ISSUES ANALYSIS OF OPEN SOURCE SOFTWARE DEVELOPMENT: A SYSTEMATIC REVIEW

Submitted By
Mr. Kareem Ullah
(Reg#226-FBAS/MSSE/F-08)

Supervised By
Mr. Shahbaz Ahmed Khan
Assistant Professor,
Department of Computer Science & Software Engineering,
International Islamic University, Islamabad.

Co-Supervised By
Mr. Saeed Ullah
Assistant Professor,
Department of Computer Science,
Fedral Urdu University Arts, Science & Technology, Islamabad.

DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

FACULTY OF BASIC AND APPLIED SCIENCES

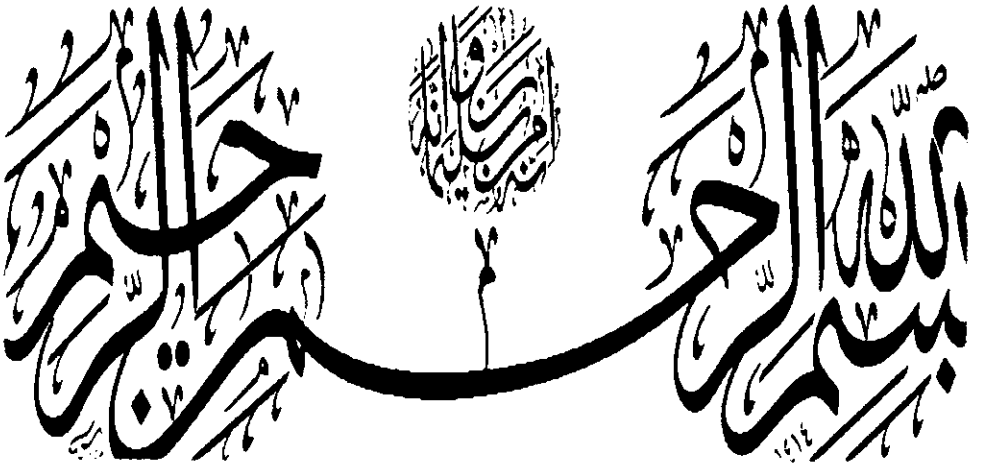
INTERNATIONAL ISLAMIC UNIVERSITY ISLAMABAD

(2011)

Accession No. TH-8387

MS
005.1
KAI

1- Computer Software – Development



With the Name of

Allah,

The most merciful and compassionate the most gracious and benifent whose help and

Guidance we always solicit at eery step, and every moment.

ISSUES ANALYSIS OF OPEN SOURCE SOFTWARE DEVELOPMENT: A SYSTEMATIC LITERATURE REVIEW

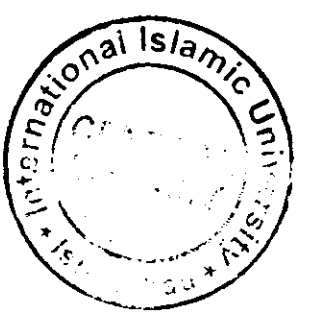
BY

KAREEM ULLAH

REG. # 226-FBAS/MSSE/F08

A Thesis

Submitted in Partial Fulfillment of the Requirements for the Award of Degree of



Master of Science (MS)

In Software Engineering

DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

FACULTY OF BASIC AND APPLIED SCIENCE

INTERNATIONAL ISLAMIC UNIVERSITY ISLAMABAD

(2011)

International Islamic University Islamabad

Faculty of Basic and Applied Sciences

Department of Computer Science and Software Engineering

Dated.....

FINAL APPROVAL

It is certified that we have read the thesis submitted by **Mr. Kareem Ullah** of **Reg. no. 226-FBAS/MSSE/F08**. It is our judgment that this thesis is of sufficient standard to warrant its acceptance by International Islamic University Islamabad (IIUI) for the degree of Master of Science in Software Engineering.

COMMITTEE

EXTERNAL EXAMINER:

Dr. Fawad Hussain

Assistant Professor, FCSE

Ghulam Ishaq Khan Institute Engineering and Technology,

Topi Khyber Pakhtunkhwa Pakistan

INTERNAL EXAMINER:

Mr. Imran Saeed

Assistant Professor, DCS&SE, FBAS

International Islamic University, Islamabad, Pakistan

SUPERVISOR:

Mr. Shahbaz Ahmed Khan

Assistant Professor, DCS&SE, FBAS

International Islamic University, Islamabad, Pakistan

Co-SUPERVISOR:

Mr. Saeed Ullah

Assistant Professor, DCS, FBAS, Federal Urdu University Islamabad

Dedication

To My Loving **ALLAH**, Who has given me the brilliant abilities to achieve the targets.

&

To My Loving Prophet Hazrat **MUHAMMAD (PBUH)**, Who is sent as Ramatulilaalaameen for all worlds.

&

To My Loving **Teachers** who always guided me

&

To My Loving **Parents** and My **Family**, especially to my **Mother**, whose prayers are reasons of my achievements

DECLARATION

I hereby declare that my whole thesis or part of it is not plagiarized from any other source. This is purely my own work. Such type of work has not been published yet by any other individual. This work was not used for any other degree in any university or institute. I wrote this thesis with my personal efforts. If it is found plagiarized from any other source, I shall be responsible for the consequences. Where necessary, I used references and quoted the text of other authors.

Kareem Ullah

226-FBAS/MSSE/F08

ACKNOWLEDGEMENT

There are always unseen hands behind the success of every person. Similar is the case with me. I would like to thank for blessings of Almighty Allah who gave me courage to complete this work. I also thank my friends and fellows who assisted me during the completion of this work. Then I am grateful to my supervisor Mr. Shahbaz Ahmed Khan and Co-supervisor Mr. Saeed Ullah who assisted me in my work as much as possible. Their kind, valuable and timely feedback made it possible for me to do such a work. Without his assistance I might not be able to complete this work in such a manner. He also encouraged me time to time and such encouragement raised my morale and I kept on doing it with patience and consistency. This is all due to my supervisors who directed me so well that I completed my work easily.

LIST OF ACRONYMS

Abbreviation	Title
SLR	Systematic Literature Review
EBSE	Evidence-Based Software Engineering
EBM	Evidence-Based Medicine
OS	Open Source
PS	Propriety Software
OSS	Open Source Software
CSS	Close Source Software
OSSP	Open Source Software Projects
CSSP	Close Source Software Development
OSSD	Open Source Software Development
F/OSS	Free or Open Source Software
SDLC	Software Development Life Cycle
PICOC	Population, Intervention, Comparison, Outcome, and Context
QAC	Quality Assessment Criteria
SMS	Systematic Mapping Studies
SM	Systematic Mapping
SE	Software Engineering
TSD	Traditional Software Development
SR	Systematic Review

ABSTRACT

Context: Open source software development is a valuable area of interest in software engineering. There is a lack of cumulative empirical knowledge about issues of open source software development and their mitigation strategies.

Objective: The objective of this study is to identify the evidence base reported issues of open source software development with their reported mitigation strategies to resolve the issues. Reported issues without evidence had not included in this report. Social issues of OSS systems have also out of scope.

