SOME OBSERVATIONS ON OPEN SOURCE SOFTWARE DEVELOPMENT ON SOFTWARE ENGINEERING PERSPECTIVES

Vinay Tiwari

University Institute of Computer Science and Applications, R.D. University, Jabalpur (MP) (INDIA)
vinaytiwari999@gmail.com

ABSTRACT:

Several argument has been made that open source software development process some times also referred as free and open source software development (FOSSD) violates the traditional software engineering principles and researches advocates to rethink and re-evaluate the studies and concepts of software engineering. The aim of this paper is to investigate the software development process models of Open source software on the software engineering perspectives. In this paper a discussion on some of development models of OSS and their comparison with the traditional development models is made and the software engineering practices followed in open source development environment is also been discussed. Principle difference and similarities of these development models with the conventional models are also discussed. The ultimate goal is to understand how open source software development processes are similar to or different from software engineering processes and to bring better understanding to the development process of Open Source Software.

KEYWORDS:

Open Source Software development, Software Development Process Model, Software Requirements, OSS development model.

1. INTRODUCTION:

Open Source software during the last ten years has gained wide spread popularity. Software like GNU Linux, MYSQL, Apache web server, Mozilla web browser, Open office, Perl programming language etc. getting the phenomenal success among the software users. The amount of free and open source software increasing exponentially with the expansion of internet and currently there are at least hundreds of thousands of such software projects. People everywhere are adopting various open source distributions or participating in the general movement by contributing their own modifications. Open source software has gained a reputation for reliability, efficiency, functionality that has surprised many people in the software engineering world. It is a relatively a new approach for the development of software, where a multitude of people work on the software without apparent central plan or traditional mechanisms of coordination[01]. Free and open source software development is an alternative to traditional software engineering as an approach to the development of software systems and gaining significant importance specially in the production of complex software products. Open Source Software developers have produced systems with a functionality that is competitive with similar proprietary software developed by commercial software organizations. Software development is undergoing a major change from being a fully closed software development process towards a more community driven open source software development process[02].

With the huge success of the open source software, the computer software can now be broadly split into two development models [03] -

DOI : 10.5121/ijcsit.2010.2611

- Proprietary, or closed software owned by a company or individual. Copies of the 'binary' are made public; the source code is not usually made public.
- Open Source software (OSS), where the source code is released with the binary. Users and developers can be licensed to use and modify the code and to distribute any improvement they make.

Open source software development method is aggressive and progressive software development method and starts with problem discovery/ idea from one pupil (mostly software developer) and completed with the help of many software volunteers. Open source software development strategies are quite distinct from that of traditional software development methods. Open source has violated many of the theories of software engineering like limited team size, decentralized project management etc.[04]. FOSSD is a way for building deploying and sustaining large software systems on a global basis and differs in many interesting ways from the principles and practices traditionally advocated for software engineering [05]. In this paper we study the Nature and characteristics of open source software and strive to clarify some of the main principles followed in open source development. In the beginning, some important aspects of the entire OSSD approach are introduced. Important characteristics of Open source software development are explained and typical roles, processes and an overall lifecycle model of typical OSSD projects is also discussed. Subsequently paper examines and compares practices, patterns, and processes of OSSD with the traditional software development models.

2. WHAT IS OPEN SOURCE SOFTWARE?

Open source software is computer software that is available in source code form that permits users to study the software, to use software freely, change and improves software as per his requirements. Generally open source refers to a program in which the source code is available to the general public for use and/or modification from its original design free of charge. OSS as defined by the open source Initiative [06] is "software that must be distributed under a license that guarantees the right to read, redistribute, modify and use the software freely". Feller and Fitzgerald [07] have outlined the following key conditions to define OSS:

1. The source code must be available to user.
2. The software must be redistributable.
3. The software must be modifiable and the creation of derivative works must be permitted.
4. The license must not discriminate against any user, group of users, or field of endeavor.
5. The license must apply to all parties to whom the software is distributed.
6. The license cannot restrict aggregations software.

