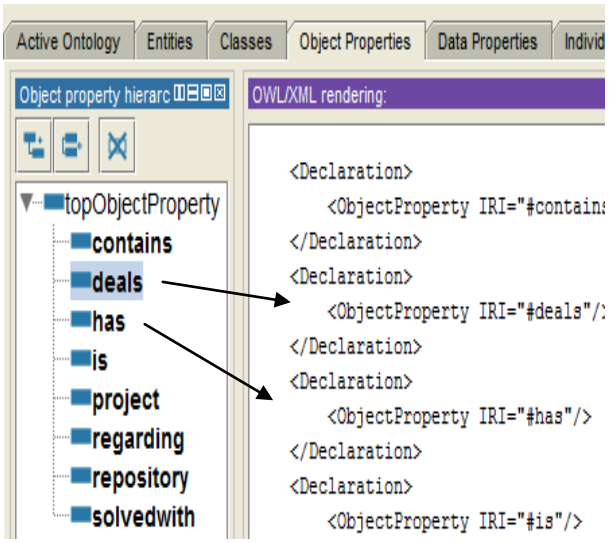

Fig 7: CVS Object Properties

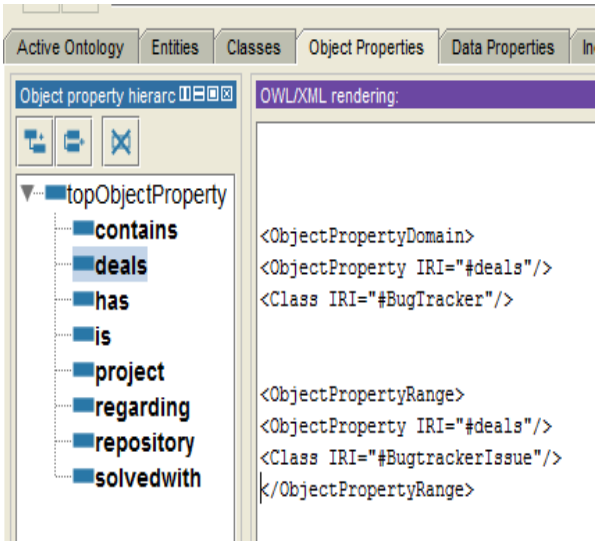

Fig 8: OWL code for link between Bug Ontology Classes

The CVS ontology is implemented by using Protégé-4.1.Figures 9 and 11 illustrate the classes and properties of CVS ontology.

OWL code generated by Protégé for the hierarchy of classes [figure 9a.] is shown in figure 9b.