Methods: The underling research questions have answered by systematic literature review process. Search strings have designed first to retrieve the results from electronic databases are included in systematic review. Then selected studies are filtered by applying study selection criteria; designed in review protocol. The quality assessment criteria are applied on the previously selected 71 studies. After applying quality assessment 17 empirical studies are obtained. 17 studies have used to extract data after reading full text of all empirical studies.

Results: The result shows the issues of OSSD and their mitigation strategies. By resolving identified issues; it may be possible to decrease the failure rate of OSSP. According to the survey conducted in European countries, 80% organizations have adopted the OSSD. Selected primary studies are 14 case studies, 2 interviews and a survey. Some issues are reported, but their mitigations strategies are still missing. Missing strategies may be reported in future.

Conclusion: No formal process has found which can be used to overcome the issues. Failure rate of OSS projects can decrease growth rate of OSSPs; therefore to mitigate the issues must be need of time. 34 issues and 13 categories of issues are found. These issues can be mitigated by 11 reported strategies.

Keywords: Systematic literature review, Evidence base software engineering, Open source software projects, Systematic mapping.

Table of Figures

Table of Contents

CHAPTER NO: 1

INTRODUCTION

1. Introduction

This section presents the thesis vision, which is helpful to the viewers. A flawless report is presented about the problem statement, aim of research, research questions, scope of research and contextual background information of thesis. This section will also introduce a research method, which is used to answer the research question. This also describes the reason, why we have selected such kind of research method? At the end, complete outline of chapters will present the quick peep of thesis.

1.1.Background

Open source software initiated in 1950s and 1960s, when the software and hardware both were sold together. During 1950s and 1960s the source code of software was freely available without any restriction in IBM and DEC user group. In 1969 first version of UNIX was written by Ken Thompson, UNIX source code was accessible freely till in seventies. From 1970-2000 the OSS was in progress without a name during these three decades. Many of the successful and broadly used software packages were developed in this period.

Growing rate of OSS projects is increasing quickly. It is estimated that OSS systems are 500,000 users, which are increasing with the rate of 700 per day. The OSS projects are also increasing with the rate of 60 new projects per day. It is estimated by Gartner group that 85% of enterprise companies are adopted the OSS systems and remaining 15% are moving to adopt in the next few years. A survey is conducted about the usage of OSS, which reports that in 13 European countries the usage rate of OSS is 78%. Survey is also conducted in US and reported that 87% of organizations are using the OSS (Kareem Ullah, Shahbaz Ahmed Khan, 2011).

In late 1997 and early 1998 group of people that are attracted in dispersal consciousness of erudite tools which are used instead of proprietary software development, and voted to propose the term of open source. The Christine Peterson is the person, who has proposed the name of open source (David Bretthauer, 2001).

The participants that participate in OSSP are highly motivated, therefore high quality is expected. According to the philosophy of OSSD it has many promises as, to produce high quality software systems. The development activities like testing and documentation are almost ignored. Enough attention has not paid for clarify system requirements and design

(Ioannis Stamelos, Angelis, Lefteris; Oikonomou, Apostolos; L. Bleris, Georgios ;, 2002). As talking about promises, OSS systems are faster, cheaper and reliable then property software development. OSSD has also face some challenges as "lack of a formal process, poor design and architecture" (Trung Dinh-Trong, James M. Bieman).

Purpose of open source software systems is to provide the independencies in order to use the system for any purpose. The word open source is representing the software development process that depends on the physically dispersed developers, which are connected by internet (Kareem Ullah, Shahbaz Ahmed Khan, 2011).

Life cycle of OSS projects development is fairly different from the CSS projects (Sandro Morasca, Davide Taibi, Davide Tosi, 2009). There is a great difference between Open source software development processes and commercial software development processes as similar to spiral model. There is no distinct Open source processes exist. Processes in OSSD are different from project to project (Jesper Holck, Niels Jørgensen, 2003).

1.2. Problem identification and Motivation

Many claims are found which directed to explore the area of open source software development. On the basis those reported claims there are greater increasing rate of conducting research in OSSD. Few researchers have some claims to support the underlying research. According to them, it's the need of time to critically mitigate the issues of OSSD. According to Andreas Bauer open source software development might not be the "silver bullet" of software engineering because; it also has some shortcomings and troubles (Andreas Bauer, Markus Pizka, 2002). F/OSS has faces many "serious challenges" that have directed the software engineering community towards ultimately failure (Brian Fitzgerald, 2004).

Even with lot of technological successes, OSSD has also faces lot of "fundamental challenges". OSS systems have been criticized due to these challenges (Morten Sieker Andreasen, Henrik Villemann Nielsen, Simon Ormholt Schrøder, Jan Stage, 2006). It would be tremendously profitable to improve the design and architectural support for OSSD and limitations of the OSSD methodology (Vijay K. Gurbani, Anita Garvert, James D. Herbsleb, 2006).

There is a shortage of cumulative information about issues of OSSD. Such type of work does not exist in the literature which can decrease the failure rate of OSSP's and increase the

knowledge of researchers and practitioners to choose mitigation strategies in OSSP development.

1.3.Aim of Research

The aim of research is to find out Issues of OSSD, which are addressed from literatures. What are reported issues of OSSD? In this question we will also find out reported strategies for reported issues. No new strategies will be made. Some issues are reported without mitigation strategies, these types of issues are calling open issues. We also have some open issues.

1.4.Scope of Research

Only issues facings in the development of Open source software projects will be part of this research work. The issues of proprietary software development will not be included, because those issues were quite different from OSSD and out of scope. We will include only those open source software development issues which are evidence-based. We will collect only few types of evidence like field/case studies, survey, experts' opinion and interviews. The other types of study that were not provided the empirical evidence will not be included in this thesis.

1.5.Research Question

RQ 1) What are the reported issues in open source software development?

RQ 2) What strategies are reported to deal with these issues?

1.6.Research Method

We required only one research method to answer these both research questions because these research questions can be solved by evidence-based solution. Evidence base method (systematic literature review) is used to answer the questions. Our research questions are of similar type as described by Barbra Kitchenham in her guidelines (Barbara Kitchenham & Charters, 2007). Therefore, to address evidence-based research questions in the area of software engineering; systematic literature review is the most appropriate research method.

Instead of SLR other research methods i.e. surveys, case study, ethnography, simulation, experiment, benchmarking are not appropriate to conduct such kind of research because these methods are not provide evidence-based solution. This research question needs a research

method that should be provided evidence-based answer. Therefore, appropriate research method for this kind of research work was systematic literature review.

Thesis Outline

In this sector a brief detail of next chapter of thesis is describes in the thesis. The complete body of thesis is also described that will increase readers understandability.

Chapter 1: Introduction

This chapter introduces the thesis purpose in detail. It delivers a clear explanation of problem statement, research questions, aim of research, and scope of research, research method and complete information of background.

Chapter 2: Open Source Software Development

This chapter has a purpose to introduce the importance of open source software development in software engineering (SE). It will also provide the concept of open source software development methodology and then types and roles of open source software projects. This will help to increase the understandability about OSSD. These roles are explained according to the activities of OSSD.

