A Review of Open Source Software Development Life Cycle Models

Munish Saini and Kuljit Kaur

Computer Science Dept., Guru Nanak Dev University, Amritsar, 143005, India
munish_1_saini@yahoo.co.in, KuljitChahal@yahoo.com

Abstract

The Life cycle for the development of traditional commercial software is well established and discussed in various texts and research papers in detail. But in case of Open Source Software (OSS) life cycle for the development is not being discussed in much detail as there is no standardized life cycle approach exists for Open Source Software (OSS) development. Different researchers and developers have proposed various life cycles for the development of OSS with respect to their own development experience, need, or application. The main focus of this paper is on reviewing and then comparing the existing Open Source Software Life Cycle (OSSLC) presented and proposed by the various researchers and practitioners. It will give a precise view that in actual what are the various development approaches are used in case of OSS and how development of open source software starts and proceeds.

Keywords: Open Source Software; Open Source Software Life Cycle, Software development Life Cycle (SDLC)

1. Introduction

Open Source Software [1] development is not a new concept. The term open source was introduced in 1998. The Open Source Initiative (OSI) [2] advocate the open source movement and provide the guidelines for issuing of open source license [3]. OSS is defined [10] as the software whose source code is available along with the software and user has the freedom to run, copy, distribute, study, change and improve the software under the licensing policies [4] of OSS. The development of OSS gains popularity due to the wide availability of the internet facility to each and every region of the world, parallel development, peer review, parallel debugging, expert developers, and feedback. Many different developers, user, or co-developers can participate in the development of the OSS. The development of OSS is always initiated by single developer or a single group, who starts the development of software for its own “personal itch” [5]. The development of OSS is different from the traditional commercial software also called as “closed software”. It is found by various researchers [7, 8] that the traditional Software Development Life Cycle (SDLC) [6] and development processes models [6] cannot be used for the development of OSS. Various researchers and practitioners are working on developing the standard development life cycle of OSS.

The rest of the paper consists of following sections and depicts the flow of information as follows. In Section II traditional SDLC is presented and discussed; various issues about its incompatibility with the development of OSS are depicted. Section III will present the review on the current research work of various researchers about the development of open source software life cycle model. It will give precise view of the various OSSL C and its development process model proposed and discussed by different researchers in their respective research papers. In Section IV, comparison about the various development life cycles and development model is specified. Followed by Section V, in which the discussion
of various life cycle models is presented. Finally, Section VI will give conclusion about the OSSLC along with the future work.

2. Description of Software Development Life Cycle

Software Development Life Cycle (SDLC) [6] describes the methodology by which the development of any software takes place. Before SDLC the process of developing the software was taken as informal activity with no formal rules and standards. This may lead to the various problems such as delay in development, cost overrun, and low software quality. The introduction of the SDLC gives the precise standard and the steps for the development of the software. SDLC overcome all the problems which are there before the introduction of the SDLC.

SDLC composed of various steps or phases which are show below in Figure1 and described further.

Feasibility Analysis: In this phase the feasibility of the project is being evaluated in terms of input data, output data, processing required to transform input to the required output, cost befit analysis, and schedule of the project. It also includes the technical feasibility in terms of software, hardware, and skilled persons. At the end of the feasibility study the feasibility report is generated.

Requirement Analysis and Specification: Various requirements for the software is being gathered and analyzed in this phase. At the end of this phase developers is known with the various requirements of the software. At the end of this phase Software Requirement Specification (SRS) document is being generated.

Designing: The Translation of the requirement specified in SRS to the logical structure is done in this phase. The Design document is produced as an output of this phase.

Coding: Actual programming is done in this phase. The design document is translated to the source code which is in executable form.

Testing: The source code generated in previous phase is being tested by using various test cases. Various types of testing techniques are used to evaluate the correctness and validness of the software.

Implementation: The developed software is implemented at the user site and software is made available to the user for actual use. User uses the software and if any error or problem is found it is reported to the system analyst.