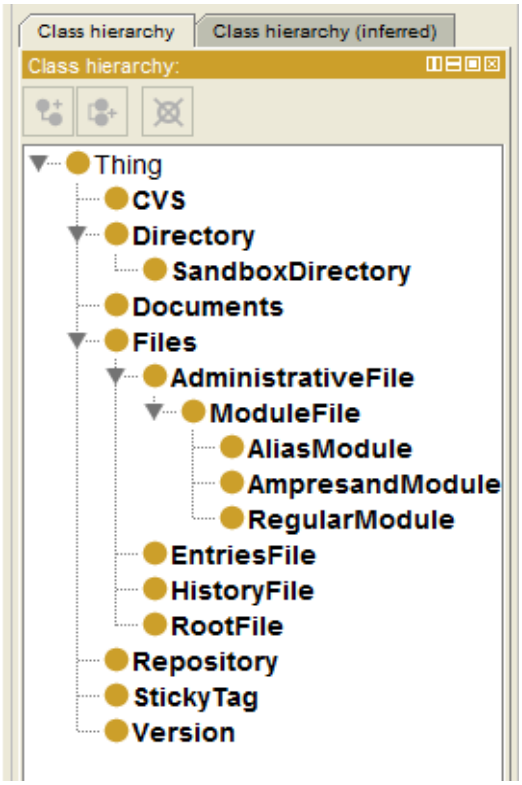

Fig 9a: CVS Ontology Representation in Protégé

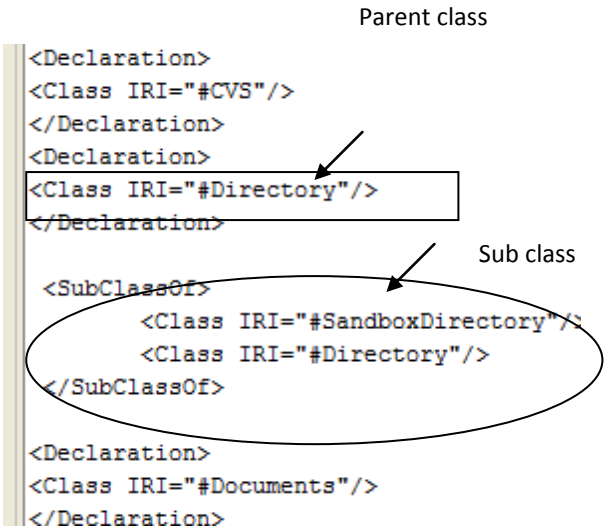

Fig 9b: OWL Code generated by Protégé

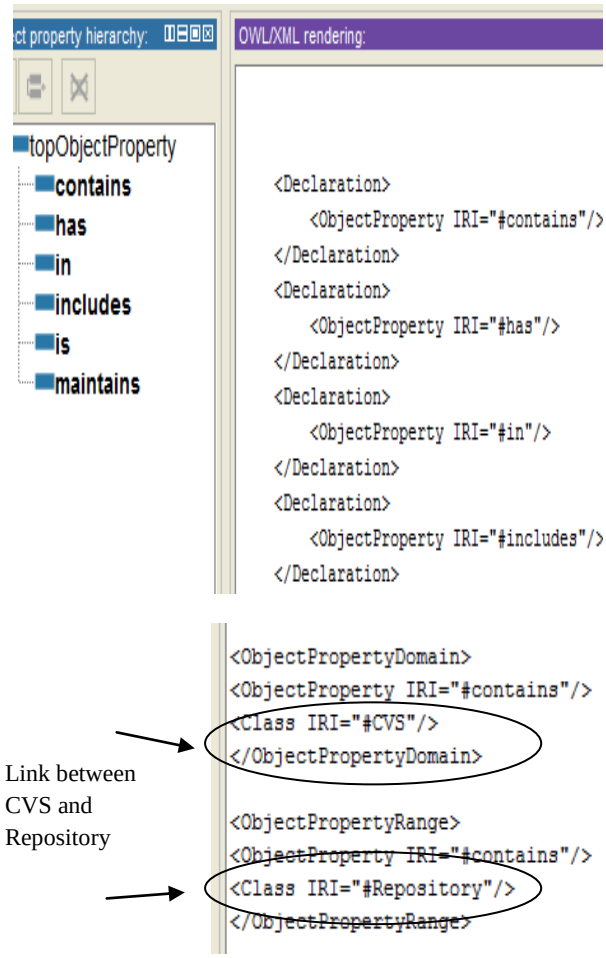

Fig 11: Object properties and relationship between CVS class

The above figure is illustrating OWL code for CVS ontology and the relationship between classes.

5. DISCUSSION

In current years, the open source software development has recognized as a possible choice to other development models. Many famous softwares like Linux, Apache and Samba are created by following open source development practices. However, there are certain anti-patterns in open source software development, because of certain constraints we could cover only communication and coordination which is considered the major problem in OSS .In our work, we have provided the ontology based solution for this anti- pattern and identified two ontologies : Bug ontology and CVS ontology for it by implementing them into software (Ontology Editor) called Protégé 4.1 which is open source software to develop knowledge based applications. Apart from communication and coordination, there are some other anti-patterns or bad practices in OSS like lack of documentation, because developers work on different places in OSS and hard to maintain the documentation in this case and this is also one of the major issues and the lack of documentation illustrates that there is no assertion that all people who are working on the particular software are following same software procedures and methodologies. Unsupported code is also an issue in OSS, when a contributor submit his code for implementing the particular features, if any changes are done by other software developer ,in this case feature needs to be updated so that it carry on to work properly. Unfortunately, some original software developers disappear and the code remains unmaintained. This leads software developers towards difficult situation.