Chapter 3: Systematic Review Process

This chapter describes the theory of evidence based software engineering (EBSE). This chapter also describes the complete process of systematic literature review including three stages and particular events of these stages. The concentration of this chapter is about the first stage of systematic literature review i.e. planning the review. The purpose of this chapter is to describe the need for systematic literature review and review protocol. Therefore complete detail of review protocol is described. Some of the threats of validity of review protocol are also described.

Chapter 4: Conducting the Review

The concentration of this chapter is about execution of review protocol which is the second stage of SLR. This chapter describes the out comings of all steps that are incorporated in the execution of SLR. These steps are; identification of research, commissioning the review,

study selection, study quality assessment, data extraction and synthesis. In the end, the outcomes of the systematic literature review are accumulated. Then our purpose of study should be fulfilled

Chapter 5: Conclusion

The concentration of this chapter is to cumulative the thesis findings. The results of SLR are described and concluded. The limitations of conducted research are also reported. Then future directions are suggested for future work.

CHAPTER NO: 2

OPEN SOURCE SOFTWARE DEVELOPMENT

2. Open Source Software Development

The basic features of OSS systems are accessibility and freely distribution of software source code for any purpose without any restrictions. It is a software development methodology that has increased rapidly in 1980's from BSD Unix and FSF GNU projects. Due to accessibility of internet over the world, OSSD has increased. Through internet the participants of OSS systems have collaborated across the world at OSSP's (Greg Madey, Vincent Freeh, Renee Tynan, 2002).

OSSD is actually depend upon the volunteers, the contributor rarely meet face to face and connect through internet. While is PSD the contributors are paid member of team. The word of open source is representing the software development process that depends on the physically dispersed developers, which are connected by internet. Accessibility of source code is the main characteristic of OSSD.

There is a strict requirement for OSS that is its source code is freely available for everyone. That can be used for any personal purpose. The user of the OSS is always free to modify, or improve the system for personal use. Physically distributed developers are connected on internet in OSSD, but it's not a strict requirement. Lot of companies and individuals has developed the software source code with (PSD) pattern. We can take an example of Netscape web browser (Mozilla project) (Kareem Ullah, Shahbaz Ahmed Khan, 2011).

2.1.Comparison of OSS and CSS Projects

As talking about the differences about OSSP and CSSP, first we have to differentiate the terminologies (OS, OSS, and OSSP). The term "open source" has pronounces the practices of OSSD, which are development approaches and licensing. The term "open source software" has pronounced the software products developed with these practices. According to the ISO 9000 these terms are as following.

"An OSSP is a temporary process to create unique software. Originally, the term Open Source referred to the users' right to get the source code of the software. The Open Source Initiative (OSI), which is a non-profit organization, launched the term in 1999 and its purpose is to promote OSS. The initiators of OSI were Bruce Perens and Eric Raymond" (Timpo Koponent, 2007).

Some more variances are also described, which has increased the understanding about open source paradigms. These variances are; In OSSP's hundreds or thousands of participants

involved. There is no work assigned as CSSD. Participants themself selected their work. There is no system level or detail level design, project plan and schedule exist (Audris Mockus, Roy T. Fielding, James Herbsleb, 2000).

The similarities between open source and free source are the two broad categories which are "copy left" and "non-copy left" software's. Non copy left (public domain software's), which has also named as "not copyrighted" directed towards the permission for source code usage for any purposes. Addition of more limits is allowed in modified version of source code. No additional limits are included in modified version of copy lefted software's. Copy lefted (*GNU GPL*) software has been available freely after modification.

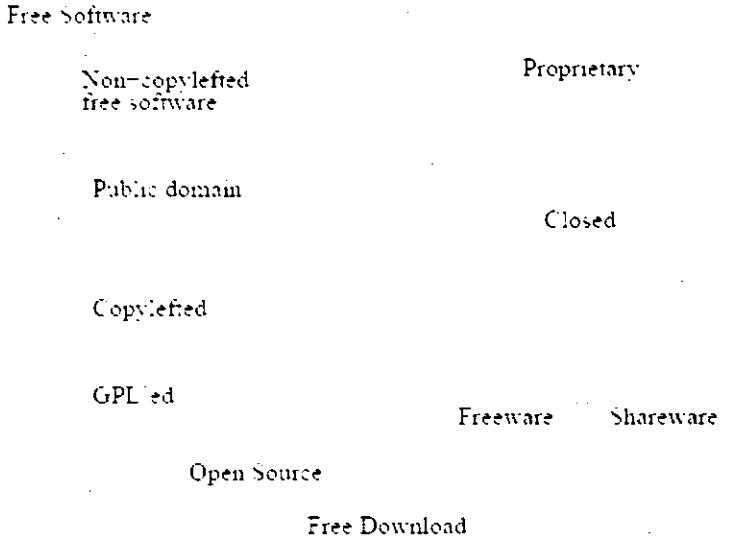


Figure 1 (Cristina Gacek, Tony Lawrie, Budi Arief, 2004)

Now comes to the PS (close source), which are not accessible freely. PS has restricted terms for usage and distribution. There are two sub categories of PS, which are shareware and freeware. These two categories allowed the use and redistribution of software freely. But its modification cannot be possible, because its source code is not released with executable binary. Freeware and free software's are confused phenomena's. As elaborating freeware, software's can't be modified. But users of shareware have to pay license fee and it can be modify (Cristina Gacek, Tony Lawrie, Budi Arief, 2004).

2.2.Roles in OSS Project development

These are some roles of participants in OSSP development. These roles describe the participation of each participant as:(Marina Marinela Gerea, 2006)

- The owner is initiator of an OSSP, starts the project and has authority to include or exclude the individual participation in OSSP
- Core developers are a group of people who develop the source code and take decisions of OSSP
- Developers who are wider group as compare the core developers are the participants who have remove the bugs from project
- Problem reporters are the larger group of participants, who reported the defects to project team
- System testers are the participants who test the project after each enhancement
- User support performs this task freely and providing replies the queries of project users
- Users are the organizations, who use the OSS systems

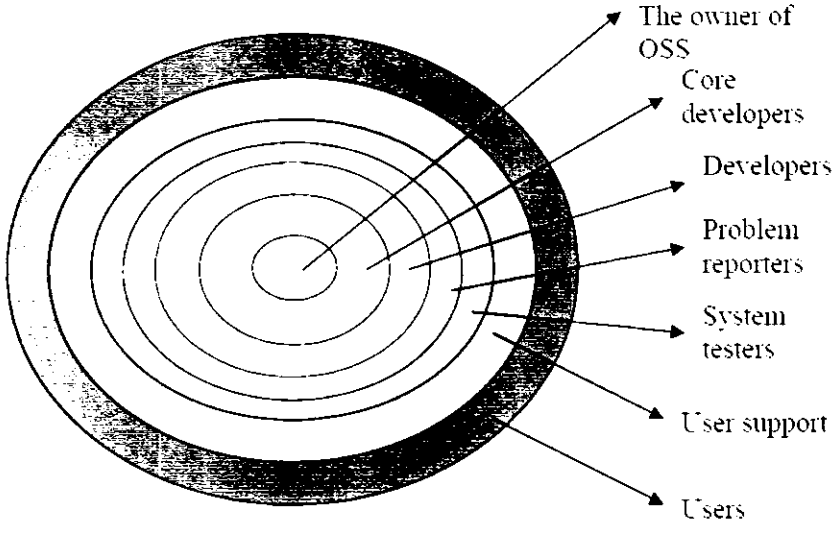


Figure 2 (Marina Marinela Gerea, 2006)

2.3.Initiation of OSS Projects

We have described the initiation of few successful OSSP. This will increase our understanding about initiation of OSSP. The Initiator of "Linux" is "Linus Torvald", who has got graduated from University of Helsinki. He has frustrated from DOS/Windows and wanted

to use a better operating system for his laptop. Initially he has use "Minix" which is developed by Andrew Tanenbaum, but frustrated again.