Maintenance: Changes if any after the implementation is incurred into the software. This phase is responsible for the post implementation and maintenance of the software for its proper working.
Figure 1. Phases of Software Development Life Cycle (SDLC)

The traditional closed software can be developed by using the various classical conventional development models such as waterfall model, iterative model, prototype model, spiral model etc [6-10]. The development of OSS cannot be done by using these conventional development life cycles models because of these following reasons.

1) Requirements may not be properly structured at the start of the development process.
2) Number of users, co-developer, and developer participating in the development process may be large and more over different users may geographically locate at different location [15].
3) No proper specification and documentations are available for OSS development.
4) OSS development process may not consider all the phases of traditional development process of closed software [8].
5) Difference in motives of development of OSS and closed software.
6) OSS requires collaborative team structure for its development.
7) The way of OSS evolution is different, as OSS allows changes to be incorporated.
8) Development of OSS is in open environment.
9) In OSS development the main focus is on coding and implementation.
10) No clear design process [15].
11) No explicit list of deliverables, schedule and project plan.
12) User can choose work of their own choice in OSS [16].

The recent software development models such as Agile [11-14], Extreme programming (XP) variation of Agile model, Evolutionary prototype models used for the development of the traditional closed software’s can be seen as one of the methods which will specify the various steps for the development life cycle of the OSS. In the next section Agile model for the development is also presented as the one of the life cycle development approach for open source software.
3. Review of Existing OSS Development Life Cycle Models

There are many approaches which try to explain the development of the OSS, but still no general approach is being agreed. In this section, review about the various life cycles and development models proposed by different researchers are being presented and discussed.

*Andrea Capiluppi and Martin Michlmayr* [18] proposed an OSS development life cycle. The researcher considered that the complementary events i.e. Cathedral (closed style) and Bazaar (open style) (by Eric S. Raymond [5, 17] in his book “The Cathedral and The Bazaar”) are considered to occur in the same OSS project. The paper has shown that the many successful OSS project consist both of these development style while their development. The proposed development life cycle model for OSS consists of mainly three phases.

1) Cathedral Phase
2) Transition Phase
3) Bazaar Phase

The development of the OSS is assumed to start with in Cathedral phase where cathedral building development style is followed. In this small group of individual developer initiate the development in the closed environment with no outside audience or participant. This phase consist the project author, core developer, and project manager. The cathedral phase follows UNIX kind of philosophy for its development. Once the software developer gained confidence in the development they publish their work on internet. This change phase is called as transition phase. Transition phase acts as link between cathedral and bazaar phases. Transition phase is initiated only when the design is stable and modular, original author lose interest in the Individual development process and a prototype of software are ready for use. In Bazaar phase the bazaar development style is followed, where the development actually proceed in the distributed environment. The interested people join the development process and perform development, maintenance, bug reporting, bug fixing *etc.*, tasks. The development of the OSS is done in collaborative manner. The author considered two case studies *Mine and Arla* to find check the applicability of the proposed life cycle and to check why Wine is so successful and very much alive with the continuous development process as compare to Arla. The various phases of the development model proposed by [18] are shown in Figure 2.

![Figure 2. OSS Development Lifecycle by Andrea Capiluppi and Martin Michlmayr [18]](image-url)

