DEVELOPING ENHANCED OPEN SOURCE SOFTWARE PROCESS MODELING (DEOSSPM)

A Thesis Presented

by

Haimanot Abun Amanu

to

The Faculty of Informatics

of

St. Mary’s University

In Partial Fulfillment of the Requirements
for the Degree of Master of Science

in

Computer Science

February, 2020

ACCEPTANCE

Developing Enhanced Open Source Software Process Modeling (DEOSSPM)

By

Haimanot Abun Amanu

Accepted by the Faculty of Informatics, St. Mary's University, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Thesis Examination Committee:

____

Internal Examiner

____

External Examiner

____

Dean, Faculty of Informatics

January 2020

DECLARATION

I, the undersigned, declare that this thesis work is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been duly acknowledged.

____

Full Name of Student

____

Signature

Addis Ababa

Ethiopia

This thesis has been submitted for examination with my approval as advisor.

____

Full Name of Advisor

____

Signature

Addis Ababa

Ethiopia

February 2020

ACKNOWLEDGMENTS

Before all, I would like to thank God, for giving me strength and commitments to begin and finalize the study.

Then I would like to extend my thanks to my advisor Dr Asrat Mulatu for giving me an opportunity to research a somewhat unique and not so much done before. Thanks especially for a flexible yet supportive advisory style. My appreciation and respect to those who have pioneered open-source software development. It is a truly unique approach and a fascinating area for research.

At last, but certainly not least, most heartfelt thanks to my wife, Hirut Wossenu for her motivation and continued tolerance of long hours at the keyboard.

Table of Contents

List of Acronyms

BSD: -Berkeley Software Distribution

CASE: -Computer Aided Software Engineering

CIFS: - Common Internet File System

CSSD: - Closed Source Software Development

CSS: -   Closed Source Software

C2Net:- Cloud Collaborative Networks

CVS: -   Concurrent Version System

DoD:- Department of Defense

FSF: -   Free Software Foundation

GNU: - Gnu’s Not Unix

GPL: -   General Public License

HTTP: -Hyper Text Transfer Protocol

KLOC –Thousands Lines Of Code

MIT - Massachusetts Institute of Technology

NCSA: -National Center for Supercomputing Applications

OSS:- Open Source Software

OSSD - Open Source Software Development

OSSDP;-Open Source Software Development process

POSIX:-Portable Operating System Interface

RAD: - Rapid Application Development

SDLC: -Software Development Life Cycle

SMB:- Server Message Block

SYS V: System V

TMRC: Tech Model Railroad Club

List of Figures

List of Tables

Abstract

Developing Enhanced Open Source Software process model (DEOSSPM) closely resembles the modeling process of conventional software where the most common elements in process modeling will be identified. This thesis firstly shows the current open source software modeling types and techniques with their limitations. After reviewing the types of modeling and techniques, a new open source software process model is proposed. The proposed model shows the common features of open source software process flows and proposed a new process modeling methods and technique. In addition, it shows the advantages with its challenges and limitations.

The proposed model is designed after reviewing different Open Source Software process models. The research tries to summarize the challenges from reviewed open source projects, and Models the open source software processes. Mainly the existing models haveno well-structured process models. This leads difficulty with reusing the codes and poor documentation.

So, many models are reviewed and analyzed through systematic research technique with thematic analysis methodology. In addition to this, questionnaires are prepared to software developers in Addis Ababa and analyzed the result. Web interviewing through email methodology is also used to design this process model. Sothe new proposed model tries to solve documentation problem and reusing codes, which were challenges of existing open source software process models.

Validating the model is performed through different questionaries' suggested to open source users of software developers from Addis Ababa Ethiopia and from open source users from the web.

Generally, this thesis attempts to describe an introductory process model for open-source software development. Common characteristics are identified and discussed with specific examples from various open-source projects. The results lend support to suggestions that open-source software development follows an adaptive lifecycle, with a flexible management model emphasizing leadership, collaboration, and accountability. Moreover, open source would seem to represent an alternative approach to distributed software development that, able to offer useful information about common problems as

Well as possible solutions

CHAPTER ONE

INTRODUCTION

1.1. Background

Free and open source [1] software is an umbrella term for software that is free and open source software. Free and open source software allows the user to inspect the source code and provides a high level of control of the software's functions compared to proprietary software.

It is software like any other software's. However it is distinguished by its license, or terms of use, which guarantees certain freedoms, in contrast to closed proprietary software which restricts these rights. Open source software guarantees the right to access and modify the source code, and to use reuses and redistribute the software, all with no royalty or other costs. In some cases, there can be an obligation to share improvements with the wider community, thus guaranteeing global benefit.

Most of the open source project developers are working remotely, mostly in their free time and mainly on their personal interest [1] [2]. But in the proprietary software development the project manager sets a target and the developers are bound to work according to the plan [3].

In general terms, open source software is licensed under terms which allow the user to practice, the so called .four freedoms.:

1. Use the software without access restrictions, within the terms of the licence applied
2. View the source code
3. Improve and add to the object and source code, within the terms of the license applied and this may include a term making it mandatory to publish modified code on the community website
4. Distribute the source code.

History

Open Source is firmly rooted in the Hacker Ethic [4]. In the late 1950's, MIT's computer culture originated the term "hacker", defined today as "a person who enjoys exploring the details of programmable systems ". Various members of the Tech Model Railroad Club, or (TMRC), formed the nucleus of MIT's Artificial Intelligence Laboratory. These individuals were obsessed with the way systems worked. The word *hack* had long been used to describe elaborate college pranks devised by MIT students; however Tech Model Railroad Club members used the word to describe a task "imbued with innovation, style, and technical virtuosity". A project undertaken not solely to fulfill some constructive goal, but with some intense creative interest was called a hack.

In 1983 [4], Richard Stallman, longtime member of the hacker community at the MIT Artificial Intelligence Laboratory, announced the GNU project, saying that he had become frustrated with the effects of the change in culture of the computer industry and its users. Software development for the GNU operating system began in January 1984, and the Free Software Foundation (FSF) was founded in October 1985. An article outlining the project and its goals was published in March 1985 titled the GNU Manifesto. The manifesto included significant explanation of the GNU philosophy, Free Software Definition and "copy left" ideas.

Open source software

The term "open source" is alternately used to refer to a philosophy, a way of doing business, and a software development methodology. The philosophy behind open source is based on the concept of free software. Free software refers not to price but to liberty, or the freedom to modify and redistribute source code. This belief is founded on a social ideal advocating collaboration through knowledge sharing.
OSS as defined by the open source Initiative [4] is software that must be distributed under a license that guarantees the right to read, redistribute, modify and use the software freely. Open source has achieved widespread recognition largely as a result of industry resistance to the notion of free software. Many free software practitioners felt that their

business and development practices were not being given fair consideration due to misconceptions about the underlying philosophy. The open source label was establishedto market the commercial viability of free software, while maintaining the same basic approach. Since traditional licenses and fees cannot be used with open-source software,

Generally Open source software is developed in voluntarily basis by the global network of developers and available free on the internet. It is often described as ‘free’ software, which reflects the liberty not the price of the software [5]. The Open Source Softwarelicense gives users to following four essential ‘freedoms [6].

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

Open source software development

The development of software [6] can also be broadly classified into Open Source Software Development (OSSD) and Closed Source Software Development(CSSD). CSSD can be defined as the one where, trained software professionals are employed in developing a software product. In many cases, these software professionals follow a defined and documented software development process. CSS products are developed for commercial purposes (for profit) and only the executableis sold through sales team/person to the licensed customers. Also, the source code is not released to public and cannot be modified as most of the products would be protected under the copyright license or patents . Further, CSSD in general has a formalized organization and structure. Some well-known examples of CSS products are Microsoft Windows, Adobe Acrobat Suite, Oracle solutions, Blackboard, etc. On the other hand, OSSD is oriented towards the joint development of a community [6][7] of developers. OSS products are developed by volunteers out of interest and any person could be volunteer to play any role in its development, based on their skills and interest. Usually, the volunteer's self-assign tasks that they would like to perform. Also, OSS is built as an open source project initiated by an individual/group of people to meet their immediate requirement . The people involved in OSS an d its development share ideas, ideologies, technologies, source code and yet work independently in a geographically distributed environment and are spread across the world . They communicate through Internet forums, e mails, and informal chats or through any other communicative channels. Also, majority of the OSS does not have corporate owner or management staffs to organize, direct, monitor, and improve the software development practices that are followed for development . Some well-known examples of OSS products are Linux, Firefox, Moodle, etc.

Table 1.1: comparison between OSSD and CSSD processes

No	Attributes	OSSD	CSSD
1	Formalized Organization	No	Yes
2	Defined structure	No	Yes
3	Follow a defined and documented development process	No	Yes
4	Most often the development happens in ad hoc fashion	Yes	No
5	Source code made available to all its user	Yes	No
6	Developed for commercial benefits and financial profits	No	Yes
7	Wider space for testing	Yes	No
8	Reliable and responsible 24X7 software support	No	Yes
9	Up to date technical reports/documents	No	Yes
10	Up to date user documents	No	Yes
11	Very intuitive and outstanding software design	No	Yes
12	Append many new feature to cope competition	No	Yes
13	Burdened with license cost	No	Yes

Software process models

A software process comprises the activities, methods, and practices necessary to develop a software system. Software process models are abstractions of a particular development approach. The purpose of a process model is to reduce complexity of understanding by removing unnecessary detail.

Process models can be either prescriptive or descriptive. A prescriptive model characterizes what is supposed to be done, whereas a descriptive model captures what is open source software is a computer software that has a source code available to the general public for use as is or with modifications. This software typically does not require a license fee. In short, OSS is a software whose source code may be freely modified and redistributed with few restrictions, and which is produced by loosely organized, ad-hoc communities consisting of contributors from all over the world who seldom if ever meet face-to-face, and who share a strong sense of commitment.

The basic principle for the OSS development process (OSSDP) is that by sharing source code, developers cooperate under a model of systematic peer-review, and take advantage of parallel debugging that leads to innovation and rapid advancement.

To develop the open source software, process flow modeling is crucial like other software developments. This software process model describes the overall flow and sequence of software project life-cycle activities, including project planning, tracking, requirements management, software construction and release. Such models can be used to develop more precise and formalized descriptions of software life cycle activities. Their power emerges from their utilization of a sufficiently rich notation, syntax, or semantics, often suitable for computational processing.

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. 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. 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.

The development process of an OSS project consists of the visible phases: These phase are problem discovery, Finding volunteers,. Solution identification, Code development and testing, Code change review, Code commit and documentation Release management

Table 1.2: Difference between traditional and OSS projects

Software Development Characteristic	Traditional Closed Source Software Development	Open Source Software Development
Code accessibility during the development	Not publicly accessible during and after development	Publicly accessible during and after development
Team member location	Usually co-located	Geographically dispersed
Environment/Norm	Physical/Hierarchical	Virtual/Decentralized
Team size	More or less fixed	Changes frequently, can become very large
Product lifecycle	Traditional	Longer, focus on continuous releases
Team member characteristic	Paid employees	Volunteers, open membership
Degree of user involvement	Relatively low	Very high
Legal and license issues	Few	Many

Process models and open source software projects

Process models are different methods of accomplishing a software project in an organized manner. Each process model has some advantages for specific type of project with the available resources and timeline [23].

In most of the open source projects, requirements are not well defined at the beginning [24]. In fact, every release in such a project generates new ideas and specific needs of different groups which, in turn, produce updated requirements. For this reason, the open source software projects generally use the software development model that is designed for rapidly changing environment.

Software process [25] that "*rapid prototyping, incremental and evolutionarydevelopment, spiral lifecycle, rapid application development, and, recently, extreme programming and the agile software process can be equally applied to proprietary and opensource software*".

However the process models are exclusively framed for proprietary software and they do not consider the option for remotely connected developers.

In the context of project management in OSS paradigm, [8] describe four types of OSS development model:

- High shared conceptualization, high modularity (the high-high case).
- High shared conceptualization, low modularity (the high-low case).
- Low shared conceptualization, high modularity (the low-high case).
- Low shared conceptualization, low modularity (the low-low case).

High shared conceptualization, high modularity: predefined project plan and large number of developers able to work concurrently. The Linux model is an example of high-high case.

High shared conceptualization, low modularity model: Generally have predefined project plan and little community efforts. The open source projects initiated by commercial company generally falls under this category.