6. CONCLUSION AND FUTURE WORK

This paper has presented insights into anti-patterns in OSS development. We have proposed knowledge based solution for anti-patterns in open source software development. Further research is recommended to classify other anti-patterns in OSS development and to present the ontology oriented solution. We have addressed the issues of lack of communication and coordination anti-patterns in this research work.Few other anti-patterns including lack of configuration management, group awareness, declining quality of OSS and many others for which ontology based solution will be developed in our future work.

7. REFERENCES

[1].Stamelos,I. Software project management anti-patterns. Published in Journal of System and Software, vol. 83, January 2010

[2].Anti-pattern, http://en.wikipedia.org/wiki/Anti-pattern, last accessed 5th January, 2012.

[3]. Smith, C.U and Williams, L.G. Software Performance Anti-Patterns.2000.In proceedings of 2nd international workshop on Software and Performance (WOSP-00)

[4].Settas,D.L., Meditskos,G., Stamelos,I.G and Bassiliades,N. January 2009. PROMAISE: A Knowledge-based System for Software Project Management Antipattern Using Semantic Web Technologies. Technical Report.

[5]. Hars, A. and Ou,S. Working for Free? – Motivations of Participating in Open Source Projects. 2001. Published in Proceedings of 34th Annual Hawaii International Conference on System Sciences

[6].Yamauchi,Y.,Yokozawa,M.,Shinohara,T. and Ishida,T. Collaboration with Lean Media: How Open Source Software Succeeds,Toru.2000. In proceedings of ACM conference on Computer supported cooperative work(CSCW-02)

[7].Gutwin,C., Penner,R. and Schneider, K. Group Awareness in Distributed Software Development. 2004. In Proceeding of the ACM conference on Computer supported cooperative work(CSCW-04)

[8]. Bouktif,S., Antoniol ,G. and Merlo,E. A Feedback Based Quality Assessment to Support Open Source Software Evolution: the GRASS Case Study.2006. In proceedings of 22nd International Conference on Software (ICSM-06)

[9].Michlmayr,M.,Hunt,F and Probert,D. Quality Practices and Problems in Free Software Projects.2005. In Proceedings of the First International Conference on Open Source Systems, pp. 24-28

[10]**.** Ankolekar ,A., Herbsleb ,J.D. and Sycare,K. Addressing Challenges to Open Source Collaboration With the Semantic Web.2003.In Proceedings of the ICSE 3rd Workshop on Open Source.

[11].Gray,K.,Koehnemann,H., Blakley , J. , Goar ,C . Mann,H. and Kagan,A. A Case Study: Open Source Community and the Commercial Enterprise.2009.In proceedings of 6th International Conference on Information Technology: New Generations.pp.940-945.

[12]. Kon,F.,Meirelles,P.,Lago,N.,Terceiro,A., Chavez,C. and Mendonca,M. Free and Open Source Software Development and Research: Opportunities for Software Engineering.2011.Published in Proceedings of 25th Brazilian Symposium on Software Engineering

[13]. Stol,K. and Baber,M.A. Challenges in Using Open Source Software in Product Development: A Review of the Literature.2010.In proceedings of 3rd International Workshop on EmergingTrends in Free/Libre/Open Source Software Research and Development

[14].OntologyEngineering,www.stiinnsbruck.at/.../domain_modeling.../Ontology_Engineering_Christian_Ammendola.pdf,Last accessed 15 February,2012.

[15].Why Use Version Control, http://www.fortnet.org/FortNet/HTML/Presentation/CVS/whyvc.html, Last accessed 10 February,2012.