*Vixie* [16] proposed a comparative model for the development of OSS. Vixie suggested that the development life cycle of the OSS can include all the elements of the classical SDLC used for the development of the closed or commerical softwares. The reseracher in his work
has correlated his theory by considering the development process of classical old OSS such as BSD, SendMail, and Bind which proves the fact that these OSS includes standard software engineering development phases of traditional SDLC i.e., Analysis, Design, Coding, Implementation, and Support. Vixie suggested that in the analysis phase, requirements are based on the need of the developers who develop the OSS. The final requirements are negotiated on the internet. The design phase in the development of OSS is not performed formally. Design issues are not written anywhere, and not made visible to the every one because it may limit the development process. Less emphasis is done on the design phase. Implementation and coding are the main concerns of the development process. Various users participate in the development of OSS. They communicate with each other informally and send change request, feedbacks, developed code. Testing of OSS is considered to be more powerful as compare to the testing of traditional software because the number of the user available to test the OSS is very large as compare to the traditional software testing. These tester or user sends there feedbacks in the form of reviews, bug report, change request. The testing which is performed in the OSS development is unstructured. The disadvantages of unstructured testing are covered by having the “uncounted stranger” [16] that tests the OSS. Vixie study presents all the differences that OSS life cycle has as compare to the SDLC, but fails to suggest an appropriate model that analyses this new process [16].

Schweik & Semenov [19] proposed an OSS development life cycle. It is a kind of organizational method which consists of these three phases. Each phase consist of different set of activities.

1) Project Initiation
2) Going “Open”
3) Project Growth, Stability or Decline

1) Project initiation phase activities
   a) The project is initiated and composed of all the initial stages of Vixie’s model.
   b) The development of OSS is initiated by the single or the group of developer, which form the core developer of the OSS development.
   c) The modular approach is followed for the development of the OSS, the modular approach provides various benefits as pointed by [19, 20].

2) Going open phase activities
   a) The project founder have choice to choose the OSS license such that it ensure that future development of the OSS take place in good manner and ensure that sufficient amount of the original requirements are solved.
   b) Technologies and websites are to be chosen as the way to share code and develop development community.
   c) Issues like version management, problem tracking are also considered
   d) Developing operational design to provide some kind of project management hierarchies.

3) Project growth, stability or decline phase activities
   a) Decide on issues related to the growth of OSS, weather it is growing or decline
b) Will the project is designed in such a way that it can attract most of the developers and global users to use the software and participate in the future development, testing, documentation.

Wynn [21] proposed the similar kind of open source life cycle with slight variation. The research proposed that the life cycle of OSS consist all the phases of the Schweik & Semnov proposed model [19] and one more phase i.e. the maturity phase. The various phase of Wynn proposed life cycle model are

1) Project Initiation
2) Going “Open”
3) Project Growth, Stability or Decline
4) Maturity Phase

First three phases are same and perform same activities as in Schweik & Semnov life cycle model [19]. The maturity phase specifies that the OSS gained a critical mass in terms of number of users, developers. The OSS reached the stage where it is said to be matured enough due to its popularity, less bug rate, less changes, working accurately most of the time. The life cycle proposed by [19, 21] consider managerial aspects and organizational structure of OSS but do not provide task related analysis of OSS development.

Mockus, A., Fielding, R. T., & Herbsleb, J. D [15] proposed another life cycle model. The model which is proposed by [22] consist decision making framework and task related project phases. The proposed life cycle consist these phases

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

The proposed model consider managerial activites (developer management and the work to be done), but not product-related activities. The proposed model considered more task-related issues than the model proposed by Schweik and Semenov [19], Wynn [21]. The researcher assumed that some kind of prototype is already exists for the software. The proposed model at some level provide support for all the phases of traditional SDLC. In this life cycle model the main emphasis is on the planning phase of the SDLC other phases design and analysis are not considered adequately.

The United States Department of Defense (DOD) proposed a life cycle model [22] for the development of the OSS. The main focus of this development life cycle is on the contributor roles who participate in the development process of the OSS. The OSS development model presented by [22] is shown in Figure 3.

The OSS development life cycle consist of following:

**Developer:** Developers consist of the persons who initiated and contributed majorly in the development of the OSS. The developer is responsible for the actual working of the OSS in right and proper manner.
Trusted Developer: Trusted developers consist of the persons who contribute in the development of the OSS continuously and they gained the trust of the initiators through their countious involvement in the development process and became the part of the core developer community. Trusted developers are allowed to make updation and changes in the trusted repository directly. The entire user, distributor sends their request for change and updation to the trusted developer.