Open source software generally developed in voluntarily basis by the global network of developers and available free on the internet. OSS is often described as 'free' software, which reflects the liberty not the price of the software[08]. The OSS license give users to following four essential 'freedoms'[09]:

- to run the program for any purpose,
- to study the working of the program, and modify the program to suit specific need,
- to redistribute copies of the program at no charge or for a free, and
- to improve the program, and release the improved modified version

3. OSS DEVELOPMENT PROCESS:

Generally, an OOS project begins with a personal need of a single developer who has a vision and tries to devise solutions for his unmet need calls this "scratching an itch" [10]. Then he or she starts and discussion with his friends and colleagues about the possible solution and making the code base. He makes this code available to others which attract the attention of other user-developers and inspire them to contribute to the project in this way the initial project community is formed and the development proceeds. Typically, anyone may contribute towards the development of the system and built Open Source Community to provide administration for the project. This initial community of interested persons start to exchange their knowledge on the topic and start working on the issue until they achieve some satisfactory result. They make their work publicly available at a place where many people are able to access it. They may announce their project at places like mailing lists, newsgroups or online news services. Other persons recognize some of their own concerns in the project and are interested in a convenient solution, too. Therefore, they review the projects result (e.g. by using it). As they look at the issue from a different perspective, they suggest improvements and even might join the project. These users now known as co-developer, helps in rapid code improvement and effective debugging. As the project grows more and more people get attached and a lot of feedback helps to get a better understanding of the issue, and possible strategies to solve it. New information and resources are integrated into the research process. The solution grows, and addresses the issue in ever better ways. The project's community is established and will react to future changes the same way it emerged originally.

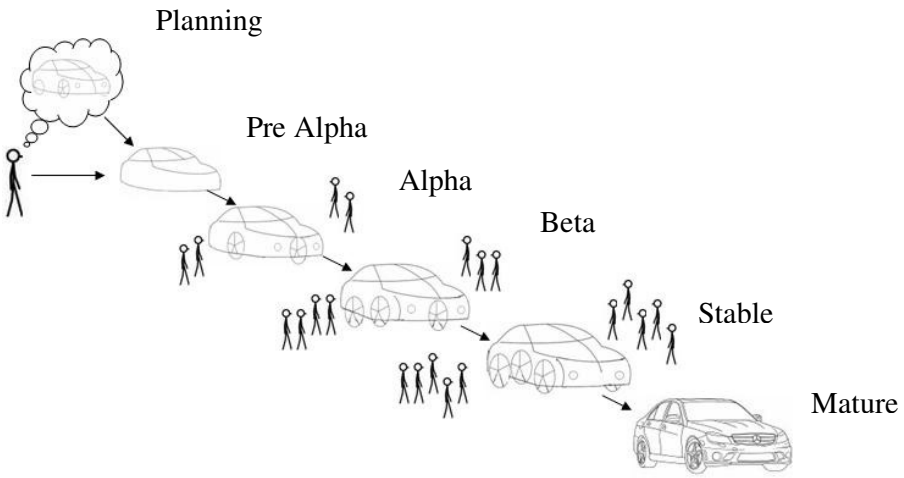

Figure 1. OSS Development Process

Gregor J. Rothfuss [11] in his research classified the various stages of an OSP is Planning, Pre-Alpha, Alpha, Beta, Stable, Mature as follows:

Planning

No code has been written, the scope of the project is still in flux. The project is but an idea. As soon as tangible results in the form of source code appear, the project enters the next stage.

Pre-Alpha

Very preliminary source code has been released. The code is not expected to compile, or even run. Outside observers may have a hard time to figure out the meaning of the source code. As soon as a coherent intent is visible in the code that indicates the eventual direction, the project enters the next stage.

Alpha

The released code works at least some of the time, and begins to take shape. Preliminary development notes may show up. Active work to expand the feature set of the application continues. As the amount of new features slows down, the project enters the next stage.

Beta

The code is feature-complete, but retains faults. These are gradually weeded out, leading to software that is ever more reliable. If the number of faults is deemed low enough, the project releases a stable version, and enters the next stage.

Stable

The software is useful and reliable enough for daily use. Changes are applied very carefully, and the intent of changes is to increase stability, not new functionality. If no significant changes happen over a long time, and only minor issues remain, the project enters the next stage.

Mature

There is little or no new development occurring, as the software fulfills its purpose very reliably. Changes are applied with extreme caution, if at all. A project may remain in this final stage for many years before it slowly fades into the background because it has become obsolete, or replaced by better software. The source code for mature projects remains available indefinitely, however, and may serve educational purposes.

4. CATHEDRAL AND BAZAAR

The Book "The Cathedral & the Bazaar" written by Eric S. Raymond, president of the Open Source Initiatives (http://www.opensource.org), is the most frequently cited description of the open source development methodology. It can be said that the research on OSSD was started by this book. In this book Raymond explains the major differences between the two types of software creation. He summarized his OSSD experiences into 19 lessons for software programmers, which have become the recommended practices of this paradigm.