Therefore in 1991 a message on Minix Usenet news group is announced to invite the interested participants for developing the new operating system. Apache is a successful OSSP, started as http. Http is developed by Rob McCool in NCS at university of Illinois Urbana. In 1994 Rob McCool left NCSA. Then Brian Behlendorf and Cliff Skolnic organized a central archive and submit the patches to archive. Therefore it is claimed that Apache has named by the fact "a patchy server". The code of Apache is written from scratch.

Postgres is developed in 1985 by Michael Stonebreaker. In 1985 Michael Stonebreaker wants to remove the problems of database management system. Therefore Postgres is initiated. In 1994 new version of Postgres is developed by Andrew Yu and Jolly Chen. SQL language interpreter is added and as a result Postgres95 was developed. Postgres95 source code is freely available and released under Berkeley open-source license.

Alice is initiated at University of Virginia by Randy Pausch to automate development of VRS. Randy Pausch continues the project and brings it to Carnegie Mellon University, in 1997. In 2000 "Alice" 2.0 new version has changed its purpose due to forking. Now "Alice" has become a tool for teaching computer programming (Conlon, 2007).

2.4.Type of OSS Projects

According to OSSP development goals, an OSSP is divided into three types. These three types of OSSP also have some differences in nature.

2.4.1. Exploration Oriented

These types of OSS systems (GNU software) are freely available. Due to free access it generates the ideas to develop something new by enhancing the previous participation. Therefore it is also required to maintain the high quality. When this type of project has released then lot of developers join it for learning. These developers are often expert programmers of OSSP.

2.4.2. Utility Oriented

These types of OSS projects are developed to provide the unwavering and vigorous service to community of OSSD. Group of core members of OSS systems collectively manage the

control of OSSP. The changes are first discussed in front of this group, which is called a council. No individual can control the whole project. A trusted contributor of OSS community can be become the part of council.

2.4.3. Service Oriented

Developer of OSSP's can select the project according to their need. These differentiations play an important role for project initiation. Exploration-Oriented OSS has to care about quality of source code to attract the participants, so that more and more developers join the project. To avoid the breakup of project due to forking, leader of project has to communicate with community and adopted changes. There is no fear about forking in Utility Oriented OSS projects, and they require coordination of community to fulfill the needs. Service Oriented OSS projects have created a social mechanism. Enhance the previous version of OSSP (Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, Yunwen Ye, 2002).

2.5.Difference Between OSS Project

These types of OSS systems are developed, when developers wanted a new program to fulfill their needs. Operating system Linux is a great example of utility oriented OSS system. The developers of this kind of OSSP have put it for public use. The need base OSS projects are initiated under this type of OSS projects, to resolve the problem.

CHAPTER NO: 3

SYSTEMATIC LITERATURE REVIEW

3. Systematic Literature Review

This chapter describes the association between systematic literature reviews, systematic mapping and Evidence-Based Software Engineering. It makes easy to understand the importance and need of EBSE. This chapter concentrates on planning stage of SLR. It describes the actions of planning stage of systematic literature review.

3.1.Evidence Based Software Engineering

The researchers require adopting some new research methodology or technology. For this purpose they require appropriate empirical evidence for adopting a new research technology, and by this way they can become successors. There are few constraints in this adoption for new research technology due to unavailability of empirical evidence of specific research area in literature. It shows that existing research methods does not solve some industrial problems. To mitigate these issues we need a research method that can remove this gap. The proposed solution for this problem is Evidence-Based Software Engineering (EBSE); which can enhance the knowledge of researchers and practitioners to choose solution for specific phenomena.

This can be happened by integrating the best existing empirical evidence in the literature of related field. This kind of cumulative knowledge should be useful for the researchers and should increase their level of confidence to adopt the technology. In ICSE'04 EBSE was designed by B.A. Kitchenham. EBSE is based on the Evidence-Based Medicine (EBM) which emphases on the combination of practical evidence in clinical research. The consequence of EBSE is apparent as software-intensive systems are used all over the world; in cars, radars, microwave, electric trains etc. Evidence-based research method also has some other purposes like education, social policy and psychiatry (Kitchenham, & Jorgensen, 2005; B. A. Kitchenham, Dyba, & Jorgensen, 2004).

3.2.Systematic Mapping Process

It is a research methodology that is common in medical research but in SE it's still ignored. This is all due to unawareness of method and non-existence of guidelines to apply it in SE. Complete structure of articles belonging to the same area have been categorized and visually summarized. It is suggested that to conduct systematic mapping studies in those software engineering areas, where enough relevant and high quality primary studies have not found

(Kitchenham & Charters 2007). Figure3 shows the 5 essential steps of systematic mapping with possible outcomes. Final outcome of last step is the systematic mapping. The main goal of SM is to provide overview of research and identify the quantity and type of research in interested area. It can be possible to analyze the trends of publications over time in a specific area (Kai Petersen, Robert Feldt, Shahid Mujtaba, Michael Mattsson, 2008).

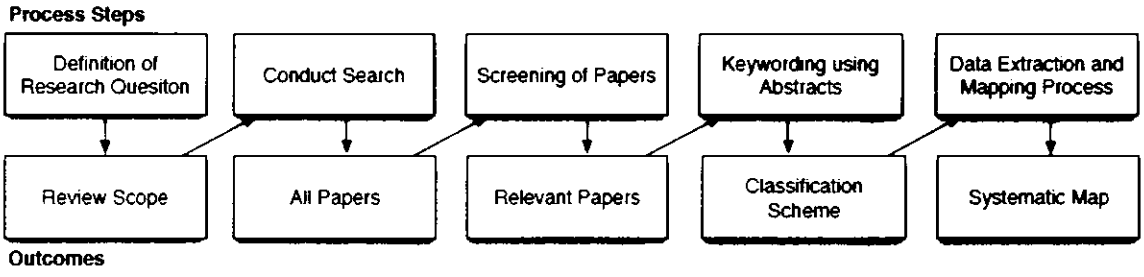


Figure 3 (Kai Petersen, Robert Feldt, Shahid Mujtaba, Michael Mattsson, 2008)

