ISSN : 2229-4333(Print) | ISSN : 0976-8491(Online)

Software Engineering Issues in Development Models of Open Source Software

Vinay Tiwari
University Institute of Computer Science and Applications, R.D. Univ., Jabalpur, MP, India

Abstract
In the recent years open source software has seen spectacular success due to the development of high quality software. Open source software development has established itself to be an efficient and successful development methodology despite the fact that no standard development life cycle exists and also the development methodology of OSS violates many traditional software engineering principles. In order to better understand the software development different development models for open source software are suggested by various researchers during the past years. This paper study and analyzed some of the open source software development life cycle models. The purpose of the study is to summarize the current knowledge about OSS development process and find out the software engineering practices of the open source development environment. A comparison between software engineering techniques of open source development methodology and classical software development methodology and weakness will also be discussed.

Keywords
Open Source Software Development, Software development models, Software Development Life Cycle, Agile software development, extreme programming, OSS Software Engineering.

I. Introduction

Open source software during the recent years has attracted software users who want high quality software and cannot afford expensive commercial version. 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. Generically open source refers to a program in which the source code is available to the general public for use and/or modifications from its original design free of charge. Open source software as defined by the open source initiative, is "software that must be distributed under a license that guarantees the right to read, redistribute, modify and use the software freely" [1]. Open source software is developed by geographically distributed teams of volunteers with no central control and that could define what or how the developers implement. [2]. 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 [3]. The open source software development strategies are distinct from that of traditional software development and differs significantly from software engineering as it is described in text books. Open source has violated many of the theories of software engineering like limited team size, decentralized project management etc. [4]. Traditional closed management with their classic way of functioning may still look at the open source model as a big mystery. The open source software development has changed the way the software is perceived, developed and deployed [5]. The sudden success and major adoption of this new and innovative software development strategy has led to rethink and re-evaluate the studies and concepts of software engineering specially those which the open source violates. This paper discusses different development models of OSS and summarizes the current knowledge about OSS development process. A theoretical study is also being done to find out the software engineering practices specially requirement analysis/project planning, system design of the open source development environment. A comparison between software engineering techniques of open source development methodology and classical software development methodology and weakness will also be discussed.

II. Traditional Software Development Methodology Vs OSS Development Methodology

Traditional software development starts with detailed requirements document that is used by the system architect to specify the system. Next comes detailed system design, implementation, validation, verification, and ultimately maintenance/upgrade. Iteration is possible at any of these steps. On the other hand it is hard to run an open source project following a traditional software development method because in the traditional methods it is not allowed to go back to a previous phase. In open source software development requirements are rarely gathered before the start of the project; instead they are based on early releases of the software product. New projects also 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" [6]. Then he or she starts and discussion with his friends and colleagues about the possible solution and making the code

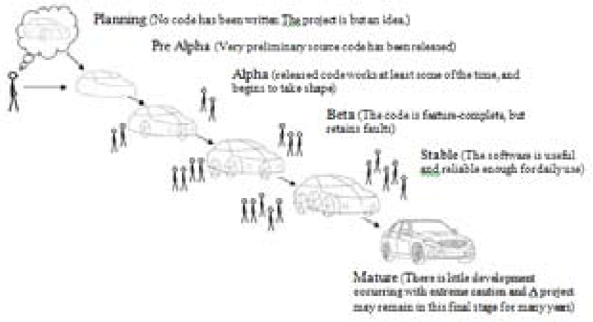
Planning (No code has been written The project is but an idea)
Pre Alpha (Very preliminary source code has been released)
Alpha (released code works at least some of the time, and begins to take shape)
Beta (The code is feature-complete, but retains faults)
Stable (The software is useful and reliable enough for daily use)
Mature (There is little development occurring with extreme caution and A project may remain in this final stage for many years)

Fig. 1: Stages in Open Source Software Development Process

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 starts 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. Gregor J. Rothfuss [7] classified the various stages of an OSP as Planning, Pre-Alpha, Alpha, Beta, Stable, Mature.

The Cathedral and the Bazaar [Eric S. Raymond, 1997] is the most frequently cited description of the open-source development methodology. In this book, Raymond makes the distinction between two kinds of software development. The first is the conventional closed source development. These kind of development methods are, according to Raymond, like the building of a cathedral; central planning, tight organization and one process from start to finish. The second is the progressive open source development, which 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."

III. Life Cycle Models of Traditional Software Development

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 is why it is usually implemented as a variant of prototyping model with the first iteration being a prototype [8].