According to Raymond conventional closed source development is like the building like a cathedral; central planning, tight organization and one process from start to finish. The progressive open source development is more like a "a great babbling bazaar of differing agendas and approaches out of which a coherent and stable system could seemingly emerge only by a succession of miracles". In his view, open source represents the bazaar, a place where people freely trade their wares and skills, and the proprietary movement is represented by the cathedral, a bricks-and-mortar institution with little flexibility for change.

The Cathedral model represents the traditional commercial software development style, using small teams, tight management control, and long release intervals. The Bazaar model represents the style of releasing early often involving a large number of pool of developers working on the product. It challenges the traditional software engineering models of using a large team of developers and testers Some of the most important ideas in Cathedral & the Bazaar include:

- Brooks' Law does not apply to Internet-based distributed development;
- "Given enough eyeballs, all bugs are shallow";
- Linux belongs to the Bazaar development model;
- The OSP model automatically yields the best results;
- The Bazaar development model is a new and revolutionary model of software development.

All these ideas are vulnerable to varying degrees. Understanding the weak points of Cathedral & the Bazaar helps to develop stronger, more comprehensive theories later.

5. TRADITIONAL LIFE CYCLE MODELS:

A software development process or life cycle is a structure imposed on the development of a software product. In this section an overview of some life cycles models for traditional software development is briefly discussed which will be later useful in the present study. The first published model of the software development process was **Waterfall model**. It begins at the system level and progress through analysis, design, coding, testing and support. Still it is well suited to projects which have a well defined architecture and established user interface and performance requirements.

When the requirements and user's needs are unclear or poorly specified a **Prototyping model is used.** It was advocated by Brooks. The approach is to construct a quick and dirty partial implementation of the system during or before the requirements phase. This quick design or prototype is evaluated by customer/user and used to refine requirement for the software to be developed.

A better model, the **"spiral model"** was suggested by Boehm in 1985. The spiral model is a variant of "dialectical spiral" and as such provides useful insights into the life cycle of the system. This model can be considered as a generalization of the **prototyping** model. That why it is usually implemented as a variant of prototyping model with the first iteration being a prototype[12].

Agile represent new approaches in the spectrum of software development methods. The aim of these practitioner-oriented software development methods is to make a software development unit more responsive to changes. These changes are imposed by rapidly evolving technology, changing business and product needs. Agile software development uses iterative development as a basis but advocates a lighter and more people-centric viewpoint than traditional approaches. Agile processes use feedback, rather than planning, as their primary control mechanism. The feedback is driven by regular tests and releases of the evolving software.

The variation of agile process is **Extreme programming** (XP) in which the phases are carried out in extremely small (or “continuous”) steps compared to the older, batch process. It is the latest incarnation of Waterfall model and is the most recent software fad. XP try improve classic waterfall model by trying to start coding as early as possible but without creating a full-fledged prototype as the first stage.

6. RELEATED WORK AND EXISTING OSS DEVELOPMENT MODELS:

There are many theoretical approaches that try to explain the phenomenon of open source. But still no generally agreed well defined standard development model for open source software exists. Open source processes can vary from project to project. There is no single universal approach to FOSSD. Projects differ a lot from each other, and there are differences even in the workings and organizational approach of a single project over time. Classifications of different development styles have been made, but there is no general consensus on taxonomy of projects[01]. The understanding towards the development process begin with the suggested model of Eric S. Raymond known as *Bazaar model* in which he described OSS development as “a great babbling bazaar of differing agendas and approaches”.

Daniel Blaney et al.[13] described the Open Source Model as parallel software development on an unprecedented and wholly innovative scale. In this model, a handful of project leaders offer up a novel software idea over the Internet and the best and brightest minds appear in droves to attack the problem, often without compensation. Amazingly, these contributors are often

scattered all over the globe and almost never meet face-to-face. The basic idea upon which most writers and techno-historians agree upon is that *people*, not practices and tools, drive this model and make it work. Before the Internet came about, the power of a "collective mind," so to speak, had never been realized.