Low shared conceptualization, high modularity model: Allows community developers to develop innovative software. There is no stable project plan and allows large number of developers.

Low shared conceptualization, low modularity model: Used by individual, small group or research projects. There is no predefined project plan and few opportunities for community developers.

Both the models shown above miss well defined documented process. So the proposed model shall be addressed this gap.

1.2. Statement of the Problem

The Open Source Software Development (OSSD) process model closely resembles the modeling process of conventional software development process model where the most common element in the development process will be identified by the OSSD process model. Lately there have been significant demands for process modeling, which have been raised because of the complexities characterized by this OSS process model from the previous methods.

In modeling open source software process, there are some problems to handle. Concurrency allows development to proceed more rapidly, however there is also some associated overhead. With many development activities occurring in parallel, frequent merges are necessary to synchronize change. Merges are time consuming, and can result in code approval bottlenecks if not properly managed. Open-source projects rely almost singularly on peer review for quality control. Other types of testing are uncommon, particularly at the design stage. Projects need to build-in as well as test-in quality. Peer review is sometimes insufficient for identifying high-level architectural flaws. It is also often ineffective at finding obscure flaws. This includes errors that do not happen often and are difficult to identify through source inspection.

Furthermore, the efficiency of large-scale peer review is unclear. Some argue that although one person will eventually find a bug, many others will nonetheless spend time looking for it. Open-source projects also rely heavily on downstream error detection, and corrections are more costly to make at the source code level than at the design stage. So although this approach appears very effective, it may also be somewhat labor intensive.

Even though, domain experts write code eliminates the possibility of misunderstood requirements; however it can also introduce feature creep. Customers with direct access to the product often have an increased desire for features. Coupled with the risks associated with an evolutionary approach to development, a design can quickly expand or wander in conflicting directions as too many quick additions are made to the code base.

Perhaps more importantly, there is no feedback loop to true end users and no imperative to create one. There is a noticeable technical bias, as product concepts tend to emphasize user activities rather than user behavior. Moreover, when developers are not experienced users of the software, they are unlikely to have the necessary expertise or motivation to succeed in an open-source project.Although much anecdotal evidence exists for a wide range of projects, this information has not been placed within a common framework.

Consequently, the current perception of what constitutes open-source software development remains somewhat subjective. Frequently cited practices are neither well documented nor universally agreed upon. Without a consistent format for discussion, it is difficult for researchers and practitioners alike to attempt to emulate or even assess open-source projects. It is therefore important to establish at least an introductory process model for open source software development. Common practices can be documented and used to develop a descriptive model that discusses the methodology in the context of contemporary software engineering. By improving the process definition, it becomes easier to study the dynamics of open-source software development objectively and inmore detail.

So, the newly process model addresses the gaps of previous models mainly two open source challenges or questions, those are documentation the whole process and reusing of codes.

1.3. Objectives

In order to tackle the aforementioned problems, general and specific objectives are set In the following sub sections.

1.3.1. General objective

The main aim ofthis research is to access the existing open source software process model gaps and propose Enhanced Process Model for open source software development projects

1.3.2. Specific objectives

In order to achieve the general objective, several specific activities shall be performed sequentially under specific objectives.

1. Survey the current literatures relating to open-source software process models.
2. Review open-source projects.
3. Identify gaps of existing Open Source Software Process Models.
4. Develop Enhanced Open Source Process Model.

1.4.Research questions

The research attempts to answer these basic questions.

1. What type of challenges do we face using open source software ?
2. Is there a well-defined process modeling techniques for the development of open source software?
3. Is documentation is important during development of open sources software?

CHAPTER TWO

LITERATURE REVIEW

2.1. Introduction

The open source software movement has received great attention in the last several years. It is often characterized as a fundamentally new way to develop software [7] that poses a serious challenge [8][9]. To the commercial software businesses that dominate most software markets today. The challenge is not the sort posed by a new competitor that operates according to the same rules but threatens to do it faster, better, cheaper. The OSS challenge is often described as much more fundamental, and goes to the basic motivations, economics, market structure, and philosophy of the institutions that develop, market, and use software.

The basic tenets of OSS development are clear enough, although the details can certainly be difficult to pin down precisely [10]. OSS, most people would agree, has as its underpinning certain legal and pragmatic arrangements

In the last ten years, open source software (OSS) has attracted the attention of not only the practitioner, but also the business and the research communities. In short, OSS is a software whose source code may be freely modified and redistributed with few restrictions, and which is produced by loosely organized, ad-hoc communities consisting of contributors from all over the world who seldom if ever meet face-to-face, and who share a strong sense of commitment [11]. The basic principle for the OSS development process (OSSDP) is that by sharing source code, developers cooperate under a model of systematic peer-review, and take advantage of parallel debugging that leads to innovation and rapid advancement [12].The most distinctive features of the open source system are identical to those that distinguish the proprietary model: the development structure, the licensing scheme and its relationship to intellectual property rights, and the policy treatment of the source code [11].

So a software process model can be defined as a simplified description of software process that presents one view of a process and may also include activities that are part of software process and products, along with the constraints that apply to the process and roles of the people involved [12].

There are many approaches which try to explain the development of the OSS, but still no general approach is being agreed.

2.2. Existing OSS development process models

Several researchers have proposed OSS process models derived from analyses of successful open source projects [13]. Opinions differ as to the stages that comprise a typical open source development project. However, regardless of the open source life cycle model that may be subscribed to, the OSSD paradigm demonstrates several common attributes.

1. parallel development and peer review
2. prompt feedback to user and developer contributions
3. parallel debugging
4. user involvement, and rapid release times
5. highly talented developers

1. Parallel development and peer review

Members of the open source process model developers review the code and the process, provide comments and feedbacks to improve the quality and functionality and do early test to catch bugs and provide enhancements as early as possible in the development cycle. These results, high quality process model. But , there is a a challenge of organizing ideas of all members and finalizing the process as scheduled.

2. Prompt feedback to user and developer contributions

During releasing each code and process, it is tested by the community who discus through mailing lists and discussions boards and provide feedback, bug reports and fixes through the project mailing list. The given feedback is considered by the developers. The cycle is done repeatedly throughout the process development until project members feel that the implementation is stable enough. This improves the quality of process model. But asa limitation, selecting better codes and process among many alternatives is very difficult. So that additional time and effort is needed to the whole process.

3. Parallel debugging

With this bug fixing techniques, all the members are downloading the codes and run the test individually. All members find a failing test and try to fix the bugs according to their skill. Finally all members open pull request and wait for approvals or feedbacks from the coordinator for merging. The challenge for parallel debugging is, it takes time and difficult to select the better ones among many alternatives that will satisfy customer need.

4. User involvement, and rapid release times

In developing the open software, it does not wait to have a fully working version to make the code pubic and release a version of the software. Rather, release early and release often leads to higher quality software because of peer review and the large base of users who are using and testing the software. Many people looking at the code benefits that, the code is reviewed for adherence to coding style; fragile or inflexible code can also be improved because of thee reviewers. But it also consumes high effort and time which is considered as drawbacks.

5. Highly talented developers

In developing open source software, highly talented developers are motivated to join the project and give their contributions. This technique gives opportunity to develop better codes and process. But, selecting the better code among many talented developers will be very challenging.

6. highly talented develop``ers

Highly talented developers are participating in open source software developments. due to this , high quality software will be developed. So that needs of customers will be achieved. In the other hand, selection of the appropriate code will be very difficult. This takes time and effort.

2.2.1. Process models for OSS

1. United States Department of Defense (DoD) Model

The United States Department of Defense (DoD) [14] [15].Proposed a process model for the development of the OSS. The main focus of the model is on the contributor roles who participate in the development process of the OSS. [10].

This OSS process flow life cycle consist of the following.

Developer*:* Developers consist of the persons who imitated 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 continues involvement in the development process and became the part of the core developer community. Trusted developers are allowed to make updating and changes in the trusted repository directly.

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 categorized as Passive and Active user. The Passive users are those who download the software for use, they never participate in development. The Active user participate in the development process 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, whereas feedback/bug report use bottom-up approach. It flows from user - distributor - trusted repository- trusted developer - developer.

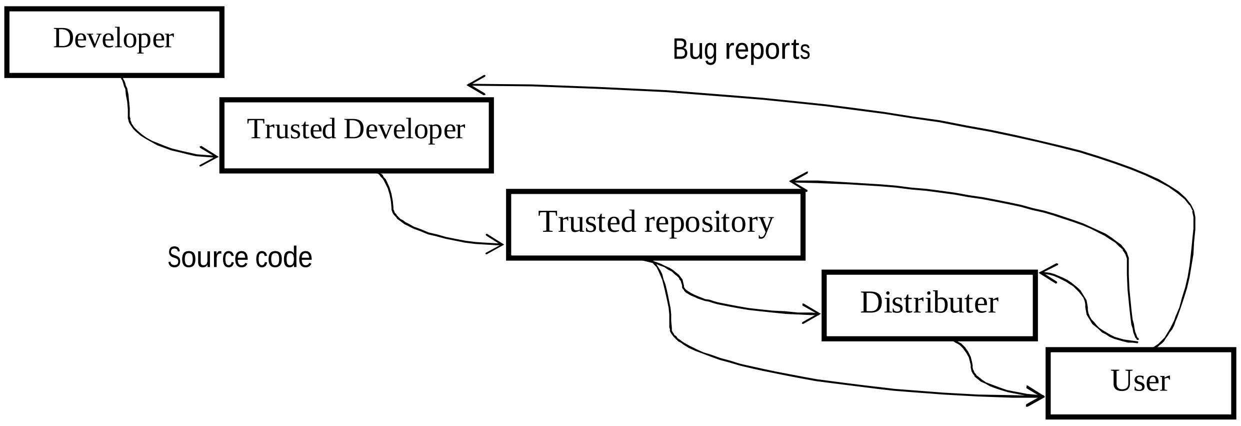


Figure 2.1: Development Model Propose by DoD

Major Features of DoD model

1. OSS is developed by collaborative process.
2. Most OSS projects have some web location as “trusted repository” where people canget the “official” version of the program or software also some related important information (documentation, bug report system, mailing lists, etc.). Users or developers can get the software directly from the trusted repository, or get it through distributors. Distributors are who acquire it and provide additional value such as integration with other components, testing, special configuration, support, and so on.
3. The trusted developers are developers who are allowed to modify the trusted repository directly. At project start, the project creators or initiator are mainly thetrusted developers and they determine who else may become a trusted developer of t is initial trusted repository. All other developers can improve the software by changing local copies and also can post their versions to the internet. But they must submit their changes to a trusted developer to update the trusted repository.
4. Users can send bug reports to the distributor or trusted repository and can be taken care accordingly

2. Cathedral-Bazar life cycle model

In an OSS development process model, the complementary events i.e. Cathedral (closed style) and Bazaar (open style) are considered to occur in the same OSS project [13].Many successful OSS projects consist both of these modeling style during their development. The proposed development model for OSS consists of mainly three phases.

1. Cathedral Phase
2. Transition Phase
3. Bazaar Phase

Their process modeling starts with in Cathedral phase where cathedral building development style is followed. In the cathedral phase, small group of individual developers initiate the development in the closed environment with no outside audience or participant. It consists the project author, core developer, and project manager. 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 various phases of the development model proposed by [13] are shown below.

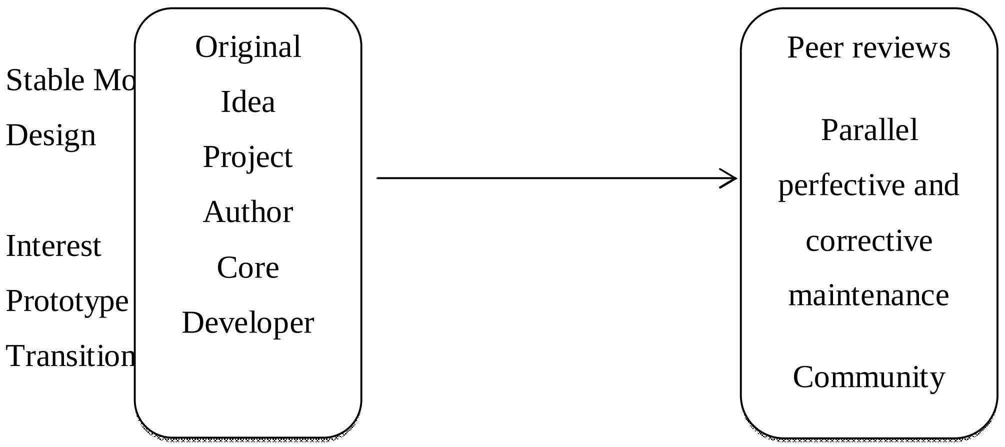

Stable Mo
Design
Interest
Prototype
Transition
Original Idea Project Author Core Developer
Peer reviews
Parallel perfective and corrective maintenance
Community

Figure 2.2: OSS Development process Lifecycle

Limitations of the model

The original idea is developed by core developers which are small in number. Although, final codes will be reviewed by peers, needs of customer could not be addressed fully. In addition to this, the model will not be accomplished in specified time frame, which brings un satisfaction for the users.