Trusted Repository: Repository means the data house. The repository in OSS development specifies the house from where all the information related to the OSS can be retrieved. The user and trusted developers can access the repository directly or through the distributor. The trusted repository specify the space from where user or trusted developer can get the official version of the software and get other related information such as bugs report, change log, documentation etc.

Distributor: Distributor is the persons who have the copy of developed OSS and they are using it and perform other task such as modify, integrate, testing, configuration etc.

User: User is the normal person who uses the OSS. User can be categorised as Passive and Active user. The Passive users are those who download the software for use, study they never participate in development. The Active user participate in the development proceess by performing task as finding bug, giving review about the OSS etc.

In this development process the flow of the source code is shown as developer - trusted developer - trusted repository - distributor - user i.e., it follows the top-down approach, where as feedback/bug report use bottom-up approach. It flows from user - distributor - trusted repository - trusted developer - developer.

Figure 3. OSS Development Model Proposed by DoD, USA [22]

Ming-Wei Wu and Ying-Dar Lin [23] proposed a life cycle model for the development of OSS. The proposed model consist Open source licensing and version control. The model is shown below in Figure 4. The proposed life cycle for the development of OSS is considered to start with the “personal itch” [5] to the individual or group of persons. The initiator look for the software which can solve its personal needs. If no existing project is found then initiation of the new project take place, otherwise the initaor will join the existing project. In both the cases mailing lists, bulletin boards, CVS version control are used for communication and controlling the development of the software. User can also participate in writing of documentation, generating patches to fix bugs, vote or chose for the OSS license, request changes, modifications, accept patches for changes and modification. Once the software gone through all these, software official version is released and process continue until software gain certain maturity.
Another Life cycle is proposed by M. Ab-bing [24]. The proposed life cycle consist of three phases; Initiation, Execution, and Releasing. All these phases include various activities. The proposed model along with the activities it performs is show in Figure 5. The initial phase of the proposed model is Initiation. In this phase if the new software is to be initiated then activities like problem discovery, finding the volunteers, and solution identification are performed. The various derivable of these activities are problem description, development team, and work plan. Work plan specifies how the problem is to be solved. The next phase of this life cycle model is Execution. In this phase actual development of the OSS takes place. Various activities which are performed in this phase are; code development and testing, code change and review, and code commit and documentation. The derivable of this phase are developed code and documentation. In this phase software like CVS, SVN are used for implementing the management. Releasing phase is the final phase of this model. In this the all the release management activities are performed and released software form the derivable of this phase. Version controlling software’s are used at this phase to control the release of the software.

Figure 4. OSS Life Cycle by Wu and Lin [23]
Jorgensen [25] provides another life cycle model which refines the Mockus, A., Fielding, R. T., & Herbsleb, J. D [15] life cycle model. The proposed model describes the product or task related activities. The model is show in Figure 6 and it presents the life cycle for changes that occurred within the FreeBSD project. The Jorgensen model is widely accepted as a frame work for the OSS development.

Various phases or activities which are performed are as follows

**Code:** Coding of the software is done in this phase. At the start of the software only person or individual group of person start coding. In OSS, the code is developed by well talented developers and code is made available for review and improvement.

**Review:** In This phase the developed code is reviewed. The independent peer review is the strength of this process.

**Pre-Commit test:** The reviewed code is than passed through unstructured testing phase. The developed code is tested to find errors. The commit operation is performed on the code which is found necessary and accurate, un-necessary code is rejected. This phase is considered most important for the development process because if not performed properly it may lead to failure of the OSS.

**Development Release:** After pre-commit test if the software is ready for development release it is done.

![OSS Development Life Cycle](image-url)
Parallel Debugging: Once the development is released the code is exposed to large number of contributors or user. They perform rigorous debugging to find all the bugs and report these bug to the core developers.

Production Release: Stable development release is then released as the Production version.