Open Source Software Development is an orthogonal approach to the development of software systems where much of the development activity is openly visible, development artifacts are publicly available over the Web, and generally there is no formal project management regime, budget or schedule [14]. Open Source Software Development is oriented towards the joint development of community of developers and users concomitant with the software system of interest as compared with traditional software development and maintenance [15]. Sharma et al [16] suggested that typically in an OSS project, developers iterate through a common series of actions while working on the software source. The development process of an OSS project consists of the following visible phases:

1. Problem discovery
2. Finding volunteers
3. Solution identification
4. Code development and testing
5. Code change review
6. Code commit and documentation
7. Release management

Life cycle starts with the problem discovery by discussing with active developers. An agenda file with a list of high priority problems, open issues and release plans is stored in each product's repository to keep track of project status. Once the problem is discovered, volunteers are found to work on the problem. Volunteers prefer to work on problems that are related to the areas they are familiar with and have been working on. After having found volunteers to work on a problem, the next step is to identify the solution. Usually, many alternative solutions are available. Developers choose solutions for their generality and portability. The chosen alternative is posted to the developer mailing list for feedback before it is implemented. Once the solution has been identified, code is developed. The developer makes changes to a local copy of the source code, and tests the changes in his or her own environment. The tested solution is posted to the developer mailing list for review. Individual developers on the list further test this solution. If they find any problems with the solution, they suggest improvements to the originator. After a careful review, the originator makes changes to the code and again tests the solution and posts the improved solution on to the list. The process is repeated until it is approved. Once the tested solution is approved by the list, it can be committed to the source by any of the developers, although it is preferred that the originator of the change performs the commit. Each commit results in a summary of changes being automatically posted to the Concurrent Version Control System (CVS) mailing list. All the members of the core group review the changes to ensure that changes are appropriate. Changes are also reviewed by developers outside the core group. A core group member volunteers to serve as the release manager as the project nears a product release. The release manager identifies outstanding problems and their solutions and makes suggested changes. The role of release manager is rotated among the members of the core group.

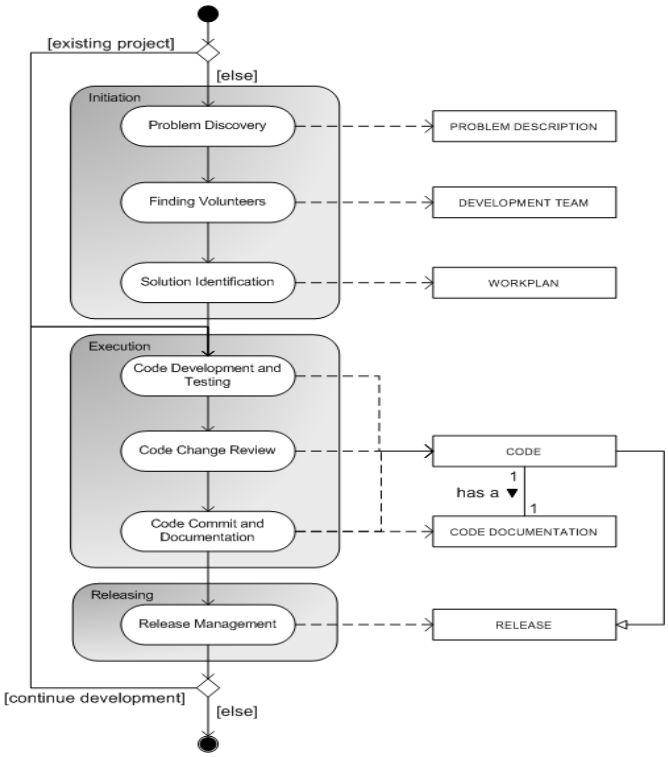

Figure 2. OSS Development Process (source wikipedia)

Ming-Wei Wu and Ying-Dar Lin [17] proposed a development model for open source by incorporating the open source licensing and version control as shown in figure 3.