3. Comparative life cycle model

The comparative model [14], proposed the development life cycle of the OSS thatincludes all the elements of the classical SDLC used for the development of the closed or commercial softwares. According to this process model, software requirements are based on the need of the developers who develop the OSS in the analysis phase. The final requirements are negotiated on the internet. The design phase in the development of OSS is not performed formally [14]. Design issues are not written anywhere, and not made visible to everyone, 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 compared to the testing of traditional software.Because the number of the user available to test the OSS is very large as compared to the traditional software testing. These tester or user sends the feedbacks in the form of reviews, bug report, change request. The testing which is performed in the OSS development is un structured. The disadvantages of unstructured testing are covered by having the uncounted stranger, that tests the OSS. It 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 [15].

Limitations of the model

Even though the model focuses mainly coding and implementation, which helps to develop the model quickly. But requirements of the software are based on the developers, unstructured testing and large number of users available to test the OSS, makes the model difficult to attain customers need and complete the model through specified time.

4. Organizational life cycle model

This model is a kind of organizational method [16] which consists of 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
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.

2. Going open phase activities

a) The project founder have a 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) If 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, and documentation. [17].

Limitation of the model

As far as the initial stage is initiated by a single or group of developers, the whole process is guided by single or few groups. This leads less quality of the software being developed. Here testing and documentation will be better due to less involvement of users. In addition to this, the model will help to accomplish the model in specified time.

5. Maturity plus life cycle model

This model proposed similar kind of open source process modeling with slight variation of organizational model [17].The model adds one more phase i.e. the maturity phase at last. The maturity phase specifies that, the OSS gained a critical mass in terms of number of users, developers. Here, the software 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 others, consider managerial aspects and organizational structure of OSS, but do not provide task related analysis of OSS development. [17][18].

Limitation of the model

This model adds the maturity phase from organizational model which gives opportunity for users to enhance the model. This gives the model a chance of being quality. But more time and great effort is needed to accomplish th model.

6. Managerial activity focusing life cycle model

This model Consist of 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 model considers managerial activities (developer management and the work to be done), but not product-related activities. Itconsiders more task-related issues than the above model.It assumed that, some kind of prototype is already exists for the software. The proposed model at some level provides support for all the phases of traditional SDLC. The model focuses on the planning phase of the SDLC. Other phases like, design and analysis are not considered adequately. [18].

Limitation of the model

This model focuses mainly planning phase. So that it lacks user involvements which is back bone of the quality of the software.

7. Licensing and version control model

This model consists Open source licensing and version control. The proposed life cycle for the development of OSS is considered to start with the personal itch to the individual or group of persons. The initiator looks 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 initiator will join the existing project. In both 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 documentations, 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 continues until software gain certain maturity [20].

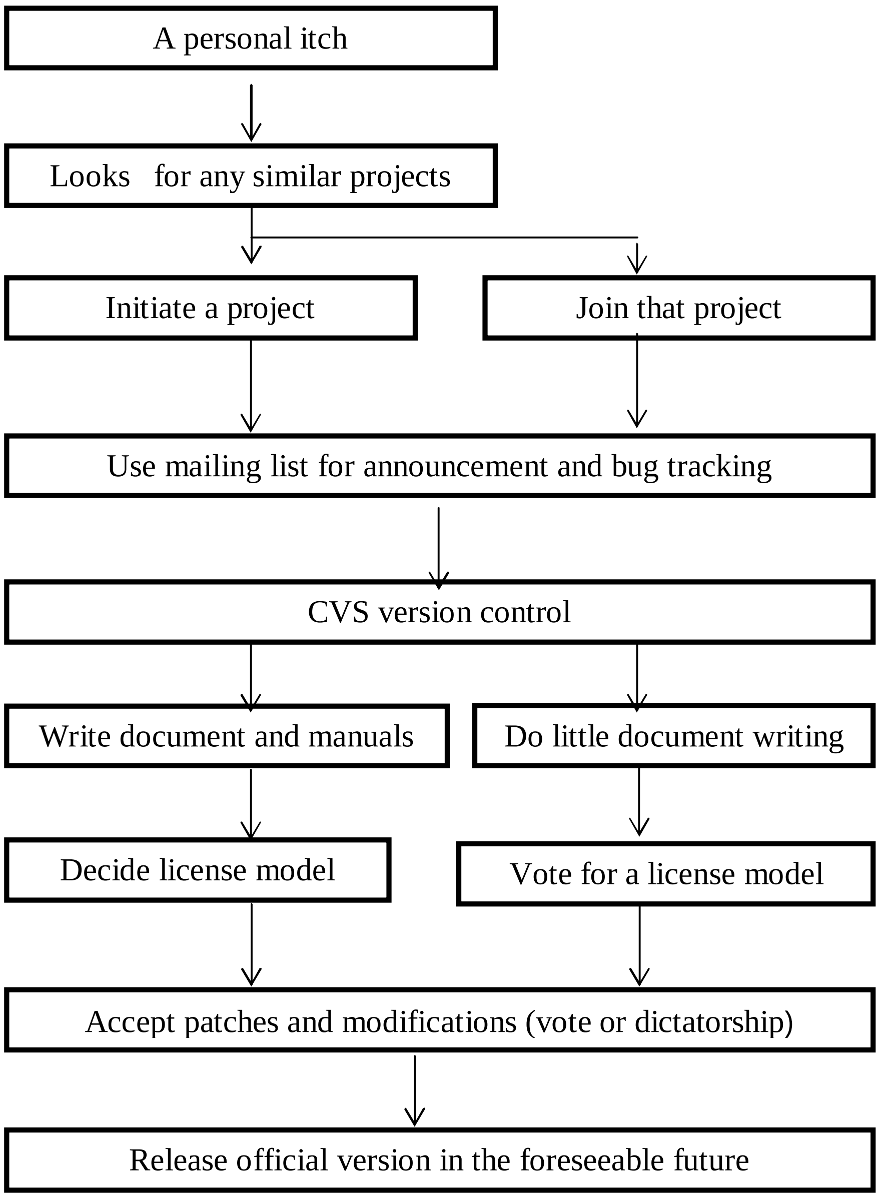

A personal itch
Looks for any similar projects
Initiate a project
Join that project
Use mailing list for announcement and bug tracking
CVS version control
Write document and manuals
Do little document writing
Decide license model
Vote for a license model
Accept patches and modifications (vote or dictatorship)
Release official version in the foreseeable future

Figure 2.3:licensing and version control model [20]

Limitation of the model

User involvement is high. So that good quality of software will be achieved. In addition to this, users are involved in writing documentations which is not feasible in other models. But the time needed to accomplish the model is very high and difficult to manage it because of complexities of the process.

8. Product-task related life cycle model

This process flow model is a refinement of the above model. it describes the product or task related activities. This model is widely accepted as a frame work for the OSS development.

Various phases or activities which are performed are as follows [21].

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 then 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.

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.

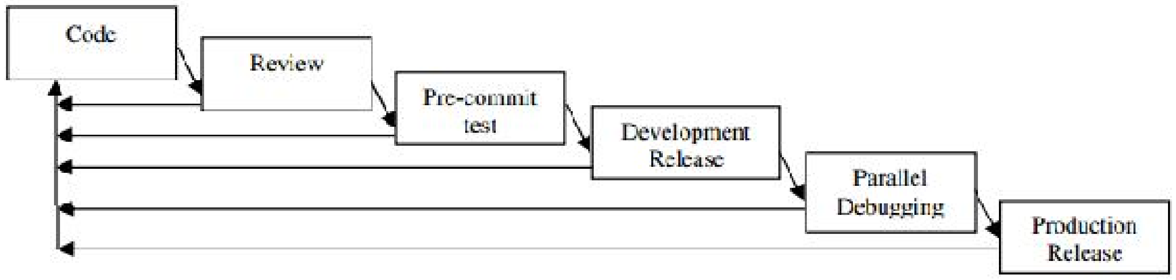


Figure 2.4: product task related model

Limitation of the model

This model describes the product or task related activities like coding, reviewing, pre commit testing, development releasing, parallel debugging and product releasing consequently. This makes the developed software will be high in quality and user involvements. But time and effort is highly needed and documentation is also very difficult.

9. Expanding Product-task related life cycle model

The model expands the above process flow model and includes the aspects of previous models. It encapsulates the classical SDLC model phases. This model replaces the initial code phase of Jorgensen model with initiation phase. 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 are 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 activates 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. [22].

Limitations of the model

The model refines the above mentioned process and the final production released software can further move to Initiation phase for further development. This cycle Continues again and again until the OSS gain Maturity. This needs more and more effort and time. In addition coordination and documentation will be very difficult.

10. Scacchi's life cycle model

This model considers 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 Mange configuration, Download and install, End user, and communicate experience. [23].

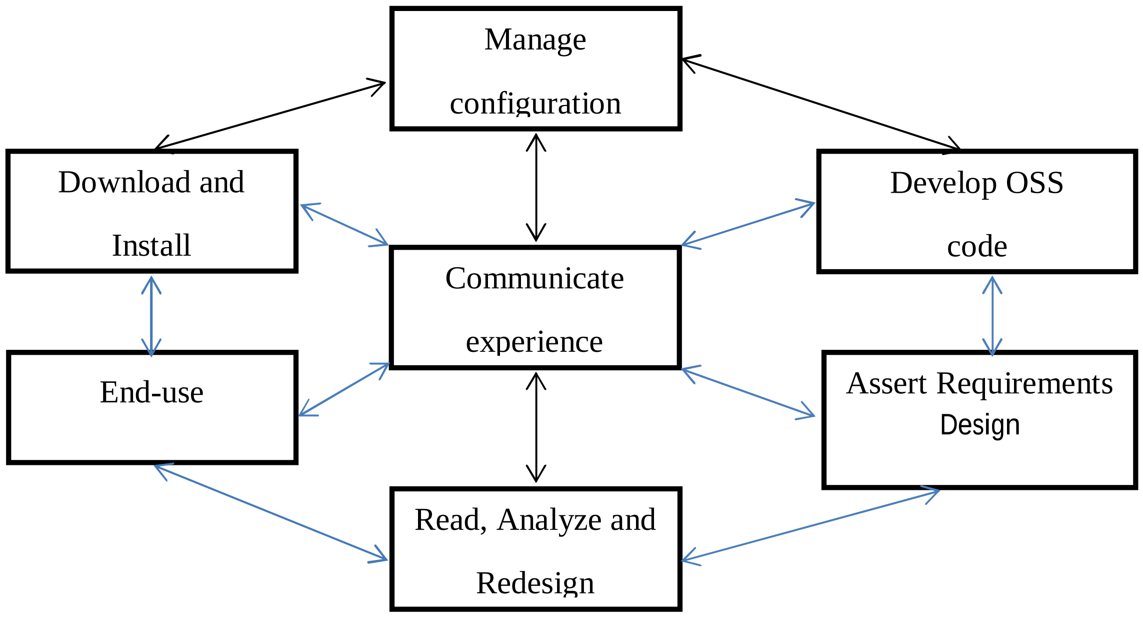


Figure 2.5: Scacchi Process flow

Limitation of the model

Scacchi model considers the cyclic nature of the development process, with centralized role of experience sharing. This centralization hinders individual talents in the development of the software development. This minimizes the quality of the software.

11. Gilliam’s life cycle model

The proposed model is considered to have the cyclic nature. Figure 2.7 also depicts various activities such as making 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. [24].