![Figure 6. Jorgensen Life Cycle for OSS [25]](image)

The development of OSS perform all these activities in a cycle i.e., the development proceed by performing these activities again and again until the maturity level of the OSS is not reached. It is also argued that there is no real end point for OSS development.

Rinette Roets, Marylou Minnaar, and Kerry Wright [26] proposed a life cycle model which expands the Jorgensen [25] life cycle model and include the aspects of previous models specifically Schweik and Semenov model. The various phases of the Rinette Roets et al are shown below in Figure 7. The researcher model encapsulates the classical SDLC model phases and provides comparision with SDLC which is shown in Figure 8

![Figure 7. OSS Life Cycle Model of Rinette Roets, Marylou Minnaar, and Kerry Wright [26]](image)
Figure 8. Comparison of SDLC with OSS Development Life Cycle by Rinette Roets, Marylou Minnaar, and Kerry Wright [26]

The model proposed by Rinette Roets, Marylou Minnaar, and Kerry Wright [26] replaces the initial code phase of Jorgensen model [25] with Initiation phase. In initiation phase can be applied to any level of project. This includes code development which is either done by the founder of the project or by the contributor of the project. The initiation phases the next move to the cycle of review and contribution. In this various developer can contribute and review the code. Next it moves to the Pre-commit test phase where the informal testing of the software is performed. It then point to the development and parallel debugging cycle. In this the developed code is made ready for development release and parallel debugging is performed on the released project by the various developers and the users. The last phase is the Production phase; in this if the developed released code is found accurate and stable, the production release is done. The production released software can further move to Initiation phase for further development. The OSS development process is assumed to follow this cycle and perform all these activities again and again until the OSS gain Maturity. It is also argued that the OSS development can never reach to end it is an ongoing and continuous process.

Scacchi’s [27] life cycle consist of various activities which form the phases of the model are shown below in Figure 9. Scacchi model consider the cyclic nature of the development process, with centralized role of experience sharing. All the phases are centrally managed by experience. The various activities which are performed are Assert requirement design, Develop OSS code, and Manage configuration, Download and install, End user, and Communicate experience.

Gilliam’s [28] proposed the OSS development life cycle. The proposed model is considered to have the cyclic nature. It repeats the various phases which are shown in Figure 10. Figure also depicts various activities such as make initial release available on the internet, find bugs, add features, contribute bug fixing, incorporating best features and patches, distribute the new releases which are performed in various phases of the model.
Figure 9. OSS Development Life Cycle Process by Scacchi’s [27]

Figure 10. Gilliam’s Model of OSS Development [28]

Figure 11. Eclipse OSS Development Life Cycle [29]
Eclipse [29] presents the development life cycle of various OSS which are built on Eclipse shown in Figure 11. Eclipse is an open-source community focused on developing a universal platform of frameworks and exemplary tools that make it easy and cost-effective to build and deploy software [30]. The various phases which are considered by the Eclipse platform are Pre-proposal, Proposal, Incubation, Mature, Top - Level, and Archived. The development process is made to continue under the various review process such as creation review, graduation review, promotion review, continuation review, and termination review. The OSS project is made to move to the next stage only when it passes the corresponding stage review process. The development of the OSS at Eclipse is managed by the hierarchy of people.

4. Comparison of various OSS Life Cycle Models

The comparison of various life cycle models proposed by different researchers and practitioners are shown below in Table 1 of Appendix.

5. Discussion

The development of open source software is accomplished by following various life cycle models. Some of the best known models used for the development of open source software are specified in this article. Apart from these specified life cycle models there are many other models (like Agile, IBM model), but we are not considering these models as they somewhat follow or refer the same steps and same approach as followed by these referred models. It is found that the development life cycle of open source software is very much different from that of traditional closed or commercial software. The most of traditional life cycle models are not efficient to be used for the development of open source software as the most of these follow the static development approach. The traditional models like Agile and Extreme programming can be used for the development of open source software by introducing the certain enhancements. Table 1 in Appendix 1 specifies the comparison of different life cycle models (proposed by different researchers for the development of open source software) along with their advantages and disadvantages. These models found to be differing by the methodology, number of phases and the feature they provide. Each research proposed their models on the basis of their own requirements and specifications; there is no standard and single quality approach found existing which can be found applicable for development of all open source software.