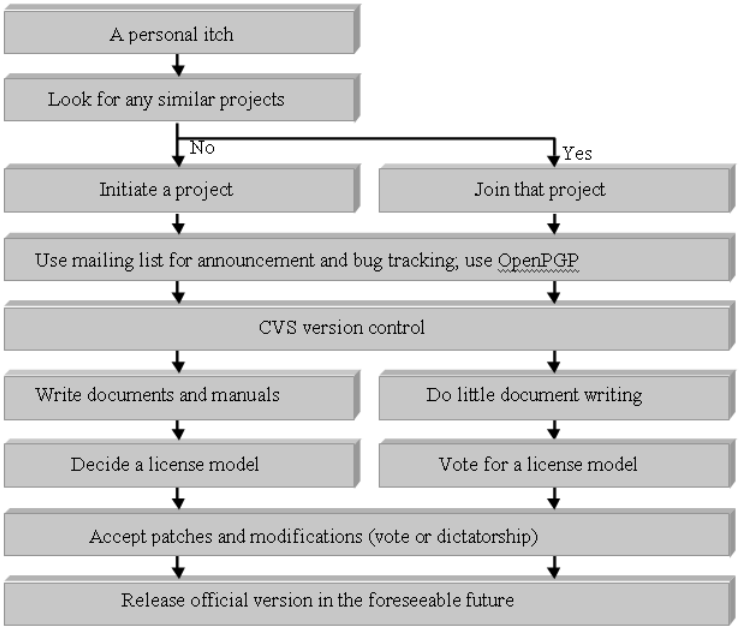

Figure 3. Open source system development cycle Source: (Wu and Lin, 2001, p.34)

The open source software development cycle, allows literally any-one to participate in the process, but having multiple participants means a massive coordination effort. Participants or co-developer scattered across the globe must agree on a version control system to avoid development chaos. Before the official release a licensing model must be decided from the three general categories i.e. free- the program can be freely modified and redistributed; copyleft the owner gives up intellectual property and private licensing finally GPL compatible where licenses are legally linked to the GPL licensing structure.

Schweik and Semenov [18] propose an OSSD project life cycle comprising three phases: 1. project initiation 2. Going 'open', and 3. Project growth, stability or decline. Each phase is characterised by a distinct set of activities. Requirements for a new project are often based on what open source developers themselves want or need. During project initiation, the project core is developed upon which others build. Going 'open' involves a choice on the part of the project founders to follow OSS licensing principles. It also ensures that the project enjoys the support of a core group of dedicated developers, shows technical promise and is of interest to future developers, and that a sufficient amount of the original requirements have been solved to create a framework in which future development can take place. In this phase appropriate technologies and web sites need to be chosen to act as a vehicle for sharing code and recruiting developers. The final phase, growth, stability or decline, poses an element of risk for open source projects: will the project generate enough interest to attract developers and users globally to use the product and participate in further programming, testing or documentation.

Mockus et al [19] describe a life cycle that combines a decision-making framework with task-related project phases. The model comprises six phases:

1. Roles and responsibilities,
2. Identifying work to be done,
3. Assigning and performing development work,
4. Pre-release testing,
5. Inspections, and
6. Managing releases.

The model has a strong managerial focus emphasizing developer management and the work to be done, rather than on product-related activities. The model proposed by Mockus et al adequately caters for the planning phase of the SDLC but is less explicit regarding other phases. Furthermore, Mockus et al assume that some sort of prototype already exists, failing to explain where design and analysis phases occur within their model.

Jorgensen [20] provides a more detailed description of specific product related activities that underpin the OSSD process. His model is shown in figure 4.

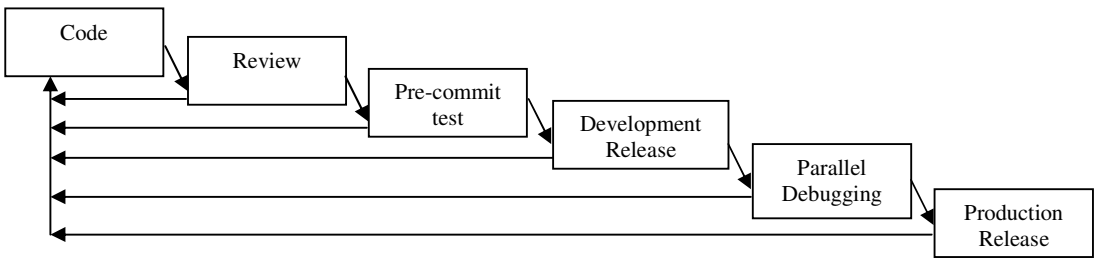

Figure 4: Jorgensen life-cycle

stages or sets of activities proposed are:

Code: Code is submitted by talented developers for review and improvement by respected peers.

Review: Most (if not all) code contributions are reviewed. Truly independent peer review is a central strength of this process.

Pre-commit test: Review is followed by an unplanned, yet thorough, testing of all contributions for a particular code change. While informal, this phase is taken very seriously as negative implications of permitting a faulty contribution can be considerable.

Development release: If the code segment is deemed release-ready it may be added into the development release.

Parallel debugging: Development releases of software undergo a rigorous debugging phase where any number of developers is able to scrutinize the code in search of flaws.

Production release: Where development versions are deemed stable, they are released as production versions.