3.3.Systematic Review Process

In software engineering the existing literature about guideline principals of SLR is rare. No authenticate standards for conducting systematic literature reviews exist (B. A. Kitchenham, et al., 2004). Few SLR guidelines are found which depends upon the practical evidence. To resolve this problem B.A. Kitchenham and J. Biolchini has designed the guidelines for conducting systematic reviews in software engineering. These both kinds of guidelines of SLR are followed by the researchers, but we will use the guidelines presented by Kitchenham and Charters (Barbara Kitchenham & Charters, 2007). According to Barbara Kitchenham guideline, the process of systematic literature review contains three main stages which are then further divided into small steps. The process of systematic literature review has stages and activities which are appropriate for our SLR. The stages are listed below: (Barbara Kitchenham & Charters, 2007)

➢ Stage 1: Planning the SLR

The activities of planning the SLR are:

- Identifying the need for a review
- Commissioning the review
- Specifying the research question
- Development of review protocol
- Evaluation of review protocol

➢ Stage 2: Conducting the SLR

The activities of conducting the SLR are:

- Identification of research
- Selection of primary studies
- Selected Study quality assessment
- Data extraction and monitoring
- Data synthesis

➢ Stage 3: Reporting the SLR

The activities of reporting the SLR are:

- Specifying dissemination mechanisms
- Formatting the main report
- Evaluating the report

All these activities are compulsory except the three stages i.e. evaluating the review protocol, commissioning the review and evaluating the report. These three activities are optional and they are completely dependent upon the review authority. Actually the activities of SLR should be followed sequentially but in reality these several activities are iteratively conducted due to refining the activates during performing SLR (Barbara Kitchenham & Charters, 2007).

3.4. Planning the SLR

Planning the review is first stage of SLR. In planning stage SLR was planned according to conducted pilot study. The included activities of planning stage of SLR are explained in the following sections.

3.4.1. Identify Need for SLR

The need for SLR is important activity of planning stage. This activity describes whether the SLR should be performed or not? It is required to assure the existence of sufficient evidence in literature. We should also identify whether any systematic review exists on Issues' analysis of OSSD or not (Barbara Kitchenham & Charters, 2007). To resolve this problem, systematic mapping technique is used to find out the existing literature.

To answer these questions, make sure that sufficient primary studies exist in the field of open source software development. It is also very important to determine that such kind of SLR

still does not exist. Make sure that there is no existence of any study which resolves the issues of OSSD relevant to our scope. After clarifying these problems it should require mitigation of OSS development issues as soon as possible. Lot of literature has directed to conduct an SLR about issues analysis of OSSD.

3.4.2. Commissioning the Review

When an organization required to conduct a SLR, but did not have time to perform it. In this situation, third party involved to perform SLR. A person should perform SLR for an organization on commissioning basis. When a systematic review commissioned, organization required to develop a commissioning document. According to Barbra Kitchenham, this phase is not required, when SLR has performed by individual for their own need as in case of PhD thesis (Barbara Kitchenham & Charters, 2007). Due to this, commissioning document has not prepared.

3.4.3. Specify the Research Question

In this phase, review question should be specified to progress the review. According to Barbra Kitchenham proposed guideline, the PICOC model is adopted to specify the research question. This will make sure that "execution process" of underlying SLR should be performed.

3.4.4. Development of Review Protocol

Review protocol pronounces the procedure which has followed to execute a systematic review. It should be helpful to avoid the biasness. Without a protocol the execution of SLR should be author oriented (Barbara Kitchenham & Charters, 2007) and this can eliminate the benefit of SLR. Review questions, search strategy, sources of search and search terms are described in review protocol and followed at execution time. A quality assessment criterion is included, applied to the candidate studies. Data extraction forms and data synthesis form are described. These forms accumulate the information of a specific phenomenon. This complete information helps the other researchers to validate the findings of the study (Khan, 2006).

This section will present the contents of the real review protocol (Appendix A) which is developed in the planning phase of systematic review. The contents of our review protocol are described as:

3.4.4.1. Background

The background of review protocol delivers the basis of systematic review. The historical insight of OSSD is described to increase the understandability about the field. From 1950 to 2010 the continuous changes that have happened in OSSD also discussed. The need of conducting this study is also explained. The detail can view in (Appendix A).

3.4.4.2. Review Question

Main focus of a systematic review is review question. This part describes the direction of the systematic review. The review questions are formulated with the help of guidelines as approved by Kitchenham and Charters (Barbara Kitchenham & Charters, 2007). The review questions are structured on the basis of PICOC i.e. Population, Intervention, Comparison, Outcome and Context. The detail can view in (Appendix A).

3.4.4.3. Identification of Research

The designed search strategy is based on the structure of review question (PICOC). We also perform a literature survey to see the terminologies as described in keywords. The search terms are derived from review question to search maximum results from relevant electronic databases. Boolean operator like 'AND' or 'OR' are used for joining multiple search terms. The review emphasized on the issues' analysis of open source software development with reported strategies. The detail search strategy can view in (Appendix A). The search terms are designed using guidelines of Kitchenham and Charters (Barbara Kitchenham & Charters, 2007). Search sources are selected on the basis of quality, relevance and reliability of the evidence. Therefore only relevant databases are selected so that the review should be based on the available and accessible evidences to ensure the reliability of review results.

3.4.4.4. Study Selection Criteria and Procedures

The Study selection criteria are designed to select primary studies for preforming the systematic literature review. Our study selection criterion is designed on the basis of review question's structure i.e. PICOC. Study inclusion and exclusion criteria are designed to include only those studies that can provide the empirical evidences about issues and strategies of open source software development.

Inclusion Criteria:

Research papers will be included on the basis of reading their titles and abstracts.

- Research papers relevant to open source software development issues will be included
- Primary studies written in English language will be considered
- Only case / field studies, experiments and experience/industrial reports will be included for gathering evidence from literature
- Research papers which are based on the expert opinion will be included
- Research papers related to the topic, which do not provide evidence, will be included as weak evidence

Exclusion Criteria:

The following type of papers will be excluded.

- Textbooks and web pages will be excluded
- Primary studies of simple open source software will be excluded
- If a paper is published in several conferences or journals then the most complete version, on the basis of studies discussed in the article, will be included

3.4.4.5. Study Quality Assessment Checklist and Procedure

The quality assessment criteria (QAC) are designed for the primary studies which can ensure the quality. In this QAC the quality of selected studies is assessed by designed checklist (Appendix A). A checklist is formulated and divided in two steps. In the first step, the general questions are included. In the second step after getting their proper answers the specific questions are asked otherwise the study is rejected. The purpose of checklist is to identify the relevant empirical evidence.

3.4.4.6. Data Extraction Strategy

The data, of remaining primary studies after applying the inclusion and exclusion criteria, is extracted in the designed data extraction form (Appendix A). All the selected primary studies are critically reviewed to collect the relevant evidences. The data extraction form is used to

collect the information from the selected studies after decision of inclusion and exclusion. Then this will become the input for data synthesis and analysis.

3.4.4.7. Data Synthesis

When the data of primary studies is extracted then it is integrated in data synthesis form. Qualitative data synthesis techniques are used to accumulate the information due to heterogeneous nature of the primary studies.