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.

IV. 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 Open Source software development. 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 [1]. 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 [9]. 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 [10]. Sharma et al [11] 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:

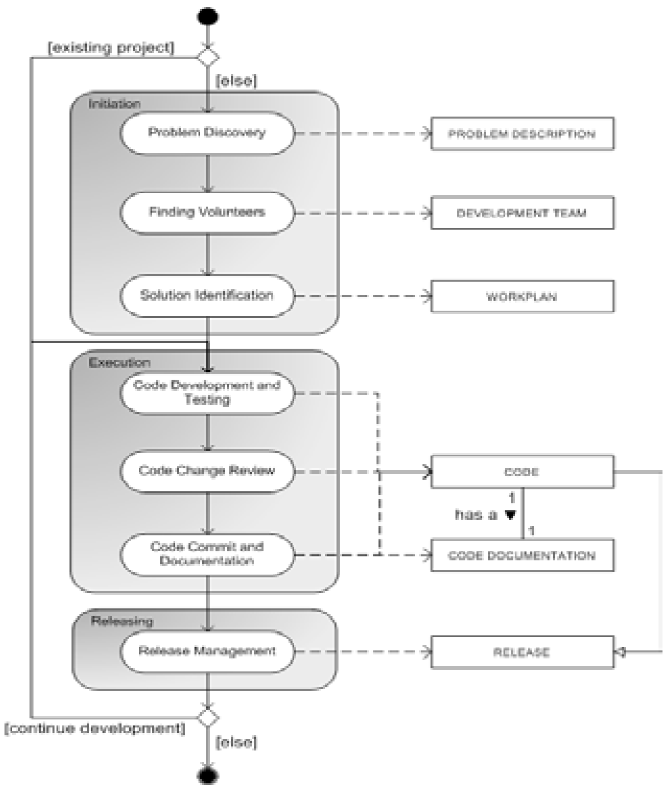

Fig. 2: OSS Development Process (source wikipedia)

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.

Woods, D. et al [12] suggested the open source development model for the enterprise as shown in Fig. 3. In this model each project begins with the initial determination of an idea or need which can originate from any one person or community. The next step in the process is the initial development for a proof of concept to determine the feasibility of the project, leading directly into the initial public prototype. The public prototype's development is the core infrastructure surrounding the project and consists of the initial steps at programming for the new software program. The intent of the public prototype is to assist in the creation of a community around the prototype with a clear understanding of the original project's proof of concept. Through time, the initial prototype is opened for review and contributions are made by others within the community. Each subsequent addition to the software program is released reflecting an incremental evolution in the product development. After this initial core development, the project will either become stagnant or will continue to evolve and mature. Stagnation or abandonment of open source projects may occur through a perception of completion for the project, poor project leadership which removes incentives for future contributors, or simply through lack of interest.