The process is repeated incrementally for new modules – reinforcing the cyclical nature of all open source projects where there is no real end point - unlike many commercial projects. Jorgensen's model is widely accepted as a framework for the OSSD process.

Rinette Roets et al [09] expands on Jorgensen's life cycle model and incorporates aspects of previous models. Their model attempts to encapsulate the phases of the traditional SDLC as shown in figure 5.

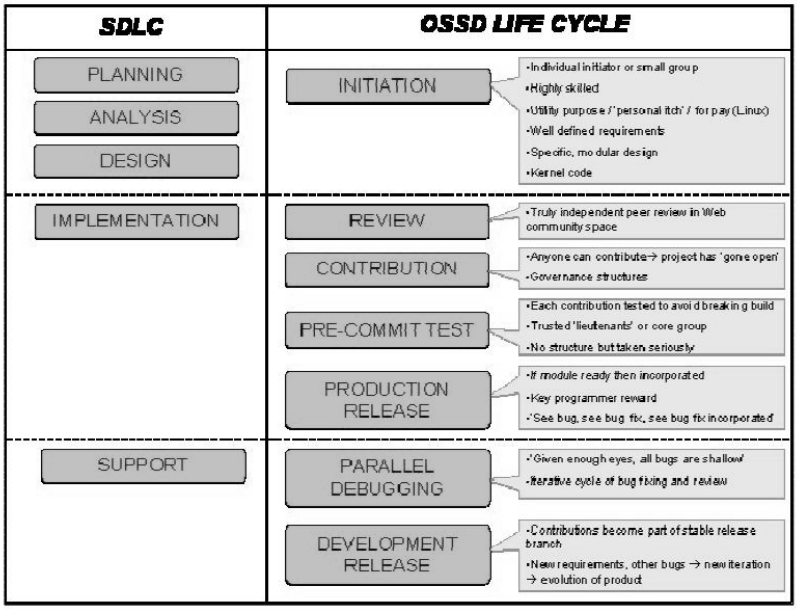

Figure 5: Model of Rinette Roets et al [09] and its comparison with SDLC

The OSS development life cycle is very different to the traditional one of planning, analysis, design and implementation. Planning, analysis and design phases are largely undertaken by the initial project founder and thus may not be part of the OSS development cycle. The OSS development life cycle is located primarily within the implementation phase.

Although many existing OSS projects have successfully developed individual practices and specific processes, it is possible to define some common characteristics that can be identified in most of the OSS Development projects [21].

- Collaborative development
- Globally distributed actors
- Voluntariness of participation
- High diversity of capabilities and qualifications of all actors
- Interaction exclusively through web-based technologies
- Individual development activities executed in parallel
- Dynamic publication of new software releases
- No central management authority
- Truly independent, community-based peer review
- 'Bug driven' development

7. OSS VS TRADITIONAL DEVELOPMENT MODELS:

From the previous discussion it is clear that open source software development is a relatively new approach for the development of software, where a multitude of people work on the software without apparent central plan or traditional mechanisms of coordination and without any clear model. Whereas traditional development methods which follows a software engineering principals is essentially a science about how software should be made not how it is made. We have seen that the model of OSS development differs significantly from the traditional software engineering models as described in text books. Therefore the comparison between the two is like a comparison between reality and idealized model of development. Comparison between the two can not tell us which one is better than the other. So the best way is to find out where the Open source software development following the software engineering principles and at which point it is not following the guidance of software engineering.

In a open source development it is hard to provide a strict schedule for the whole project and provide a precise process model for it because it does not have formal document for Requirement, Design, Testing, and so on[14]. OSS project generally begins with a single developer who has a personal goal or vision whereas in traditional software development, software companies first try to find the needs of their customers, and then use some process models for their development. The development process consist some strict phases, such as Requirement Engineering, Testing etc. Although the requirement engineering as per the software engineering principles is not done in OSS project but on OSS the developers are users of the software, they understand the requirements in a deep way. As a result, the ambiguity that often characterizes the identification of user needs or requests for improvement in the traditional software development process is eliminated as programmers know their own needs[22].

The traditional in-house software developments rely on employees in one or several companies generally situated in same country. The open source development is totally different. It has contributors living all over the world. They speak different languages; have different culture and education backgrounds. Therefore, it is impossible for design a unify standard for code, documents. Besides, the OSS development teams are not as stable as corporation-dominated development during the whole life cycle of the software. More people leave, and even more people engage. This directly causes the unstable code quality. On the other side the OSS is supported by people all over the world, therefore, the efficiency of bug finding and fixing may be higher as Compared with traditional software.