One of the data synthesis forms list the issues of open source software development. Second form lists the reported strategies against the issues in open source software development. Third form collects the information about which primary study is covered for each issue of open source software development. Fourth form collects the number of conducted studies against each issue. The fifth form collects the issues and their reported strategies with the 'study ids' to cover them. This information delivers a solid basis for analyzing the given data and drawing conclusions.

3.4.4.8. Dissemination Strategy

The final report of SLR is written in the thesis format which is a partial requirement of International Islamic University, Islamabad. This report is viewed by both internal SIG committee as well as external review committee.

3.4.4.9. Project Schedule

The project schedule describes the timeline of the research project with a date of completion of activities. It is designed before execution of SLR. The schedule of activities can be seen in (Appendix A).

3.4.5. Evaluation of Review Protocol

The review protocol is developed following the guidelines which are designed to perform the SLR (Barbara Kitchenham & Charters, 2007). The review protocol is first analyzed by the thesis supervisor and co- supervisor. This review protocol is verified by pilot study which assures its execution. The pilot study shows that the protocol is implementable and provides good results. The protocol is reviewed by Dr. Mahmood Niazi and Dr. Barbra Kitchenham. According to their feedback the protocol is modified.

3.4.5.1. Threats to Validity

Due to single author involvement in development of review protocol, the chance of bias can be happened. Review protocol is verified by experts but even then it has validity threats. If this review protocol should be published then its validity threats can overcome. We will try to publish this protocol.

CHAPTER NO: 4

CONDUCTING THE REVIEW

4. Conducting the SLR

In this chapter the execution of SLR is described in detail. It is major stage which takes more time and effort in performing the review. In this stage all the activities are completed as planned. When a review protocol is accepted and verified then execution of SLR starts.

4.1.Identification of Research

The focus of this stage is to find out the relevant empirical studies in literature to execute the SLR. This is the most difficult and problematic stage which requires more care. Therefore a query is developed which can find out the whole relevant studies existing in the electronic databases. These queries are executed and verified by the thesis supervisor and co-supervisor. To find out the studies we require the resources (Khan, 2006). Only those electronic databases are used which are available in International Islamic University Islamabad domain.

4.1.1. Generating a Search Strategy

The search strategy is designed to find out the maximum existing relevant literature. The search strategy should be unbiased. Search strategy is developed according to the guidelines of SLR (Barbara Kitchenham & Charters, 2007). When the search strategies are developed the search terms and search sources are included to gather the results.

4.1.2. Source Selection

Search sources are selected on the basis of accessibility of database and availability of relevant studies. The best electronic databases are included to search the relevant empirical studies that can be able to become part of SLR. Google scholar and cite-seer also included but later on, we also include the other well-known databases as: IEEE, Springerlink, ACM, ISI WEB of KNOWLEDGE and Science Direct. These all databases are accessible and recommended by thesis supervisor. The references of selected studies are reviewed and some new studies are also included after consulting with the supervisor. The author also looked for conferences and journals which are not included in the selected databases but no further evidences are found. Some studies are not accessible therefore to getting them we contacted with primary author of studies.

4.1.3. Search Terms

The search strategy is designed on the basis of review question. The search terms (Appendix A) are derived from the main words used in the research question as like i.e. open source software development. This stage is iteratively performed and queries are modified many times. After modification, the designed queries are applied on all selected electronic databases. In this with way more appropriate results found then these results should be included.

Table 1: Search Strings for Each Electronic Database

Database	Search String
IEEE 1.1	("Abstract" :"open source software development" OR "open source software" OR "Free/Libre" OR "free software development" OR "FLOSS") AND ("Abstract" :"limit*" OR "cons" OR "issue*" OR "challenge*" OR "risk*" OR "problem*")
IEEE 1.2	("Abstract" :"open source software development" OR "open source software" OR "Free/Libre" OR "free software development" OR "FLOSS") AND ("Abstract" :"lack*" OR "drawback*" OR "disadvantage*" OR "flaw*" OR "shortcoming*")
ACM 1.1	(Abstract :"open source software development" OR Abstract :"free software development" OR Abstract :"FOSS" OR Abstract :"collaboration software development" OR Abstract :"Free/Libre") AND (Abstract :"limit*" OR Abstract :"cons*" OR Abstract :"issue*" OR Abstract :"challenge*" OR Abstract :"risk*" OR Abstract :"problem*")
ACM 1.2	(Abstract :"open source software development" OR Abstract :"free software development" OR Abstract :"FOSS" OR Abstract :"collaboration software development" OR Abstract :"Free/Libre") AND (Abstract :"lack*" OR Abstract :"drawback*" OR Abstract :"disadvantage*" OR Abstract :"flaw*" OR Abstract :"shortcoming*")
Science Direct	Abstract("open source software development" OR "Open source software" OR "OSS" OR "Free/Libre" OR "free software development" OR "FLOSS") AND Abstract("issue*" OR "limit*" OR "challenge*" OR "drawback*" OR "risk*" OR "problem*" OR "disadvantage*" OR "cons*" OR "lack*" OR "flaw*" OR "shortcoming*")
Springer link 1.1	("open source software development") and ("issues" or "limitations" or "challenges" or "risks")
Springer link 1.2	("open source software development") and ("drawbacks" or "problems" or "disadvantages" or "cons")
Springer link 1.3	("open source software development") and ("lack" or "flaw" or "shortcoming")
Springer link 1.4	("Free/Libre") and ("lack" or "flaw" or "shortcoming")
Springer link 1.5	("Free/Libre") and ("issues" or "limitations" or "challenges" or "risks")
Springer link 1.6	("Free/Libre") and ("drawbacks" or "problems" or "disadvantages" or "cons")

Springer link 1.7	("free software development") and ("issues" or "limitations" or "challenges" or "risks")
Springer link 1.8	("free software development") and ("drawbacks" or "problems" or "disadvantages" or "cons")
Springer link 1.9	("Free software development") and ("lack" or "flaw" or "shortcoming")
ISI Web of Knowledge	Title=("open source software development" OR "free software development" OR "distributed software development") AND Title=("issue*" OR "limitation*" OR "challenge*" OR "risk*" OR "disadvantage*" OR "problem*" OR "con*" OR "lack*" OR "drawback*")

Search queries are developed for each electronic database, as shown in table 1, to get maximum existing empirical studies. The date of execution of search query is also documented so that when the search query is return then new studies could be identified. The search queries are return on each electronic database after few days to check the search string. At that time same results are obtained.

4.1.4. Publication Bias

Due to the problem of accessibility grey literature does not include. Only electronic databases are used to collect data. Selection of search source is also required validation to avoid bias. All the accessible databases in the domain of IIUI are used without any consideration. This problem can be resolved by reviewing external authority (faulty member of Keele University in UK).

4.1.5. Bibliography Management and Document Retrieval

To manage the references in sophisticated manner, Endnote X 2.0(bibliographic tool) has used. Mismanagement of references due to manual arrangement can be error pron. Endnote make it easy to manage the references in sophisticated manner. It also reduces the effort and time. The obtained results of electronic database are inserted in Endnote and it is created notes of each article automatically. It is also helpful in finding duplicate results, which is impossible in manual arrangement (Khan, 2006).