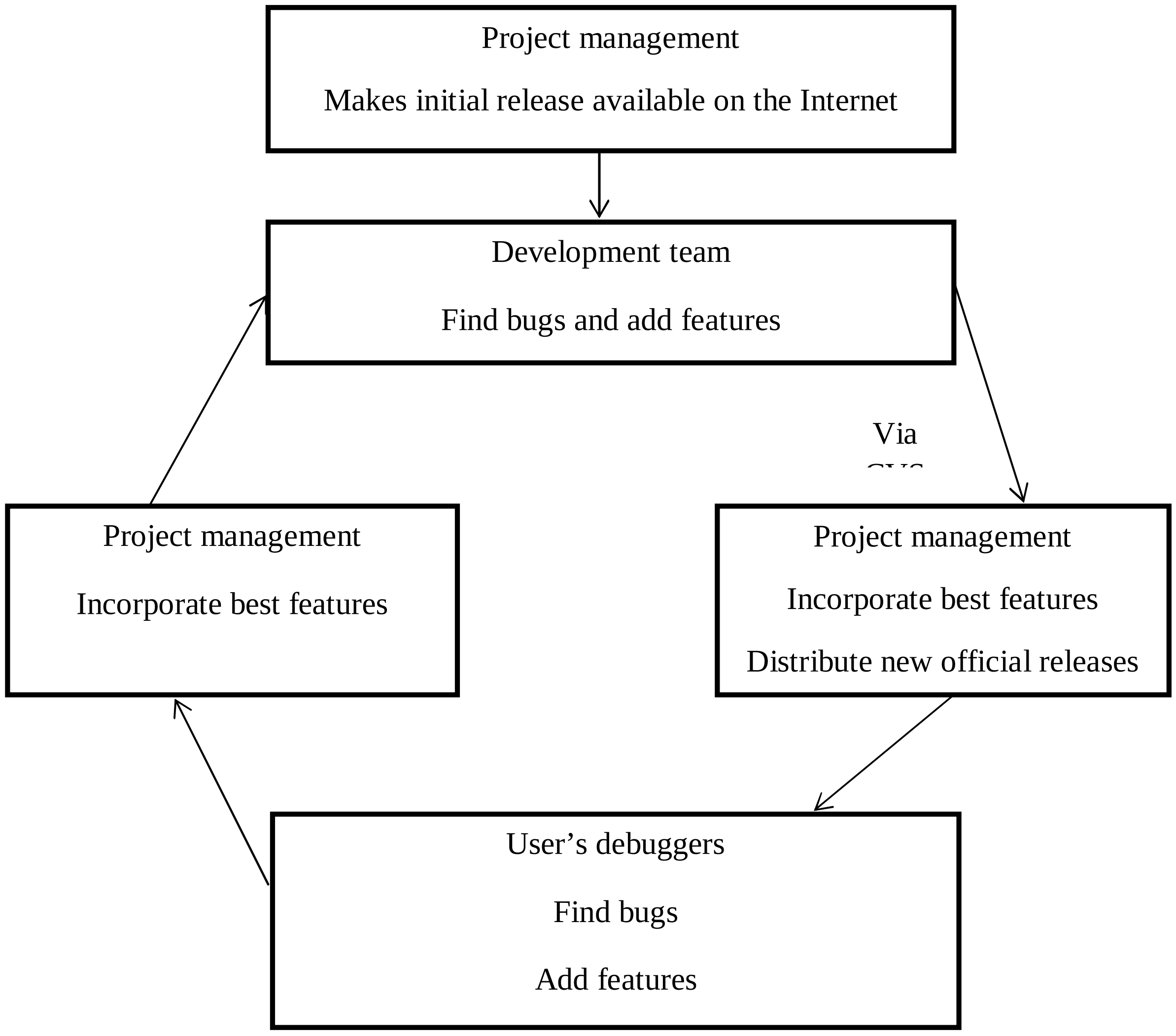

Project management
Makes initial release available on the Internet
Development team
Find bugs and add features
Via
Project management
Incorporate best features
Project management
Incorporate best features
Distribute new official releases
User's debuggers
Find bugs
Add features

Figure 2.6: Gilliam model

Limitation of the model

As far as various activities such as making initial release, finding bugs, adding features, and bug fixing are cyclically performed, great effort and time is needed and difficult to manage the whole process.

Generally, software process modeling has some features. In contrast to software life cycle models, software process models often represent a networked sequence of activities, objects, transformations, and events that embody strategies for accomplishing software evolution. Such models can be used to develop more precise and formalized descriptions of software life cycle activities. Their power emerges from their utilization of a sufficiently rich notation, syntax, or semantics, often suitable for computational processing. [25]. A process model constitute four important process elements namely activity, products, roles, and tools. Activity consists of one or more process steps, which may run in parallel with other process steps. Products are artifacts which are normally under configuration control. Roles describe the responsibility and rights of the human who performs the process steps or who is in charge of the human agents. It should be noted that a person may play more than one role and a role may be associated with several people. Tools covers any tools used in the production of the software including compilers, debuggers, editors and even CASE tools.

There are several modeling techniques that exist in modeling the OSSD process. These modeling techniques provide different methods in capturing the OSSD process. The modeling techniques can be chosen based on their suitability to the user needs and user should know which techniques do provide the needed requirement. [26].

Table 2.1: Comparison between various OSS Models with their characteristics

Process models	Phases	Methodology	Advantages	Disadvantages
Cathedral bazar model	• Cathedral • Transition • Bazaar	• Development of every OSS start in cathedral phase and then transition is done to Bazaar Phase	• Simple • Generalized view of OSS development life cycle	• Does not specify the parameters for the transition
Comparative model	• Analysis Design • Coding Implementation • Support	• Consider all the phases of SDLC • Main focus is on coding and implementation	• Presents all the differences that OSS life cycle has as compare to the SDLC	• Fails to suggest an appropriate model that analyses this new process
Organizational life cycle model	• Project Initiation Going “Open” • Project Growth, Stability or Decline	• The project is made open to the outside world by sharing or publishing the work on internet. • Project growth is accessed and categorized to growing,stable. or decline	• Organizational Model • The modular approach is followed for the development of the OSS • Each phase is characterized by different set of activities	• Do not considered the task-related analysis of OSSD process

Maturity plus model	• ProjectInitiation Going “Open” • Project Growth, Stability or Decline • Maturity	• project reaches critical mass in terms of the numbers of users and developers it can support .	• Consider managerial aspects and organizational structure of OSS	• Do not considered the task-related analysis of OSSD process
Managerial activity focusing model	• Rolesand Responsibilities Identifying work to be done • Assigning and performing development work • Pre-release testing Inspections Managing releases	• all the activities and phases which are defined has main focus on the managerial activities	• Consider decision making framework and task related project phases. • Consider managerial activities • More emphasis is on planning phase of life cycle	• Do not considered product-related activities • Fail to explain where design and analysis take place in this model
Licensing and version control model	• Personal Itch • Look for similar project • Initiate • Version Control • Documentation • Decide OSS license • Patch generation • Releasing	• Specifies the process how the generation of the OSS take place by performing all these activities along with generation of licensing and version controlling.	• Consider open source licensing and version control. • Participation in the development is open for any-one	• Does not specify the how and where designing, testing, and planning take place
Product-task related model	• Code • Review • Pre-Commit Test • Development • Release • Parallel • Debugging • Production Release	• Perform all 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.	• Task- related Model • Widely accepted at micro and macro level as a framework for OSS development	• Does not specify where or how the planning, analysis, and design take plac
Expanding Product-task related model	• Initiation • Contribution and Review • Pre-Commit Test • Development and Parallel Debugging • Production	• All these phases are performed repeatedly until OSS again Maturity	• Combination of task-related, organizational and comparative models .	• Do not specifies the version control mechanism of OSS development

Scacchi's model	• Assert requirement • Design, • Develop OSS code, • Mange configuration, • Download and install, • End user	• Specifies the development process by performing all these activities in cyclic manner.	• Consider the cyclic nature of the development process • Centralized role of experience sharing	• Do not specify the design, analysis, planning phases of the development life cycle.
Gilliam's model	• Project Management • Development Team • User/debugger	• 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.	• Cyclic nature • Parallel Debugging	• Do not specify the, design, analysis, planning phases of the development life cycle.
Eclipse	• Pre-proposal • Proposal • Incubation • Mature • Top - Level • Archived	• Various phases which are used for development of OSS are performed repeatedly.	• Projects are managed by a hierarchy named the Project Leadership Chain, consisting of the Project Management Committee (PMC),. • Various type of reviews are performed	• Somehow implements restriction on the development process. • At critical mass situation it is difficult to perform reviews and manage.

2.2.2. Characteristics of most OSS development models

1. Interwoven development cycle

The open source development model is characterized by a series of interwoven processes [26] that continually improve code quality, instead of a strictly linear progression to a release. Unlike the big reveal that typically accompanies the traditional software development model; the open source model encourages continuous and independent feature development. This enables new features to be integrated as they are ready, which in turn allows other developers to build upon them more quickly and produce a more competitive product.

2. Release early and often

Release early and often [27]results in highly iterative development, and minimizes the amount of change between development releases, making regressions and breakages easier to diagnose. This release philosophy allows for continuous peer review, where all members of the community have the opportunity to comment and offer suggestions and bug fixes. It also encourages small, incremental changes that are easier to understand and test while developers are actively engaged, rather than being discovered during a separate final test cycle. A side benefit is that the code is frequently reviewed for adherence to coding style, and fragile or inflexible code can be found and improved early in the development cycle.

3. Peer review

The open source development process emphasizes peer review throughout the entire development life-cycle. Developers are expected to submit their code to project mailing lists for periodic public peer review, particularly when a feature achieves a development milestone. This helps to ensure that others outside of the development team are aware of the changes, and can provide feedback before the design is final and implementation complete. Other members of the open source project review the code, provide comments and feedback to improve the quality and functionality, and test to catch bugs and provide enhancements as early as possible in the development cycle. When a feature is complete and ready to be considered for integration, the project maintainer also provides a level of review prior to accepting the code. By the time code is integrated into the main product, it has undergone a number of detailed inspections by others outside of the development team. The result is improved, higher quality code.

2.2.3. Conclusion

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 above. It is found that the development life cycle of open source software is very much different from that of traditional closed or commercial software.

Most of traditional life cycle models are not efficient to be used for the development of open source software which follows the static development approach. The above 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. So designing generalized and standardized process model will be very necessary.

CHAPTER THREE

METHODOLOGY

3.1. Introduction

To design a new process model, more than forty related researches are reviewed. Among those papers, very few researches are focusing on process models. Almost all papers have common challenges or gaps of reusing the code, well organized process model and documentation of the process. So, the paper uses a design science methodology. An artifact of designing the process will be modeled. To design the model the paper systematically reviews those research papers thematically. In addition to this, some questionnaire are prepared for software developers. With those methodologies the paper analyzes gap and proposed new model.

3.2. Research design

The research design is qualitative particularly design research methodology. The purpose of the design is to formulate a well-defined process model for open source software development.

3.3. Sources of data

To get valid and reliable data, the use of appropriate data source is very important. Therefore the source of data for this research includes primary and secondary data sources. The primary data were collected from software developers from Ethiopia. They were considered as a primary source because of either direct involvement in the software development process modeling. The secondary source of the data was documents related to open source software process modeling.

The data was extracted from these sources through both quantitative and qualitative methods. Questionnaires

3.4. Sampling technique

The sample is selected randomly from software developers in Ethiopia. All the samples are selected from Addis Ababa because of the enterprises are well established when we compared to others found in outside Addis Ababa**.**

3.5. Sample size

It is assumed that, most software developers have similar background and experience. So that small sample size would represent the large number software developers. So that it incorporates five software developing enterprises in Ethiopia. The enterprise is selected randomly. Those are:Gebeya Inc found in Addis Ababa at Bole sub city wereda 2 , Sina building 2nd floor. Minab It solution**found at** Gene Commercial Center5th Floor Office No.517 Megenagha, Addis Ababa. Zala tech plc found in AddisAbaba at Bole sub city Cameroon road ,Alpha It solution found Addis Ababa, Piasa, ,Arada sub City , Iceaddis, Addis Ababa, Bole sub city, Bole Medhanialem , Zewudu Building , 5th floor, Apposit LLC and TABY Engineering PLC. Found at Addis Ababa, Bole Sub City at Cameroon Road , Both the enterprises are mainly organized for developments of web sites, graphic designs, customer software development and the like .

3.6. Instruments of data collection

To collect data for the study, different data were employed. These were questionnaire; and document analysis of data gathering tool.