The Evolution and Maintenance of software is another important point to compare. Traditional in house development have systematic plan for system testing and maintenance. Whereas in Open source development Code quality is maintained largely by "massively parallel debugging"

(*i.e.,* many developers each using each other's code) rather than by systematic testing or other planned, prescriptive approaches. Although Open Software development encourages active participation of potential users but not pay enough attention on reflection and reorganization

It is hard to run an open source project following a more traditional software development method like the waterfall model, because in these traditional methods it is not allowed to go back to a previous phase. But Open source development is not software engineering done poorly. The various software engineering processes are at the center of OSP activity, and are heavily dependent on each other and processes from other areas. Ultimately, the degree of quality with which these processes are performed determines the resulting quality for the products of an OSP[11]. Requirements analysis, prototyping, testing, version and release management, bug triaging, deployment, and documentation make up software engineering in open source project. Alfonso Fuggetta [23] mentions that "rapid prototyping, incremental and evolutionary development, spiral lifecycle, rapid application development, and, recently, extreme programming and the agile software process can be equally applied to proprietary and open source software". Firstly if we find the analogy of OSS development with prototype we find that the requirements artifacts are not clearly defined in OSS project and typical OSSD processes start after the release of a software prototype as OSS and are just aimed to improve and maintain this prototype. Thus, an already existing software prototype seems to be a prerequisite for the requirements definition processes which typically occur in OSSD projects. The spiral model originally proposed by Boehm is an evolutionary software process model that couples the iterative nature of prototyping with the control and systematic aspects of the linear sequential model. In this model software is developed in a series of incremental releases. The OSS development process starts with early and frequent releases. Over time, both the Spiral Model and the Open Source Development Model continue to refine their respective projects by an iterative process[13]; however, the open source development also seems to adopt some parts of the incremental approach, in which essential features are grown methodically and are not released until they are properly finished.

Agile –methods represent new approaches in the spectrum of software development methods. The aim of these practitioner-oriented software development methods is to make a software development unit more responsive to changes. Juhani Warstaa and Pekka Abrahamsson [04] show that OSS and agile development methods have many similarities. An agile software development method has been defined with the following characteristics: incremental (small software releases, with rapid cycles), cooperative (customer and developers working constantly together with close communication), straightforward (the method itself is easy to learn and to modify, well documented), and adaptive (able to make last moment changes). It has been find by various researchers that the OSS development method is rather close to the definition of an agile software development method. Fuggetta[23] has mentioned one more open source development method that is the Agile method Extreme programming (XP). XP try improve classic waterfall model by trying to start coding as early as possible but without creating a full-fledged prototype as the first stage. Open source development is also starts with the development of source code by developer's personal itch. All the Agile methods are in essence applicable to open source software development, because of their iterative and incremental character. XP & Open Source Development share the same root. Both XP and most open source are rooted in minimization of planning, organization, testing, etc. rather, both tend to focus on maximizing time spent programming. Another Agile method, internet speed development, is also suitable for open source software development in particular because of the distributed development principle it adopts. Internet-Speed Development used geographically distributed teams to 'work around the clock'.

8. CONCLUSION:

Despite the fact that no standard development life cycle exists in open source development, OSSD is getting success as development methodology. Open source software progress has modified the method of software development, updating, and maintenance. OSS is becoming famous in the field of software development and million of people are getting the benefit from this type of development. Different views on development model of open source software are suggested by the various researchers. This paper has analyzed the nature and characteristics of the OSS development life cycle model and different views of researchers. It has been argued that OSS development violating the principles of software engineering. A theoretical study has also been made to find out the validity of this argument by comparing OSS development model with the software engineering development models as portrayed in the software engineering books. It can not be said that Open source is completely violating the software engineering principles. Consciously or unconsciously some principles of conventional development is followed in OSS development. This is particularly true in case of large OSS projects. To make the team work possible large successful projects do define and enforce some rules. In small OSS development projects with fewer developers, development process may not be well defined. But it can not be said that in OSS development projects software engineering is done poorly. It is instead a different approach to the development of software systems.

9. REFERENCES:

[01] Otso Kivekas, (2008), *Free/Open Source Software Development: Results and Research Methods,* Master's Thesis, University of Helsinki.

[02] Amit Deshpande, Dirk Richle, (2008), *The total growth of Open source,* Proceedings of the fourth conference on Open Source Systems (OSS 2008), Springer Verlag.