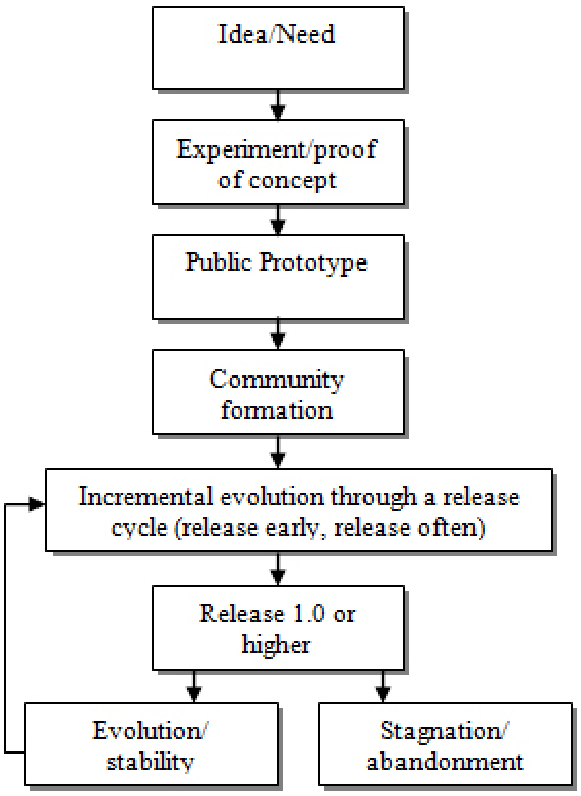

Fig. 3: Woods, D. (2005) Open Source for the Enterprise, p. 17, Fig. 1-1

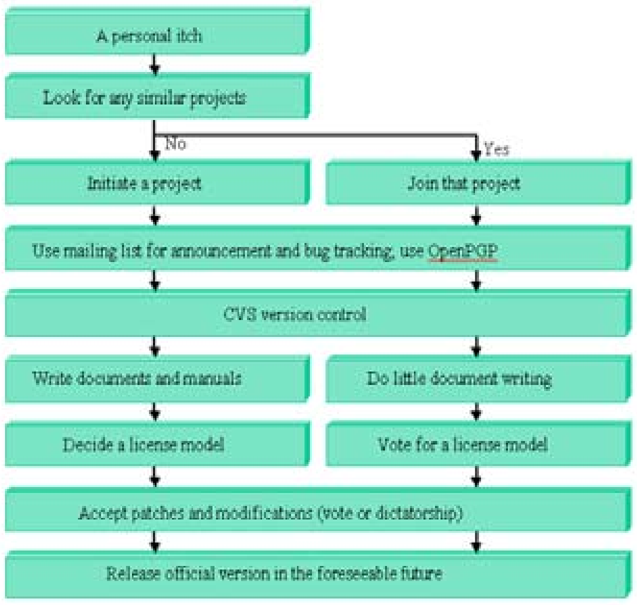
A personal itch
Look for any similar projects
No
Yes
Initiate a project
Join that project
Use mailing list for announcement and bug tracking, use OpenPGP
CVS version control
Write documents and manuals
Do little document writing
Decide a license model
Vote for a license model
Accept patches and modifications (vote or dictatorship)
Release official version in the foreseeable future

Fig. 4: Open source system development cycle
Source: (Wu and Lin, 2001, p.34)

Schweik and Semenov [13] 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 characterized 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.

Ming-Wei Wu and Ying-Dar Lin [14] proposed a development model for open source by incorporating the open source licensing and version control as shown in Fig. 4. 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.

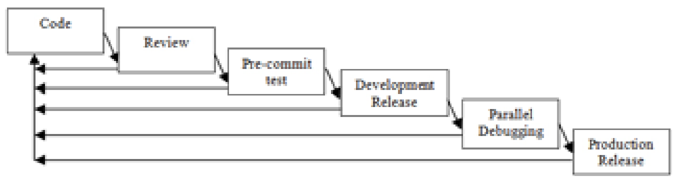

Fig. 5: Jorgensen Life cycle

Mockus et al [15] 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 [16] provides a more detailed description of specific product related activities that underpin the OSSD process. His model is shown in Fig. 5.

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.

Roets Minnar et al [17] expands on Jorgensen's life cycle model and incorporates aspects of previous models. Their model (Fig. 6) also attempts to encapsulate the phases of the traditional SDLC. In this model Jorgensen's code phase is replaced by generic initiation phase. This phase refers to developed code that is used as a prototype for further progress on a particular project. The initiation phase moves into a cycle of code review and further contribution. The number of iterations occurring at this phase. Once a piece of code is considered adequate for inclusion in a development release, pre-commit testing is performed to ensure that this new piece of code, once added, does not break the existing release. A process of debugging and reincorporation of code into the development release then takes place. This is again an iterative process occurring within the community web space. Eventually, code forms part of a production release which is generally managed by a core developer. Production releases take the form of a prototype that can be used in the initiation phase of the next iteration of that project, component or code segment.

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 [18].

- 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

V. Software Engineering: OSS vs. Traditional Software Development Models

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.

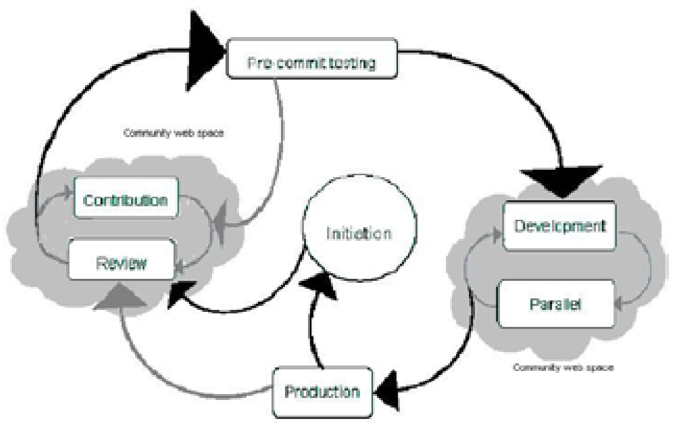

Fig. 6: Model of Roets, Minnaar, et al.