Questionnaire was conducted for software developers in Ethiopia particularly in Addis Ababa. In addition to this, different types of documents written on open software process modeling was analyzed

3.7. Technique of Sampling

The sample is selected randomly from software developers in Ethiopia. All the samples are selected from Addis Ababa because of the enterprises are well established when we compared to others found in outside Addis Ababa**.**

3.8.Data collection methods

In order to gain a better insight into possibilities for developing process model for open software, thematic analysis research is conducted systematically. Documents are reviewed some related questions are prepared and answered through email and telephone. In addition to this, questioners are prepared and given to some software developers and then analysed.

3.9.Method of analysis

The reviewed data, the questioner and the web interviewee are transcribed and thematic analysis was conducted. Before suggesting the new process model, data is organized systematically.
Around ten professionals give response to the questionnaire, according to the responses, documentation problem and well organised modelling are great problems in open source software developments.

3.10. Method of Data Analysis

Qualitative method of data analysis was applied for the data obtained from questions and document analysis. On the whole, the results of the study were presented, analyzed and summarized accordingly. This study aimed at developing enhanced open source software process modeling.

.

CHAPTER FOUR

PROPOSED ENHANCED OSS PROCESS MODELING

4.1. Introduction

The existing open Source software offers significant benefits, compared to typical commercial products. Commercial products often stress on advancement and updating of visible features for getting marketing advantages. It is very difficult to measure quality attributes such as stability, security, reliability etc. Commercial software put notices basically on the quality of mostly used features. Whereas Open Source software developing community consists of very bright, very motivated developers, who are mostly unpaid but are much disciplined to their work. In addition to that, all the users of Open Source software have access to the source code of the software and debugging tools. For this reason, the users can suggest the developers about the bugs by feedback or they can fix the bugs if possible by modifying the source code and even can enhance the software by providing actual changes to the source code. Because of the availability of source code and right to modify the code by users, sometimes the quality of software produced by the Open Source software development community exceeds the quality of same type software produced by purely commercial organizations.

As Open Source Software is not made by a group of people under a common roof, for this the open source software does not follow the conventional model like waterfall, iterative enhancement, spiral etc. Even there is no standard open source development model also. Some open source developers use some model as their own.

The paper has been going to discussed some of those models and have proposed a new model called open agile incremental model, which can be used for open source software development.

. The proposed model combines the conventional software process modeling and open source software process modeling techniques together. Starting from feasibility study of the software to the dissemination stage, the model tries to address the necessary issues concerning with the software process modeling. All the necessary stages of software process model are included in this proposed model

4.2.Presentation, analysis and interpretation of data

Table 4.1: Responses of software developers

Questionnaires	Company Name				
	Alpha IT Solution	Gasha consulting	Atlantic IT Solutions	Zala tech plc	WEBS PRIX
Are you using open source software for your different purpose?	Yes	yes	yes	no	yes
Process modeling is very important for developing open source software	agree	agree	agree	agree	agree
Is there any process model for open source software that you know before?	I don't know	I don't know	I don't know	I don't know	I don't know
documentation necessary for open source software's	agree	agree	agree	agree	agree

`

Sample responses of three software developers

1. What type of challenges do we face during using open source software?

Alpha IT Solution

- Software version consistency
- no maintenance or support is included for free
- No vendor releasing updates
- Testing

Gasha consulting

- technical support
- there is no company that is willing to support us for free
- how the software works and modify --- from public internet forums only which is not comfortable

Atlantic IT Solutions

- lack of support in the local context
- lack of some of reference materials

2. Is there a well-defined process modelling technique for the development of open source software

Alpha IT Solution

- No, but we use platforms for community use

Gasha consulting

- We don’t believe there is one well-defined process modelling technique

Atlantic It solution

No, because:

- lack of interest in software process modeling
- open source software projects are ingrained in the hacker culture and represents the antithesis of software engineering

3. Process modelling is necessary during open source software development

Alpha IT Solution

- Process modelling is important in open source software. But, open source software by its nature cannot model a specific organization’s business process
- it is important to document whatever process model the project implements so that the users of that software understand the business rules being enforced.

Gasha consulting

- Yes, it will provide a clear outline for implementation as well as code refactoring for software developer

Atlantic IT Solutions

- Sharing source code, developers cooperate under a model of systematic peer-review, and take advantage of parallel debugging that leads to innovation and rapid advancement
- Today, Linux and Apache Server are used in the Internet’s public servers. This demonstrates that process modeling can produce software of high quality and functionality.
- Other success stories include Perl, Python and PHP programming languages, sendmail mail handler, Mozilla browser, MySQL database server, Eclipse and Net Beans Java integrated development environments.

4. What is your opinion using process modelling during software developments?

Alpha IT Solution

- Most software projects are moving towards the agile software development processes. Agile process emphasizes communication and collaboration over exhaustive documentation (which must occur in order to properly model a business’s entire process).
- Process modelling will help increase business efficiency more than it directly helps software development. The better the business process is, the better the software will be as

Gasha Consulting

- It’s a good practice that could save time and resources to outline the business logic mapped out through the processes for clearer understanding and accurate implementation strategy.

Atlantic IT Solutions

- Developers who want to join an open source software project must discover its underlying development process by using public information sources on the Web.
- These sources include process enactment information such as informal task prescriptions, community and information structure, work roles, project and product development histories, electronic messages and communications patterns among project participants.

5. Why process modelling of open source software is very important? What is your opinion

Alpha IT Solution

- Process modelling will help increase business efficiency more than it directly helps software development. The better the business process is, the better the software will be as well

Gasha consulting

- To help a smoother adoption of the platform to a set of use cases with a wider range.
- helps to set operation standard of the platform which are usually referenced in deployment and refactor documentation.

Atlantic IT Solutions

- Developers who want to join an open source software project must discover its underlying development process by using public information sources on the Web..

6. Is documentation is important during development of open source software?

Alpha IT Solution

- Most software projects are moving towards the agile software development processes. Agile process emphasizes communication and collaboration over exhaustive documentation (which must occur in order to properly model a business's entire process).
- Process modelling will help increase business efficiency more than it directly helps software development. The better the business process is, the better the software will be as

Gasha consulting

- Yes it is important but exhaustive documentation using the Waterfall methodology is no longer considered useful
- Process models for both specific and standard uses are defined to help a smoother adoption of the platform to a set of use cases with a wider range. Process modelling helps to set operation standard of the platform which are usually referenced in deployment and refactor documentation.

Atlantic IT Solutions

- Developers who want to join an open source software project must discover its underlying development process by using public information sources on the Web.
- These sources include process enactment information such as informal task prescriptions, community and information structure, work roles, project and product development histories, electronic messages and communications patterns among project participants.

7. Do you agree that, well defined process modelling of open source software is mandatory?—

Alpha IT Solution

- Some documentation is important during software development. However, exhaustive documentation using the Waterfall methodology is no longer considered useful. Using the Agile/Scrum software development process, which is the preferred methodology nowadays, documentation is minimal (it is replaced with collaboration with the business analysts/product owners). So, documentation is important for sure, but extensive documentation that sooner or later becomes outdated is not recommended. -

Gasha consulting

- Yes, it will help clarifying the implementation or modification process both after and during deployment

Atlantic IT Solutions

- No.
- The Huge's like Apache, Mozilla, Net Beans, or any other open source projects, provide documents on their Web portals that explicitly and precisely describe what development processes are employed

8. In your view, is reusing of codes in open source software development is easy? Why?

   Alpha IT Solution

- This really depends on the open source software and the quality of the design. Some open source projects produce quality code that can be easily reused, some do not.

  Gasha consulting

- Yes, if one follows the appropriate guidelines of use and implementation of logics it should fairly understandable as most open source software products has a community around them.

Atlantic IT Solutions

- Not that easy, everything may seem or work well. Then when you want to modify them you need to know where to touch the software, and then you will understand the hustle.

9. Is process modelling is important during developing of open source software , if yes why?

Alpha IT Solution

- Process modelling is important but not as important as it used to be. Most software projects are moving towards the agile software development processes. Agile process emphasize communication and collaboration over exhaustive documentation (which must occur in order to properly model a business's entire process). Therefore, for rapid software development, the business analyst or the product owner should be available to answer questions around business processes. But, in general, process modelling will help increase business efficiency more than it directly helps software development. The better the business process is, the better the software will be as well

Gasha consulting

- Process models for both specific and standard uses are defined to help a smoother adoption of the platform to a set of use cases with a wider range. Process modelling helps to set operation standard of the platform which are usually referenced in deployment and refactor documentation.

Atlantic IT Solutions

- Developers who want to join an open source software project must discover its underlying development process by using public information sources on the Web. These sources include process enactment information such as informal task prescriptions, community and information structure, work roles, project and product development histories, electronic messages and communications patterns among project participants.

10. What is your general opinion on process modelling of open source software developments

Alpha IT Solution

- Process modelling is important for all software projects, not just open source. The better the process is modelled to reflect actual business processes, the better the quality of the software will be in that it enforcing those processes properly

Gasha consulting

- **Process modeling is very important for open source software development**

Atlantic IT Solutions

- If (process modelling){everything will be easy to understand} else{needs patience, dedication, coffee and few hours of sleep to understand}

4.3. Supports from empirical study

An empirical study is conducted to investigate the current practices and challenges of open source software process modeling. The findings of the empirical study provided inputs and the required conceptual support to the design of open source software process modeling.

The following key findings were summarized from chapter three that can be used as bases for the designing of open source software process modeling.

- Well defined process modeling for open source software development is necessary for reusing codes and other purposes.
- Documentation of the whole process is very important for open source software process modeling.

Some key challenges for open source software process modeling:

As far as open source software development is developed openly by individuals, it is difficult to manage the process.

Process modeling for open source software developments is not easy like traditional software development.

Documentation of open source software development is not easy like traditional software development.

4.4.Conclusion of the analysis

All of the software developers are using open source software for their software developing purpose. In addition to this all of them agreed that process modeling is necessary for open source software's especially for the purpose of documentation.

In the questionnaire the software developers respondents the following .main points

1. Challenges of using open source software
    - Software version consistency
    - No vendor releasing updates
    - Testing
    - there is no company that is willing to support us for free
    - lack of support in the local context
    - lack of some of reference materials
2. there is no well-defined process modelling technique in ope source softwares
3. necessity of process modelling for open source software
    - it is important to document
    - provide a clear outline for implementation as well as code refactoring for software developer

4.5.Properties of the proposed model

The model has common properties of modifying and used its source code and claims as your own freely. It is like any other software. However it is distinguished by its license, or terms of use, which guarantees certain freedoms, in contrast to closed proprietary software which restricts these rights. The model guarantees the right to access and modify the source code, and to use, reuses and redistribute the software, all with no royalty or other costs in some cases, there can be an obligation to share improvements with the wider community, thus guaranteeing global benefit. These, apparently simple guarantees, have powerful implications: encourage reuse enable innovation, flexibility, and easier integration Open source software is licensed under terms which allow the user to practice.

1. Use the software without access restrictions, within the terms of the license applied
2. View the source code
3. Improve and add to the object and source code, within the terms of the license applied and this may include a term making it mandatory to publish modified code on the community website
4. Distribute the source code.

4.6. The proposed model process

It 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” [4]. 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 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. 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.

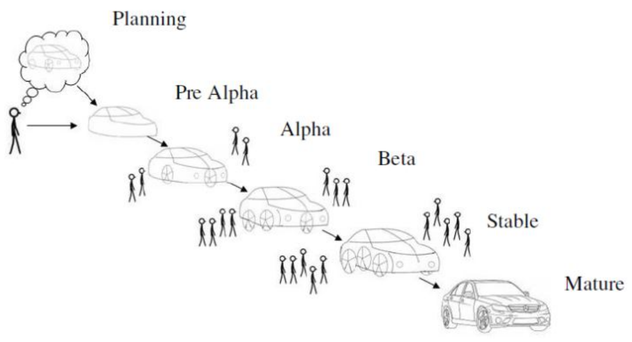


Figure 4.1: proposed OSS process stages

4.7. Steps of developing the proposed model

1. Planning

In this stage, no code has been written, the scope of the project is still in idea. As soon as tangible results in the form of source code appear, the project enters the next stage**.**

2. Pre-Alpha

During Pre-Alpha stage 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.

3. Alpha

After pre-Alpha stage, the next stage is Alpha stage. Here, 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.

4. 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.