[03] Postnote on *Open Source Software*, Parliamentary office of Science and Technology, No. 242, June 2005 source internet, www.parliament.uk/parilamentary_office/post/pubs2005.cfm

[04] Juhani Warsta and Pekka Abrahamsson, *Is open source software development essentially an Agile method?*, Third workshop on open source software engineering, Portland, Oregon, USA.

[05] Sommerville, I., *Software Engineering*, 7th edition, Addison Wesley, New York 2004.

[06] Open source Initiatives, http://www.opensource.org/docs/osd.

[07] Feller J. and Fitzgerald B. (2000), A framework analysis of the open source software development paradigm. In *Proceedings of the twenty first international conference on Information systems', International Conference on Information Systems*, pages 58–69, Brisbane, Queens-land, Australia.

[08] Morten Sieker Andreasen, Henrik Villemann Nielsen, Simon Ormholt Schrøder, Jan Stage, (2006), *Usability in Open Source Software Development: Opinions and Practice,* Information Technology and Control, Vol. 35, No. 3 A

[09] Rinette Roets, Marylou Minnaar, and Kerry Wright, (2007) Open source: *Towards Successful Systems Development Projects in Developing Countries,* Proceedings of the 9th International Conference on Social implications of computers in developing countries, Sao Paulo, Brazil, May 2007.

[10] Eric S. Raymond, (1999), *The Cathedral and the Bazaar: Musingson Linux and Open Source by an Accidental Revolutionary*, O'Reilly & Associates.

[11] Gregor J. Rothfuss, (2002), *A Framework for Open Source Projects*, Master Thesis, University of Zurich, November.

[12] Roger S. Pressman, (1997), *Software Engineering A Practitioner's Approach, McGraw Hill, 4th Edition.*

[13] Daniel Blaney, Diana Lenceviciene, Ben Peterson, And Zijiang Yang, *Open Source Software Development Model,* source internet (scholar.google.com).

[14] Yi Wang, Defeng Guo EMOS/1: An Evolution Metrics Model for Open Source Software, source internet.

[15] Walt Scacchi, Joseph Feller et al, (2006), *Understanding Free/Open Source Software Development Process.*, Software Process Improvement and Practic;11:95-105

[16] S. Sharma, V. Sugumaran, and B. Rajgopalan, (2002), *A Framework for creating hybrid-open source software communities,* Information Systems Journal, vol. 12, pp.7-25.

[17] Ming-Wei Wu and Ying-Dar Lin., (2001), *Open source software development: an overview*. Computer, 34(6):33–38.

[18] Schweik, C. M., & Semenov, A., (2003), *The institutional design of open source programming*: *Implications for addressing complex public policy and management Problems,* source internet.

[19] Mockus, A., Fielding, R. T., & Herbsleb, J. D., (2000), *Two case studies of open source software development: Apache and Mozilla.* ACM Transactions on Software Engineering and Methodology,11(3), 309-346.

[20] Jorgensen, N., (2001), Putting it all in the trunk: Incremental software development in the Free BSD open source project. *Information Systems Journal, 11*(4), 321-336

[21] Stefan Dietze, (2005), *Agile Requirements Definition for Software Improvement and Maintenance in Open Source Software Development* **,** *Proceedings of SREP'05, Paris, France, August 29–30, 2005.*

[22] Kevin Crowston, Barbara Scozzi, (2002), *Exploring the Strengths and Limits of Open Source Software Engineering Processes: A Research Agenda*, 2nd Workshop on Open Source Software engineering, May 25, 2002, Orlando, Florida

[23] Fuggetta, A., (2003), *Open Source Software- An Evaluation*, Journal of System and Software, 66,77-90.

Authors' Short Biography: **http://www.rdunijbpin.org/ufaculty.htm**

Mr. Vinay Tiwari is qualified Computer professional having done PGDCA with distinction (1989) and MCA (2000). He has more than 19 years professional experience, 15 years of teaching experience at UG level and 09 years at P.G. level. He is a regular teaching counselor of Indira Gandhi National Open University for BCA/MCA courses from last 15 years and R.D. University Distance Education for last 5 years. He is a permanent resource person of Computer Refresher Courses organized by Academic Staff college for college teachers. His Area of interests are Computer Programming, Web Designing and Software Engineering. In the last 5 years he has attended 3 International and 5 National conferences organized at different places and presented research papers. His two books has already been published on computers