5. Conclusion and Future Work

OSS development has gained so much popularity due the various features it provides. Many successful OSS developments have attracted a large portion of the researchers for their own personal interest and to earn geek fame. The development of OSS is at peak right now. But the standardized life cycle for the development is not published yet as the SDLC for traditional software exists. In this paper an attempt is made to review the various OSS life cycles proposed by various researchers in various research articles. This paper gives a complete and compact study about the various existing OSS life cycle. The comparison of various OSSLC models provides the better understanding about all these proposed and existing OSSLC models. It is found that there donot exist any standardised life cycle model existing which can be used for development of all open source software. In future, a lot of work can be done in generalization and standardization of open source software development life cycle models and its approaches.
Acknowledgements

This research work is done under the UGC sanctioned research project entitled “Tracking open source evolution for the characterization of its evolutionary behaviour”. We acknowledge the UGC for giving the grant for doing the research work.

References


Appendix

Table 1. Comparison between various OSS Life Cycle Models

<table>
<thead>
<tr>
<th>Life cycle by</th>
<th>Phases</th>
<th>Methodology</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vixie</td>
<td>1.Analysis  2.Design  3.Coding  4.Implementation  5. Support</td>
<td>1.Consider all the phases of SDLC  2.But the main focus is on coding and implementation  3.Debugging and testing is performed by large group of uncounted strangers</td>
<td>1.Comparative Model  2.Presents all the differences that OSS life cycle has as compare to the SDLC</td>
<td>1.Fails to suggest an appropriate model that analyses this new process</td>
</tr>
<tr>
<td>Schweik &amp; Semenov</td>
<td>1.Project Initiation  2.Going “Open”  3.Project Growth, Stability or Decline</td>
<td>1. The project is initiated and composed of all the initial stages of Vixie’s model.  2. The project is made open to the outside world by sharing or publishing the work on internet.  3. Project growth is accessed and categorized to growing, stable. Or decline</td>
<td>1.Organizational Model  2. The modular approach is followed for the development of the OSS  3. Choice to choose the OSS license, technologies, websites version management, problem tracking are also considered  4. Each phase is characterized by different set of activities</td>
<td>1.Do not considered the task-related analysis of OSSD process</td>
</tr>
</tbody>
</table>
| Wynn | 1. Project Initiation  
2. Going “Open”  
3. Project Growth, Stability or Decline  
4. Maturity | 1. Consider all the phases of Schweik & Semenov life cycle and introduces a new phase called as maturity phase  
2. In maturity phase a project reaches critical mass in terms of the numbers of users and developers it can support. | 1. Organizational Model  
2. Consider managerial aspects and organizational structure of OSS  
3. Variation of Schweik & Semenov Model | 1. Do not considered the task-related analysis of OSSD process |
|---|---|---|---|---|
| Mockus et al | 1. Roles and Responsibilities  
2. Identifying work to be done  
3. Assigning and performing development work  
4. Pre-release testing  
5. Inspections  
6. Managing releases | 1. In this model all the activities and phases which are defined has main focus on the managerial activities  
2. Prototype of the software is considered to go through all these phases. | 1. Consider decision making framework and task related project phases.  
2. Consider managerial activities  
3. Considered more task-related issues than the model proposed by Schweik and Semenov Wynn  
4. More emphasis is on planning phase of life cycle | 1. Do not considered product-related activities  
2. Prototype of software should exist before  
3. Consider planning phase of SDLC but less emphasis is on other phases of life cycle.  
4. Fail to explain where design and analysis take place in this model |
| DOD | 1. Developers  
2. Trusted developers  
3. Trusted Repository  
4. Distributor  
5. User | 1. The main focus of this development life cycle is on the contributor roles who participate in the development process of the OSS | 1. Specifies the flow of source code and it is in top-down manner  
2. The bug report/feedback follows bottom-up approach | 1. Does not specify how the OSS is going to develop. |
| Wu & Lin | 1. Personal Itch  
2. Look for similar project  
3. Initiate  
4. Version Control  
5. Documentation  
6. Decide OSS license  
7. Patch generation  
8. Releasing | 1. Model specifies the process how the generation of the OSS take place by performing all these activities along with generation of licensing and version controlling. | 1. Consider open source licensing and version control.  
2. Participation in the development is open for anyone  
3. Free, copy left, and GPL are the three categories of the open source licensing considered in this model | 1. Does not specify how and where designing, testing, and planning take place |
| M. Abbing | 1. Initiation  
2. Execution | 1. Consider the development of each OSS go through all these phases. | 1. Generalized view of development process | 1. Does not specify the process and method how these |
<table>
<thead>
<tr>
<th>Model</th>
<th>Activities</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jorgensen</td>
<td>1. Code, 2. Review, 3. Pre-Commit Test, 4. Development, 5. Release, 6. Parallel Debugging, 7. Production Release</td>
<td>1. Model presents the life cycle for changes that occurred within the FreeBSD project. 2. Perform all these activities in a cycle i.e. the development precede by performing these activities again and again until the maturity level of the OSS is not reached.</td>
</tr>
<tr>
<td>Rinette Roets et al</td>
<td>1. Initiation, 2. Contribution and Review, 3. Pre-Commit Test, 4. Development and Parallel Debugging, 5. Production</td>
<td>1. All these phases are performed repeatedly until OSS again Maturity 2. Expands the Jorgensen life cycle model and include the aspects of previous models specifically Schweik and Semenov model. 3. Encapsulate the classical SDLC model phases and provide comparison with SDLC</td>
</tr>
<tr>
<td>Gilliam’s</td>
<td>1. Project Management, 1. Activities such as make initial release available on the internet, 1. Cyclic nature</td>
<td>1. Do not specify the design, analysis, planning phases of the development life cycle.</td>
</tr>
</tbody>
</table>