Traditional software development methods follows a software engineering principals is essentially a science about how software should be made not how it is made. The development model of OSS 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.

The software engineering tasks include analysis of the system requirements, Design, development, implementation of the software, and testing the software to verify that it satisfies the specified requirements. In traditional software development as discussed earlier, these tasks comprises in four broad phases: planning, analysis, design, and implementation. In Open Source Software development, these stages tended to be conFig.d differently. The first three phases of planning, analysis, and design are concatenated and performed typically by a single developer or small core group [19]. The planning phase is probably best summarized by Raymond's phrase of a single developer perceiving "an itch worth scratching." This leads to construction of an initial prototype. 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[20]. Design decisions also tended to be made in advance before the larger pool of developers starts to contribute. Systems are highly modularized to allow distribution of work and reduce the learning curve for new developers to participate (they can focus on particular subsystems without needing to consider the system in its totality). The implementation phase in the Open Source Software development life cycle consists of several subphases as suggested by Feller and Fitzgerald (2002), which include Code submission, Review, Pre-commit test, Development release, Parallel debugging (the so-called Linus' Law "given enough eyeballs, every bug is shallow") and finally a Production release. 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.

Roets Minnar et al also compared their model in a similar manner to the traditional SDLC as shown in Fig. 7. Planning, analysis and design phases are largely undertaken by the initial project founder.

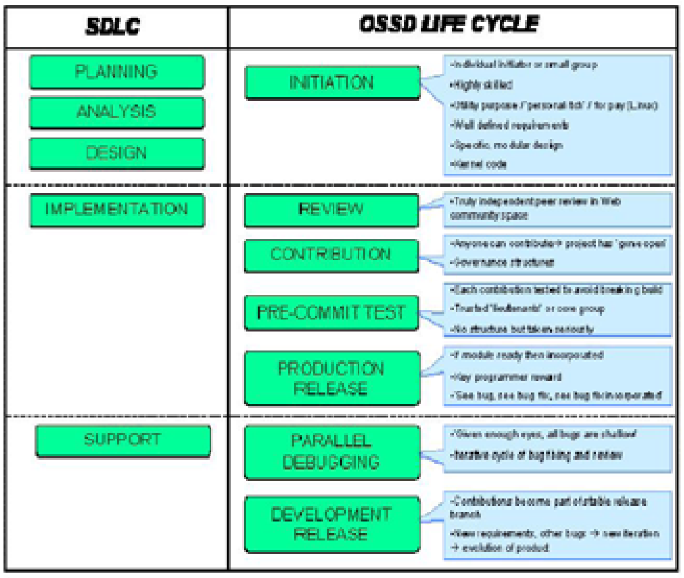
SDLC
OSSD LIFE CYCLE
PLANNING
ANALYSIS
DESIGN
IMPLEMENTATION
SUPPORT
INITIATION
-Individual initiator or small group
-Highly skilled
-Utility purpose / 'personal itch' / for pay (Linux)
-Well defined requirements
-Specific modular design
-Kernel code
REVIEW
-Truly independent peer review in Web community space
CONTRIBUTION
-Anyone can contribute → project has 'gone open'
-Governance structure
PRE-COMMIT TEST
-Each contribution tested to avoid breaking build
-Trusted 'lieutenants' or core group
-No structure but taken seriously
PRODUCTION RELEASE
-If module ready then incorporated
-Key programmer reward
-See bug, see bug fix, see bug fix incorporated
PARALLEL DEBUGGING
-Given enough eyes, all bugs are shallow
-Iterative cycle of bug fixing and review
DEVELOPMENT RELEASE
-Contributions become part of stable release branch
-New requirements, other bugs → new iteration → evolution of product