5. Stable

During this stage, 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.

6. Mature

Mature stage is the last stage that 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.

4.8. Enhanced Agile Incremental: The proposed process modeling

Major OSS are initiated by some group of people from different organizations in voluntarily basis and contributed by all interested people of different country and culture. It somehow may follow Evolutionary or Incremental Model as it is started from very small unit and gradually increased. But as this kind of software is not managed by a single organization the requirement specification is confusing so that design is dependent on individual perception and skill. But all interested people create an open forum regarding the development procedure where they put different queries and answered by all as possible. Or sometimes the first group of people who takes the initiation is creating an organization and other people join accordingly. There are 11 steps in the newly proposed Open Agile Incremental Model.

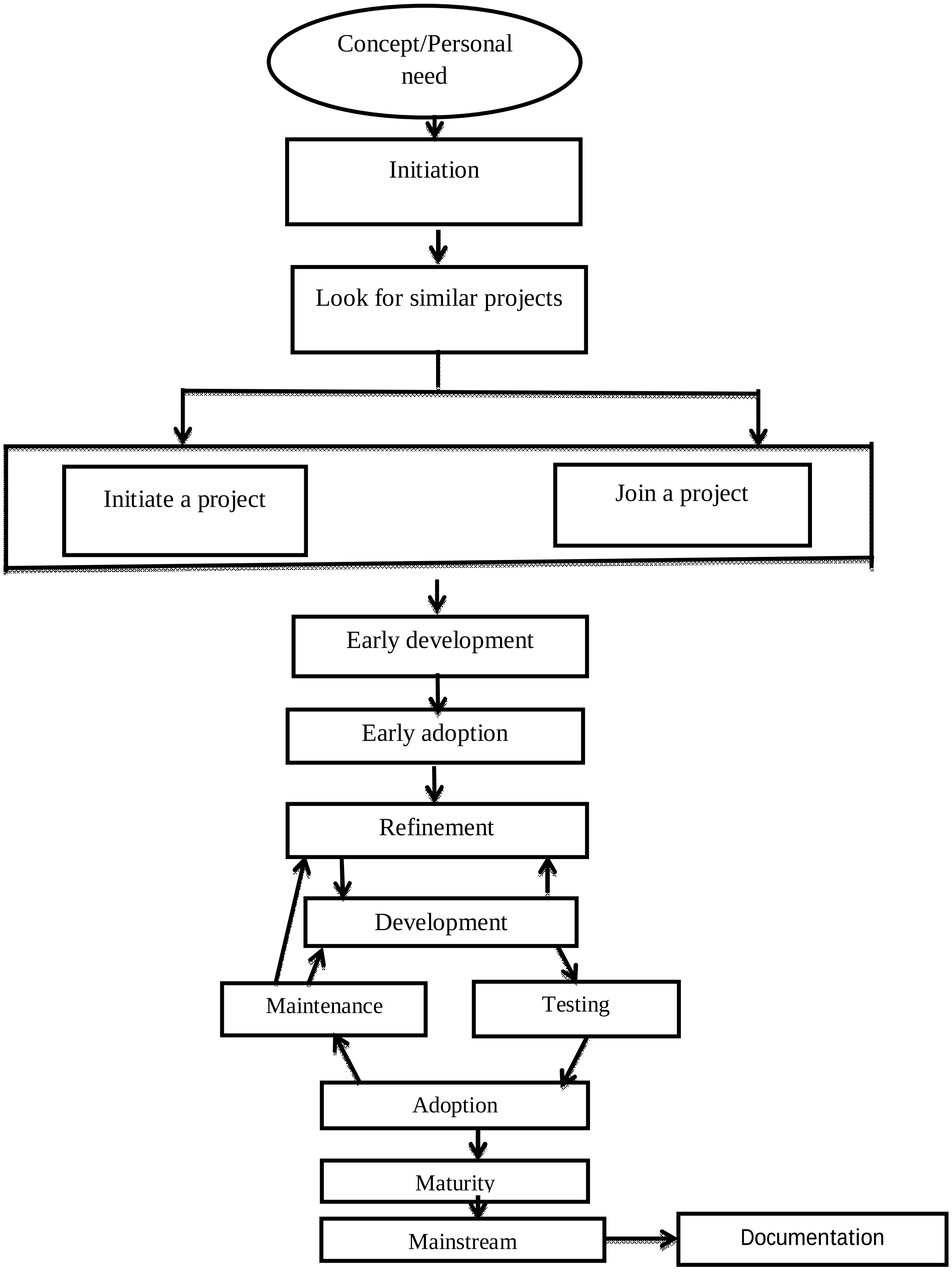


Figure4.2: the proposed model

4.9. Phases of the proposed model

Phase 1:Concept

Some ideas have been thrown around by a few people. Some codes have been written. A project site is up, and there is maybe a working prototype to show off and talk about. There are no such documents placed there. A forum or website can be launched due to this purpose.

Phase 2: Initiation/Bootstrap

There are one or two serious authors/committers. There might already be one or two actual users experimenting with the framework. The developments are continuously going on. Few discussions are going on using some blog posts on that forum or few related articles may be published. A controlling authority can be formed who guides or control the development process and program code contributed from various people.

Phase 3: Early development

A few more committers have signed on, but still there are only one or two primary contributors. A few more users have started playing with it now and are providing feedback. A mailing list has certainly been set up. The build is more stable and works in many environments.

Phase 4: Early adoption

More committers are signing on now and contributing more and more. The project is at a state with many of the baseline features are done and the product is quite usable now. There are a few dozen users experimenting with it and even building some interesting applications upon it. Multiple people are discussing about it and contributing a bit. The forum/authority is starting to materialize the discussion into categories and sub-topics. Some sparse documentation has started to emerge in the form of some limited API documents; some FAQ‘s and getting started guides. So after this phase all contributors get a piece of software and concrete guidelines. The tasks carried by controlling authority are increased. Because now they have to maintain the changes and modification made by different contributor throughout the world.

Phase 5: Refinement of the concept / Specification

In this phase based on the feedback and development scenarios the actual concepts may be refined and displayed as guideline. The prototype or very first version of OSS is now made. So any modification or refinement of the original concept is done. The specification guidelines are very concrete in this phase. This does not mean that the modification of the concept are not done later phase. Any good suggestions can lead to a new version of the OSS at any phase.

Phase 6: Development

There are several devoted committers/contributor now, cranking out serious features and functionality. The beta versions are distributed for free. There are several users using the OSS and post their feedbacks or queries on the forum's wall/blog. The codes are available for download regularly. Some restrictions may be imposed or not to get it. The code is growing by all. It is really taking shape and being fleshed out. At this point documentation is moderate and is growing into more scenarios and topics.

Phase 7: Testing/ Validation

The developed code of single contributor is tested individually by the developer and then uploaded to be added with the main program code. The authority may test it again after integrating with the main code. If some modifications required the codes are again send to the actual developer or published in the forum mentioning the bugs so that contributor may debug it accordingly. The forum can take help of the experts also.

Phase 8: Adoption

Many people are submitting patches and other forms of contributions. The inner circle of committers is growing. There are frequent point releases available for download. There is now likely a basic installer which helps configure your environment for the framework. Frequent, in-depth blog posts by noted authors. The wiki (A web site or set of web pages that allows almost anyone to edit and add content) is now very rich and there are frequent contributions. Documentation at this point is adequate. The codes are now packaged and are distributed for free or with very nominal cost.

Phase 9: Maintenance/ Support

After adoption of the OSS now it is time for maintaining the software. Maintenance can be many types. Corrective: Correcting errors that were not discovered during the product development phase. This is called corrective maintenance. The corrections are made available to the users for free or as new release. Perfective: Improving the implementation of the system, and enhancing the functionalities of the system according to the user's requirement. This is called perfective maintenance. Adaptive: Porting the software to work in a new computer environment/platform or to cope up with new kind of operating system. This is called adaptive maintenance. It is a time consuming process and there is no exact guidelines for that. Any type of maintenance may be required any time and may be raised by any user or developer any time.

Phase 10: Maturity

The maintenance phase may add a new kind of directions and may lead to another phase of development and sometimes refines the requirement. The contributors again start to develop another version of the OSS. The concepts are put in the wiki and guideline also. This process cycle may run for several years or decades and slowly moves to different levels of maturity. There are so many patches and contributors that a hierarchy has been set up to evaluate and approved changes. Multiple releases create management issues with patches and defects requiring more organization. There over a thousand users by now. There are releases and installers for various scenarios or environment. There is lots of documentation activity at this phase.

Phase 11: Mainstream and documentation

In this phase, many of the original contributors have moved on or are less involved. New generations of contributors have taken over and are taking the project into different directions and expanding it greatly. There are hundreds of thousands of users. Perhaps there are even consulting companies forming business services around the project and offering commercial support. The project has its own active web site with lots of content, guides, add-ons, forums, blogs, etc. Finally the model is documented by the project initiators.

4.10.Characteristics of the proposed model

The proposed model has some characteristics. The model is designed with individuals collaborated with others. It follows modularity designing, iterative and incremental works, with these characteristics; the proposed model follows combination of Agile and Incremental methodology. So the model is Agile Incremental.

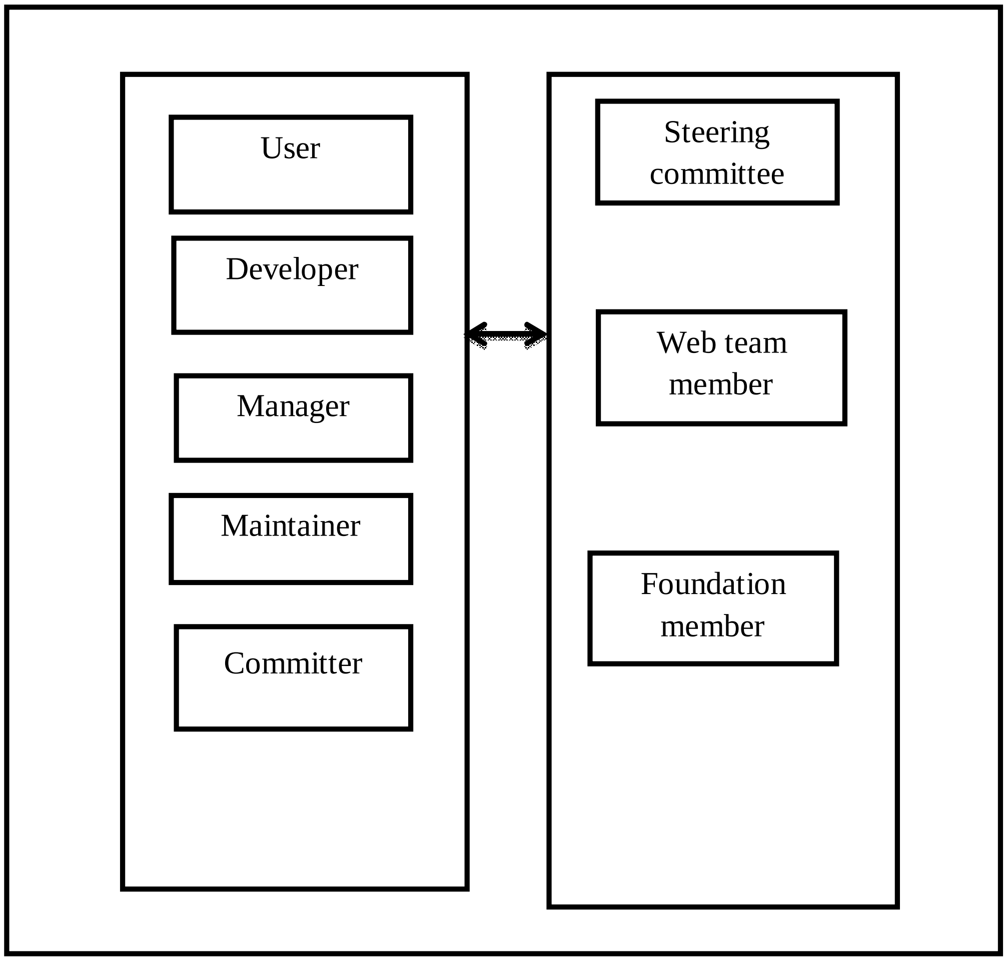


Figure 4.3: Definitional level of the Proposed Model

Definitional Level description of the model

User: that contributes to the project and downloads the release. The user also documents the project

Developer: develops, reviews and test codes.