activities and phases are going to be performed and managed.
<table>
<thead>
<tr>
<th>Eclipse</th>
<th>1. Pre-proposal</th>
<th>1. Various phases which are used for development of OSS are performed repeatedly.</th>
<th>1. Projects are managed by a hierarchy named the Project Leadership Chain, consisting of the Project Management Committee (PMC), the project leaders of the Top-Level projects and regular project leaders.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. Proposal</td>
<td>2. Specifies the OSS development method followed by Eclipse platform in development of various well established OSS by eclipse</td>
<td>2. Meritocracy feature of development</td>
</tr>
<tr>
<td></td>
<td>3. Incubation</td>
<td></td>
<td>3. Various type of reviews are performed</td>
</tr>
<tr>
<td></td>
<td>4. Mature</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. Top - Level</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6. Archived</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 2. Development Team | find bugs, add features, contribute bug fixing, incorporating best features and patches, distribute the new releases which are performed in various phases of the model. | 2. Version Control 3. Parallel Debugging | planning phases of the development life cycle. |

1. Various phases which are used for development of OSS are performed repeatedly.
2. Specifies the OSS development method followed by Eclipse platform in development of various well established OSS by eclipse
3. Various type of reviews are performed

1. Projects are managed by a hierarchy named the Project Leadership Chain, consisting of the Project Management Committee (PMC), the project leaders of the Top-Level projects and regular project leaders.
2. Meritocracy feature of development
3. Various type of reviews are performed

1. Somehow implements restriction on the development process.
2. At critical mass situation it is difficult to perform reviews and manage.