Three categories of Endnote have maintained to secure the record. These categories named as: "Included papers", "Excluded papers" and "Duplicate studies". Few studies are confusing in nature. To categorize these individual studies it is required to consult them with thesis supervisor.

4.1.6. Documenting the Search

The search strings are designed on the basis of abstract and full text of studies. The results obtained on the basis of full text are relevant then from abstract. Therefore full text results are included in the review, after consulting with supervisor. The obtained results from databases as in table 3 are completed on 15-Sep-2010. In review protocol, the search process for each electronic database is already documented (Appendix A).

Table 2: Search Documentation for Each Database

Database	Documentation
Digital Library	Name of Database: IEEEXPLORE Search string for a database: • Only English papers • Search criteria are abstract only • A search query is executed Date of search 15-09-2010
Digital Library	Name of Database: ACM Search string for a database: • Only English papers • Search criteria are abstract only • A search query is executed Date of search 15-09-2010
Digital Library	Name of Database: Science direct Search string for a database: • Only English papers • Search criteria are abstract only • A search query is executed Date of search 15-09-2010
Digital Library	Name of Database: Springer link Search string for a database: • Only English papers • Search criteria are abstract only • A search query is executed Date of search 15-09-2010
Digital Library	Name of Database: ISI Web of Knowledge Search string for a database: • Only English papers • Search criteria are abstract only • A search query is executed Date of search 15-09-2010
Advance Search	Name of Database: Googlescholar.com Search string for a database: • Only English papers • Search criteria are abstract only • Use keywords

	Date of search 15-09-2010

Table 3: Integrated Search Results

Basis of selection	No. of Papers found in each database					
	IEEE	ACM	ISI WEB OF KNOWLEDGE	SPRINGER VERLAG	SCIENCE DIRECT	Total
Abstract	89	53	03	95	48	288
Full text	49	40	02	40	06	137

4.2.Study Selection

In this step every individual study collected in pervious step should be analyzed according to the inclusion and exclusion criteria. After applying criteria, selected studies are included in SLR.

4.2.1. Study Selection Criteria

In this step relevant evidence has filtered from the existing literature by applying the designed study selection criteria to the review question. Study selection criteria consist of study inclusion and exclusion criteria. These both criteria have also included in review protocol to avoid researcher's bias. To validate inclusion and exclusion criteria, it has piloted on few studies and also analyzed by supervisor, co-supervisor and external reviewer. After mutual agreement this criteria adopted. Only those studies included which have passed the criteria. No weak evidence has used as primary study. The study is excluded if it does not provide relevant evidence.

4.2.2. Reliability of Inclusion Decisions

Inclusion criteria has designed and applied by only researcher therefore it requires to avoid the bias. To solve this problem, thesis supervisor counter check inclusion criteria after and before applying on few studies. The rejected studies have first discussed with thesis supervisor to take decision about inclusion or exclusion.

4.2.3. Study Selection Process

After applying the study select process, duplicate studies have identified. Endnote (bibliographic tool) is used to find duplicate studies. According to this tool, there are 113/614

duplicate studies exist. The changing status of obtained studies can viewed in table 4. Endnote also helps us to maintain the list of included and excluded Primary studies.

Table 4: Study Selection Results

Study inclusion/exclusion criterion	Query results	Studies left
Total studies retrieved from electronic databases	614	614
No. of duplicate studies	113	501
Excluded studies after reading titles and abstracts	213	288
Simple OSSD studies excluded after reading full test	151	137
No. of studies excluded after recommendation of thesis supervisor	66	71

In the next round, abstracts and titles of 501 remaining research papers have analyzed to exclude the irrelevant studies. After reading abstract and title we have found 213/501 irrelevant studies. Now in next round, studies of simple open source software development have excluded after reading abstracts and conclusion. In this round, excluded studies are 151/288. Now 137 studies left which have studied in detail and also discussed with the thesis supervisor. As a consequence, 66 more studies are disqualified as they does not supporting the review questions. Now 71 selected studies have remained for the next step of execution.

4.2.4. Study Quality Assessment

The study quality assessment is a major activity of the SLR, and it is designed to filter out primary studies. After applying the QAC the evidence base primary studies are found, which are included in SLR. As compare to study selection criteria, it's more detailed in nature. QAC has extracted the studies with quality score.

4.2.5. Hierarchy of Evidence

A systematic review has set the threshold for experiments, case studies, field studies and industrial experience reports. Experiments and case studies are more valuable than others. Then field studies, expert opinion and experience reports have also selected. But in OSSD there is no experiment exist which is relevant to review question. Experiments do not conduct as geographical nature of OSSD cannot be supporting for it. Lot of case studies, interviews and surveys has found to conduct this systematic literature review.

4.2.6. Developing Quality Instrument

Quality instrument has developed to measure the quality. Quality of an individual study can be determined with the help of quality assessment criteria (QAC). QAC has also included in review protocol. The problem of bias will also considered here. To solve this problem, thesis supervisor and co-supervisor analyze the QAC. The checklist designed by Barbra Kitchenham for SLR is followed. The checklist has completed after recommendation of supervisor and co-supervisor.

4.2.7. Using Quality Instrument

The designed QAC contains seven different questions from which two questions are general and other 5 are detailed questions as concentrating on the each study design.

There are three possible answers like Yes/No/Partial for each question with value 1, 0, 0.5 accordingly. For each study minimum quality threshold was 1.0. The cause for adopting this threshold is that the answer of the three questions can be 0.5. Therefore, the appropriate threshold for the studies has 1.0. The minimum quality threshold determined:

- Did the study provide empirical evidence or not?
- Does the study clearly state the issues of open source software development?

When the obtained primary study has passed the initial criterion then it has tested for the next detailed criteria. The selected primary studies are rejected as they do not pass the initial criteria. The primary studies are finally included and extracted after consulting with the thesis supervisor.

4.3.Data Extraction

The purpose of data extraction stage is to integrate and summarize the required information in sophisticated mannered. Endnote helps to managing the research publications. Lot of information has retrieved form Endnote instead of manually collected one by one. A list of included primary studies has also maintained. The information gained from the selected primary studies is included in the systematic review.

4.3.1. Data Extraction Form

Data extraction forms have developed to extract all required information. The obtained information has essential part for the review question. To avoid researcher's bias, data extraction forms have designed in the review protocol (Appendix A). The DE form has consists of two types of questions; generic and specific questions. . The generic questions are like extraction date, title of study, name of database, year of publication etc. The specific questions are relevant to the area of interest i.e. issues of open source software development.

The specific questions concentrated on the detailed description of the results of the studies and identification of issues and their reported strategies of open source software development. When review protocol has developed then reliability of data extract form has checked by performing the pilot study. When pilot study validates the data extraction form then few changes have finalized with recommendation of thesis supervisor. The data extraction form has also tested by an external reviewer.

4.3.2. Data Extraction Procedure

When SLR is conducted then data should be extracted by more than one researcher. This may decrease the chance of bias. During the underling SLR this is not happened. Single author has extracted data from primary studies therefore to avoid the bias; it has reviewed by thesis supervisor. Data has cumulated from primary studies then thesis supervisor counter check this information by random sampling. With the involvement of thesis supervisor the results are validated. Few uncertainties are found among the outcomes of primary studies, which are discussed and fixed after consulting with the thesis supervisor.