Manager: manages the project web site and releases process

Maintainer: Manges the project code repository and and package releases.

Comiter: commites codes in the code repository

Steering committee: steers a project, creates new project and manage the existing project

Web member: manages the web portal

Foundation member: manages the fundation.

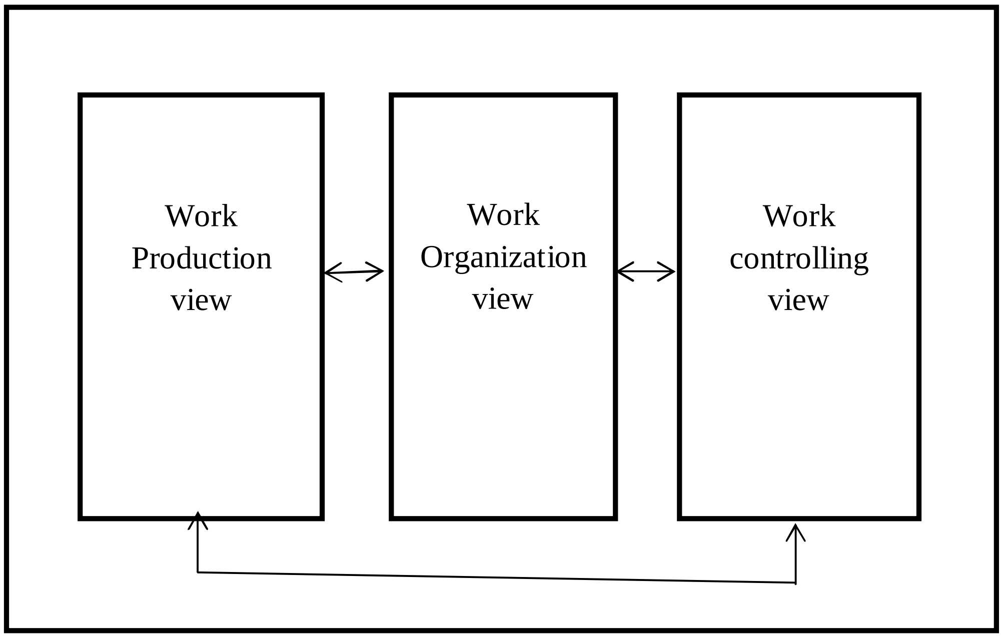


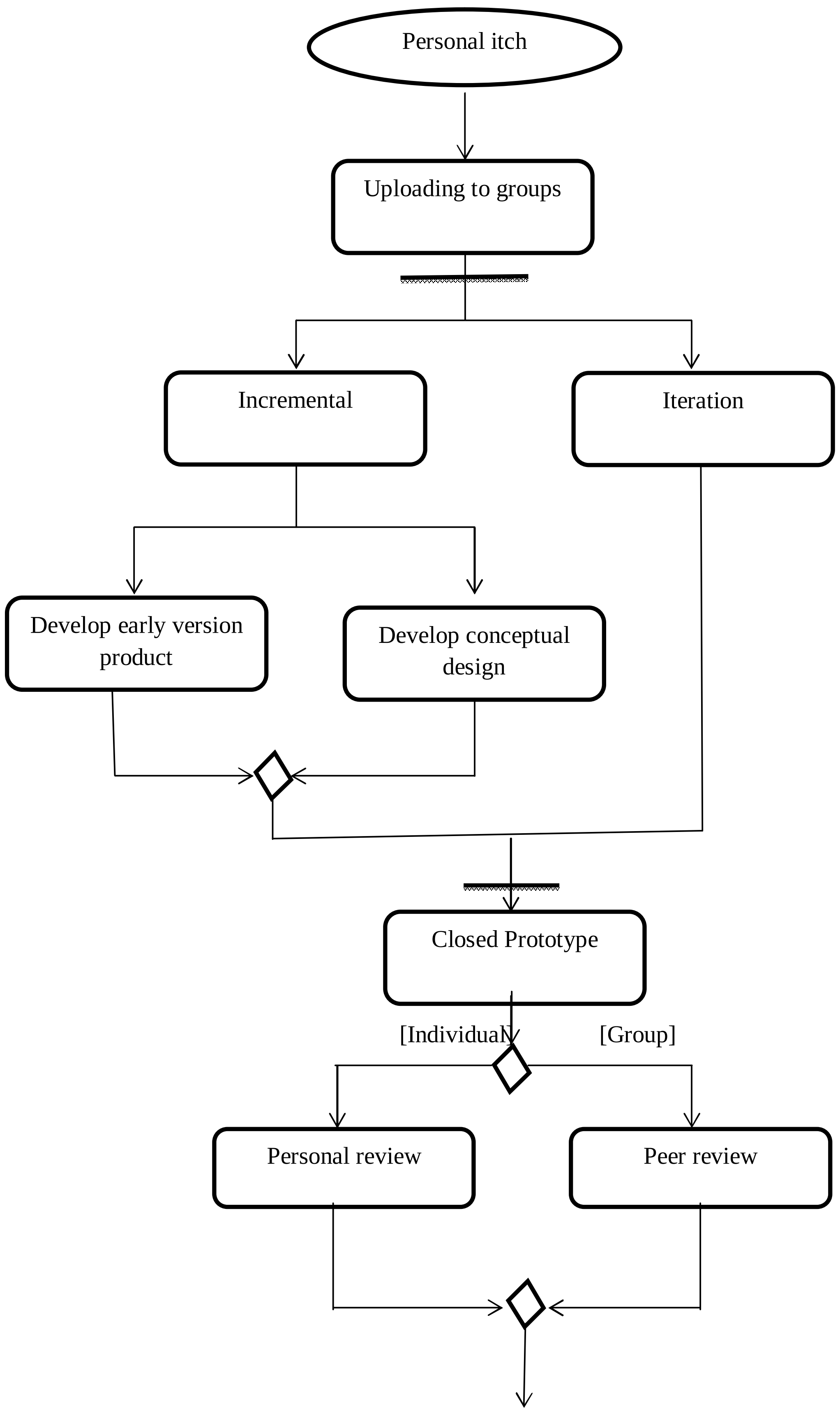
Figure 4.4: Proposed Model development views

Development view description

Work production view: This includes both tasks and states of the product relating to design, coding, and testing.

Work organizational view: This view addresses the social aspect of development. This includes factors relating to communication and coordination, key roles and responsibilities, and motivation.

Work control view: focuses on direction. More specifically, itdeals with mechanisms for guiding development. This includes planning, approval, data gathering, support, and documentation.


Personal itch
Uploading to groups
Incremental
Iteration
Develop early version product
Develop conceptual design
Closed Prototype
[Individual]
[Group]
Personal review
Peer review

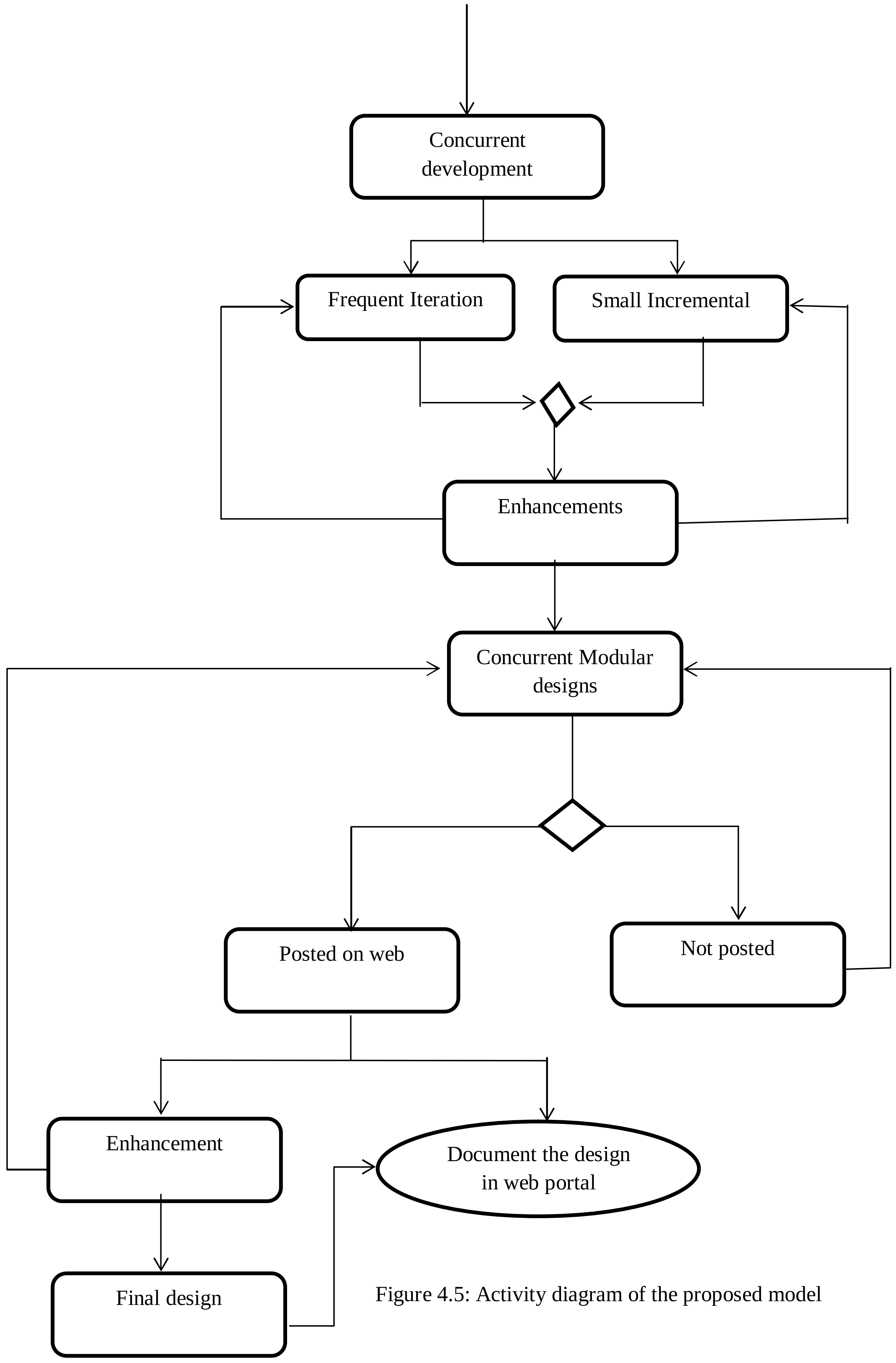


Figure 4.5: Activity diagram of the proposed model

4.11. Validation of the proposed model

By studying some OSSDLC models [27] we can conclude that all models are having different characteristics but basic features are similar. All other existing models are based on the basic priority of OSSD and useful. They also comprise different advantages of OSSD. Also it is very tough to decide which one is better and what are the relative disadvantages. Because of lacking regulation and discipline among developers it is not very easy to follow any OSSDLC models truly. Also if we categorize the software according to Bohem’1982 [28][29] there are three types of software, *Organic, Semi-detached and Embedded*, according to size, complexity and resource required for development. Only first two types of software can imply any OSSDLC easily. Also on the basis of the size and complexity of the software some models may be proved useful and efficient than others. Without proper application of various OSSDLC on different OSS development worldwide and surveying thoroughly the effectiveness of different components cannot be estimated and compare usefulness. With the already existing models, described in several research papers and case studies; there are some advantages of using the proposed model. See below the comparison between existing models and the proposed ones.

Table 4.2: comparison between existing models and the proposed ones

Metrics	Existing Models	The Proposed Model
quality	OSS features result in quality software. Also There is little tolerance for failure to adhere to the tacitly accepted norms [30][31].	Refinement of Concepts or Specification helps out to achieve best solution and thus increase quality
Speed development	Reuse of code increase development speed. The more people are creating code and adding value to a project, the product is released quickly and it becomes valuable to a user group.[13]. Critics question whether open source provides a rapid development environment and suggest that the result could be slower given the absence of formal management structures. The open source community is likened to a large, semi-organized mob with a fuzzy vision [32] [31].	My proposed model requires a very good project initiation and control and thus eliminates the problem of absence of formal management structure or irregularities if any.
Collaboration	A further important feature of the OSSD model is the nature of the development community. Large numbers of geographically dispersed programmers are joined by the Internet to produce complex software and largely without pay. Reasons for participation in open source projects are mainly due to lots of challenge, improving skills, motivated wish for human welfare, fun, as well as for financial reward [30]	The proposed model ensures collaboration in large in all phases.