Fig. 7: comparison of OSSD life cycle with SDLC

It may be concluded that open source development is not software engineering done poorly. The various software engineering processes are at the center of open source development 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 open source. Requirements analysis, prototyping, testing, version and release management, bug triaging, deployment, and documentation make up software engineering in open source project. Alfonso Fuggetta [21] 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[22]; 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. Moreno Muffatto[23] highlighted the Similarities between various software development models and the open source development process and find that open source community has taken advantage of the strong points of existing development process models. The development of open source software is based on the community of voluntary and independent software developers. A lot of importance is given to human resources, i.e. each programmer's knowledge and skills, just as in the Build and Fix model. End-users are involved in the development process as well by providing important feedback. This involvement means that particular attention is given to user requirements, as is the case in the Waterfall Model and all of those that followed. The development community then uses feedback to more accurately define product specifications and produce frequent releases, as is the case in the Prototyping Model. Large-scale open source projects are managed using modularity. The overall project is carefully divided into subprojects and the various functions and features are developed in an incremental (Iterative Model) and concurrent (Evolutionary Model) way. The parallel development of different modules (Evolutionary Model) also speeds up the development process. Overall, we can see that the open source community has taken advantage of the strong points of existing development process models. The way software is developed in this community is becoming more and more similar to the way it is developed using the Spiral Model.

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 close to the definition of an agile software development method as OSS projects are iteratively developed, incrementally released, reviewed and refined by software development peers in an ongoing agile manner. Fuggetta [21] 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.

VI. Conclusions

Open source software development methodology in recent years is emerging as an alternative approach for the development of software projects despite that fact that no mature development methodology exists. There are many theoretical approaches that try to explain the phenomenon of open source development. This paper summarizes the nature, characteristics and current knowledge about OSS development process also compares and examines software engineering practices followed in open source development. Different development models of Open Source software has been discussed and theoretical study has been made to find out the software engineering practices in OSS development by comparing it with the traditional software engineering development models. It has been observed that OSS development process not completely violating the software engineering principles. This is due to the fact that most of the OSS developers are the IT professionals having knowledge of software engineering principles. Their hidden knowledge of software engineering enforce them to apply consciously or unconsciously the traditional software engineering principles in OSS development. This is particularly true in case of large OSS projects where the team work is possible only after defining and enforcing some rules. Analysis shows that Agile software development practices are closely aligned to OSSD practices and OSS approaches can be seen as one variant of the multifaceted agile methods[04].

VII. References

[1] Open source Initiatives, [Online] Available : http://www.opensource.org/docs/osd.

[2] Kirk St. Amant , Brian still (edited), "Handbook of Research on Open source Software Technological, Economic and Social perspectives", Pub Information Science References, USA (2007).

[3] Tiwari Vinay, "Some Observations on Open Source Software development on Software Engineering perspective", International Journal of Computer Science and Information Technology vol. 2 No. 6 pp. 113-125 (2010).

[4] Juhani Warsta, Pekka Abrahamsson, "Is open source software development essentially an Agile method?", Third workshop on open source software engineering, Portland, Oregon, USA.(2003)

[5] Vixie, P., "Software Engineering, open sources: voices from the open source revolution", Sebastopol, California: O'Reilly and associates. (1999).

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

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

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

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

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

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

[12] Woods, D., Guliani, G., "Open Source for the Enterprise Sebastopol", CA: O'Reilly Media, Inc. (2005)

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

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

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

[16] Jorgensen, N., "Putting it all in the trunk: Incremental software development in the FreeBSDopen source project". Information Systems Journal, 11(4), 321-336,(2001).

[17] Rinette Roets, Marylou Minnaar, Kerry Wright, "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, (2007).

[18] Stefan Dietze, "Agile Requirements Definition for Software Improvement and Maintenance in Open Source Software

Development", Proceedings of SREP'05, Paris, France, August 29–30, (2005)
[19] Brian Fitzgerald, "The Transformation of Open Source Software", MIS Quarterly Vol. 30 No. 3, pp. 587-598, (2006)
[20] Kevin Crowston, Barbara Scozzi, "Exploring the Strengths and Limits of Open Source Software Engineering Processes: A Research Agenda", 2nd Workshop on Open Source Software engineering, , Orlando, Florida, May 25, 2002
[21] Fuggetta, A., "Open Source Software- An Evaluation, Journal of System and Software", 66,77-90, (2003).
[22] Daniel Blaney, Diana Lenceviciene, Ben Peterson, Zijiang Yang, "Open Source Software Development Model", source internet (scholar.google.com).
[23] Moreno Muffatto, "OPEN SOURCE A Multidisciplinary Approach", Imperial college press,(2006).

Mr. Vinay Tiwari is qualified computer professional having done PGDCA with distinction (1989) and Master in Computer Applications (2000). Currently pursuing his Ph.D. Degree from R.D. University, Jabalpur. He has more than 19 years professional experience, 15 years of teaching experience at UG level and 10 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 7 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 4 International and 5 National conferences organized at different places and presented research papers. His two books have already been published on computers.