Metrics	Existing Models	The Proposed Model
Releases	OSS is premised on rapid releases and typically has many more iterations than commercial software. This creates a management problem as a new release needs to be implemented in order for an organization to receive the full benefit. It is very tough to decide for organizations whether these newer versions will continue to support business needs [30].	The model is having a new release all the time when it completes the Adoption phase. The organizations will use and suggest improvements at general Support or Maintenance phase. That will definitely lead to newer kind formal requirement or concept and initiates another cycle of development.
Support issues	Wheatley [34] mentions the lack of accountability from a single vendor. While open source projects have a wide variety of resources (developers themselves, Internet mailing lists, archives and support databases) that can be tapped for support, the problem is that there is no single source of information, no help desk that provides definitive answers to problems. Open source developers are not contracted and therefore cannot be forced into creating documentation [32]	The Refinement of Concept or Specification phase of the proposed model leads to documenting the requirements and modification history. Also the Validation and Adoption phase may generate high quality review report that can be used in future development or release. The developers also can do documentation with their coding to increase their code acceptance.

CHAPTER FIVE

CONCLUSIONS AND FUTURE WORKS

5.1. Conclusion

Open source movement is a social movement. All people of the world must be blessed with the advancement of technology to improve their lifestyle. Technology must not be in doomed in some companies for their business profit. With that, it must be kept in mind that the developed software must be adequate to fulfill the requirement of the new technological advancement of the user. An accepted standard model must be there for the development of OSS, so that the developed software becomes usable for a large number of people scattered in the whole world, not to a limited group of people. Research must go on, on the open source development life cycle to find well accepted model, and so that user can get standard software for their uses.

With this reason, the paper focuses on process modeling in open source software developments. Many papers are addressed with in this context and review the papers especially in process modeling.

After reviewing the documents,methodologies are designed to address the statement of the problem. Systematic research methodology with thematically addressed technique is used for document reviewing. For the methodology, open software users in the web and software developers in Addis Ababa are included in the questionnaire.

Finally, enhanced Agile Incremental process model is designed for open source software developments.

This model tries to address the problem of documentation and reusing models again in the open sources software development process with in its limitations.

5.2. Future Works

The proposed model will not be clear with its advantage and disadvantage, unless the model is used for development of some software project. In a future work it is better to use this model for some Open Source Software development to find its strong and weak features. Also better to find the complexity of development, development speed, number of bugs present, best testing methodology etc. of an Open Source Software developed by this model.

The research indicates that, there is no well-defined and accepted model for the OSS process developments. So further study shall be proceed especially, documenting and managing the process as whole.

REFERENCES

[1] J. Feller, B. Fitzgerald, S. A. Hissam, and K. R. Lakhani, Perspectives on Free and Open Source Software. The MIT Press, 2007.

[2] S. K. Sowe, Emerging Free and Open Source Software Practices , 1st ed. IGI Publishing, 2007.

[3] K. Schwalbe, Information Technology Project Management (with Microsoft Project 2007 CD-ROM) , 6th ed. Course Technology, 2009.

[4]. J. Feller and B. Fitzgerald, Understanding Open Source Software Development . Addison-Wesley Professional, 2001.

[5].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

[6]. Rinette Roets, Marylou Minnaar, and Kerry Wright, (2007) Open source: Towards Successful Systems Development Projects in Developing Countries, Proceedings of the 9th InternationalConference on Social implications of computers in developing countries, Sao Paulo, Brazil, May2007.

[7]. C. DiBona, S. Ockman, and M. Stone, Open Sources: Voices from the Open Source Revolution. Sebastopol, CA: O'Reilly, 1999.

[8]. S. McConnell, “Open-Source Methodology: Ready for Prime Time?,” IEEE Software, vol. July/August, no. 4, 1999, pp. 6-8.

[9]. P. Vixie, “Software Engineering,” in Open Sources: Voices from the Open Source Revolution, C. DiBona, S. Ockman, and M. Stone, Eds. Sebastopol, CA: O'Reilly, 1999, pp. 91-100.

[10]. B. Perens, “The Open Source Definition,” in Open Sources: Voices from the Open Source Revolution, C. DiBona, S. Ockman, and M. Stone, Eds. Sebastopol, CA: O'Reilly, 1999, pp. 171-188.

[11]. B. Rosa, “Proprietary Software vs FOSS.” [Online], Dissertation (1 essay),*IPR University Center*, Spring2012, available: www.iprinfo.com [feb 15,2017].

[12].Somerville, 2004, software engineering (9thedition).[On-line]. Available: http://www.SoftwareEngineering-9.com/Web/Cleanroom[Feb, 17, 2017]

[13]. Open Source Licence Proposa http://cio-nii.defense.gov/sites/oss/ Open_Source_Software_(OSS)_FAQ.htm

[14]. Beck, Kent; et al. "Manifesto for Agile Software Development". http://agilemanifesto.org/. Retrieved 2010-06-14.

[15] Cockburn. A. “Agile Software Development”, Addison-Wesley, 2002.

[16] Beck, Kent; et al. "Manifesto for Agile Software Development". http://agilemanifesto.org/. Retrieved 2010-06-14.

[17] DeMarco, T., Boehm, “The Agile Methods Fray”, IEEE Computer, Vol 35, no 6 June 2002, pp 90-92.

[18]. D. E. Wynn, “Organizational structure of open source projects: A life cycle approach”, presented at the Proceedings of the 7th Annual Conference of the Southern Association for Information Systems, Savannah, **(2004)**.

[19]. A. Mockus, R. Fielding and J. D. Herbsleb, “Two case studies of open source software development: Apache and Mozilla”, *ACM Transactions on Software Engineering and Methodology, vol. 11, no. 3,* ***(2000)****, pp. 309-34).*

[20]. Open source License Proposal” http://cio nii.defense.gov/sites/oss/open_source_software _ (OSS) _FAQ.htm”

[21]. M.-W. Wu and Y.-D. Lin, “Open source software development: an overview”. *Computer, vol. 34, no. 6,* ***(2001)****, pp. 33-3.*

[22]. N. Jorgensen, “Putting it all in the trunk: Incremental software development in the Free BSD open source project”, *Information Systems Journal, vol. 11, no. 4,* ***(2001)****, pp. 321-336.*

[23]. R. Roets, M. Minnaar and K. 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,* ***[May,2007)****.*

[24]. W. Scacchi, "Open Source Software Development Processes, version 2.5(online" http://www.ics.uci.edu/~wscacchi /Software Process/Open-Software-Process-Models / Open-Source-Software-Development-Processes.ppt, **[2002]**.

[25] J. O. Gilliam and L. Gazette, "Improving the Open Source Software Model with UML Case Tools", *vol. 67,* [*Jun, 2001*]).

[26]. W. Scacchi. (2001, Feb.). "Process Models in Software Engineering." John Wiley and Sons, Inc,.[On-line].27(3). Available: www. *E*ncyclopedia *of Software Engineering* [Feb,17, 21, 2017].

[27]. Z.Kamal "process modeling languages: a literature review."*Malaysian Journal of Computer Science {on-line}. 14(2), pp.26.available: email: kamal_zamli@hotmail.com{Feb,18, 2017}*.

[28]. I, Mohd."Modeling the open source software development processes using IDEF3 standard"Thesis *Submitted in Fulfillment of the Requirements for the Degree of Master of Computer Science, [Jan, 2013*].

[28] Beck, Kent; et al. "Manifesto for Agile Software Development". http://agilemanifesto.org/. Retrieved 2010-06-14.

[29]. Pressman,Roger S, "Software Engineering", McGraw Hill, pp79-88.

[30] Rishab A. Ghosh, Bernhard Krieger, Ruediger Glott, Gregorio Robles, "Free/Libre and Open Source Software: Survey and Study" , International Institute of Infonomics University of Maastricht, The Netherlands June 2002

[31] Valloppillil, V. "Open source Initiative (OSI) Halloween I: A (new?) software development methodology" 1998

[32]N. Bezroukov, "Open Source Software Development as a Special Kind of Academic Research (Critique of Vulgar Raymondism)," http://firstmonday.org/issues/ issue4_10/bezroukov/index.html

[33]Tom Adelstein, "How to Misunderstand Open Source Software Development", December 1, 2003, http://www.consultingtimes.com/ossdev.html

[34] M. Wheatley, CIO Magazine, "The Myths of. Open Source", March 2004, http:// /www.cio.com/archive/030104/open.html

Appendices

Research Questions and Answers Given by Web Users

1. **What is Open Source software?**

   Answer:

   Open Source software is software that can be freely accessed, used, changed, and shared (in modified or unmodified form) by anyone. Open source software is made by many people, and distributed under licenses that comply with the Open Source Definition .

2. **Can Open Source software be used for commercial purposes?**

   Answer:

   Absolutely, all Open Source software can be used for commercial purpose (not proprietary); the Open Source Definition guarantees this.

3. **Can we restrict how people use an Open Source licensed program?**

   Answer:

   No. The freedom to use the program for any purpose is part of the Open Source Definition. Open source licenses do not discriminate against fields of endeavour.

4. **Can we stop "evil people" from using the program?**

   Answer:

   No. The Open Source Definition specifies that, Open Source licenses may not discriminate against persons or groups. Giving everyone freedom means giving evil people freedom, too.

5. **What is "free software" and is it the same as "open source"?**

   Answer:

   "Free software" and "open source software" are two terms for the same thing: software released under licenses that guarantee a certain specific set of freedoms.

6. **What is "copy left"? Is it the same as "open source"?**

   Answer:

   "Copy left" refers to licenses that allow derivative works but require them to use the same license as the original work.

7. **What is a "permissive" Open Source license?**

   Answer:

   A "permissive" license is simply a non-copy left open source license — one that guarantees the freedoms to use, modify, and redistribute, but that permits proprietary derivative works.

8. **Can we call any program "Open Source" even if it doesn't use an approved license?**

   Answer

   Please don't do that. If you call it "Open Source" without using an approved license, you will confuse

9. **What is the best open source tool for building an institutional repository?**

   Answer:

   In OpenDOAR can see other software used in the repositories.

10. Will open source software take more market share than commercial ones?

Answer:

Technical support will be key to the success of open source. Hundreds of people will be sitting in a IRC chat room, chatting, discussing, at the same time helping many who are in urgent need of help. I got help through IRC chat.

11. How OSS is typically developed?

Answer:

OSS is typically developed through a collaborative process.

12. How can we evaluate OSS process?

Answer:

OSS process should be evaluated in principle the same way you would evaluate any option, considering need, cost, and so on. In some cases, the sources of informationfor OSS differ.

St. Mary's University

Faculty of Informatics

Introduction

This questionnaire is prepared for the sake of the partial fulfilment of the requirementsfor the degree of master of science in computer science**.** So, through this brief survey, your answers will be helpful in **developing process modelling for open source software** .your response will only be used for survey purpose only.in any case , if you have questions regarding the survey , please call Haimanot Abun at +251911306542(cell phone). Thank you very much for your time and suggestion.

Part I

Put '√' mark in front of the options given below

1. Are you using open source software for your different purpose?

   Yes☐ No ☐

2. Open sources are preferred than conventional for developing software

   ☐ Agree ☐Disagree

3. Process modelling is very important for developing open source software.

   . ☐ Agree ☐ Disagree

4. Is there any process model for open source software that you know before?

   ☐ I know ☐ I don't know

5. If open source software process is modelled like conventional ones, problem of documentation and reusing of codes will be solved.

   ☐ Agree ☐ Disagree

Part II

Please fill the blanks below.

1. What type of challenges do we face during using open source software?

----

2. Is there a well-defined process modelling technique for the development of open source software . if no, what is the reason behind you expect

   ----

   ----

3. Process modelling is necessary during open source software development? If yes what would be your opinion?

   ----

4. What is your opinion using process modelling during software developments?

----

5. Why process modelling of open source software is very important? What is your opinion

   ----

   ----

6. Is documentation is important during development of open source software?

   Why?

   ----

7. Do you agree that, well defined process modelling of open source software is mandatory?--

   Why?

   ----

8. In your view, is reusing of codes in open source software development is easy?

   Why?

----

9. Is process modelling is important during developing of open source software , if yes why?

----

10. What is your general opinion on process modelling of open source software developments?

Thank you for your cooperation