4.3.3. Multiple Publications of Same Studies

Lots of duplicate studies have found which should be removed during the study select stage of SLR. Same studies have found with different publishers. This problem has fixed after consulting with thesis supervisor. All these type of studies has placed in same DE form. But the studies that are duplicated and published in some other conference; should be rejected. A list of duplicate primary studies has also maintained for consulting with supervisor.

4.4.Data Synthesis

Purpose of data synthesis is to summarizing the obtained information that is collected from primary studies. Data synthesis can be done by two methods, qualitative synthesis and quantitative synthesis. In this systemic review qualitative synthesis is used instead of than other. The obtained results are heterogeneous in nature; therefore it cannot be possible to use both methods. Tabular representation is used to organize the obtained results. This representation is more suitable than any other. The results obtained from the primary studies are reported below:

What are the reported issues in open source software development?

Answer: The list of issues of open source software development can view in table 5. These issues are arranged with the help of 'issue ids' as like I-1, I-2 etc. Each study and reported strategy also arranged by 'study ids' or 'strategy ids' as like SD-1, ST-1 etc. Finally, 15 primary studies were selected to gather the empirical evidences. Only 11 reported issues of open source software development are identified from these studies.

Table 5: List of Issues of Open Source Software Development

Issue Categorization	Issue Ids	Issues of Open Source Software Project	Study Ids	Studies Cover the Issue
Requirement issues	I-1	Improper requirement of OSS project	SD-10	1
Communication issues	I-2	Scope of team and status of project activities are unknown	SD-1,SD-9	2
	I-3	OSS projects are not develop with active participation	SD-1	1
	I-4	Modified changes are not suitable into the OSS project	SD-1	1
Code Reusing issues	I-5	By reusing of source code in OSS projects, dependability established between projects	SD-6	1
	I-6	Due to dependencies in OSS project, it's difficult to install and use	SD-6	1
	I-7	Reusing of source code generates extra work	SD-6	1
	I-8	Finding and implementing of reusable resources take more time than from start	SD-6	1
	I-9	Quality issues are produced due to source code reusing	SD-6	1
	I-10	Adapting and integrating of source code take more time than building from start	SD-6	1
	I-11	Understanding of source code take more time than building from start	SD-6	1
	I-12	Performance of OSS project has been affected by source code reusing	SD-6	1
	I-13	Security issues are produced due to source code reusing	SD-6, SD-9	2
Integration issues of OSS components	I-14	Perform duplicate tasks	SD-8, SD-9	2
	I-15	Spending time on changes are wasted	SD-8	1
	I-16	Source code Integration issue	SD-8	1
Testing issue	I-17	Testing has always ignored in OSS project	SD-3,SD-13,SD-10	3
Maintenance issue	I-18	Problematic maintenance of OSS projects	SD-2,SD-4,SD-13, SD-5	4
Documentation issue	I-19	Non-existence of documentation	SD-1 SD-9, SD-10	3
Quality problems	I-20	Source code developed by ex-developers is unsupported	SD-9	1
	I-21	Configuration management	SD-9	1

	I-22	Users do not know how to report bugs	SD-9	1
	I-23	Attracting contributors towards the project is difficult	SD-9	1
Architectural issue	I-24	Architectural issue	SD-11	1
Deployment issue	I-25	Poor deployment support	SD-10	1
Releasing issues	I-26	Release mismanagement issue	SD-15	1
Miscellaneous	I-27	Frustration of end user with rapidly releases	SD-5	1
	I-28	Lack of Platform independence	SD-5	1
	I-29	Poor support for compile time and run time configurations	SD-5	1
	I-30	Ensuring coherency of system wide properties	SD-5	1
	I-31	Developed interface is not user friendly	SD-4	1
	I-32	No risk assessment and achievable project goals set	SD-12	1
	I-33	No formal process followed in OSSD	SD-14	1
Evolution issue	I-34	Evolution and development of OSS component is more difficult than others	SD-16	1

What strategies are reported to deal with these issues?

Answer: The reported strategies that can resolve the issues are listed in table 6. These are approved strategies from literature which can mitigate the reported issues i.e. RQ 1. If an OSS Project adopts these strategies, from start to end then it can be expected that the failure rate of OSS projects may decreased. OSS projects have a great failure rate, if its failure rate decreases then our purpose of research should be complete. The reported strategies have arranged by 'strategy ids' i.e. ST-1, ST-2, etc.

Table 6: Reported Strategies to Resolve the Reported Issues

Strategy ID	Reported Strategies of Open Source Software Development	Study ID	No of Studies Cover the Strategy
ST-1	Good communications support our forthcoming in open source software projects.	SD1	1
ST-2	Defect management process	SD2	1
ST-3	Reiterative testing activity during the whole development process of OSS project	SD3	1
ST-4	End-user involvement in OSSD	SD-4	1
ST-5	Incrementally improve the quality and performance of open source systems Minimize human effort by applying automation judiciously Minimize unnecessary end user overhead without compromising	SD-5	1
ST-6	Management need to appoint the champions as sponsors', reuse-librarians, or reuse-coordinators	SD-7	1
ST-7	Announcement and discussion of one's choice of work	SD-8	1
ST-8	Three practices are suggested : • Infrastructure • Processes • Documentation	SD-9	1
ST-9	Successful business plan based on open source technology must address the issues.	SD-10	1
ST-10	Architecture refactoring activity in order to repair software architectures	SD-11	1
ST-11	Consistently follow established guidelines as planned	SD-15	1

Table 7: Reported Strategies for Each Issue of Open Source Software Development

Issue ID	Strategy ID	Issues of OSS Development	Reported Strategies of OSS Development	Study ID Covers the Strategy
I-1	ST-9	Improper requirement of OSS project	Successful business plan based on open source technology must address the issues.	1
I-2	ST-1	Scope of team and status of project activities are unknown	Good communications support our forthcoming in open source software projects.	1
I-3		OSS projects are not develop with active participation		
I-4		Modified changes are not suitable into the OSS project		
I-5	ST-6	By reusing of source code in OSS projects, dependability established between projects	Management need to appoint the champions as sponsors', reuse-librarians, or reuse-coordinators	1
I-6		Due to dependencies in OSS project, it's difficult to install and use		
I-7		Reusing of source code generates extra work		
I-8		Finding and implementing of reusable resources take more time than from start		
I-9		Quality issues are produced due to source code reusing		
I-10		Adapting and integrating of source code take more time than building from start		
I-11		Understanding of source code take more time than building from start		
I-12		Performance of OSS project has been affected by source code reusing		
I-13	ST-6, ST-8	Security issues are produced due to source code reusing	• Management need to appoint the champions as sponsors', reuse-librarians, or reuse-coordinators • Three practices are suggested as Infrastructure, Processes, documentation	2
I-14	ST-7	Perform duplicate tasks	Announcement and discussion of one's choice of	1

I-15		Spending time on changes are wasted	work	
I-16		Source code Integration issue		
I-17	ST-3	Testing has always ignored in OSS project	Reiterative testing activity during the whole development process	1
I-18	ST-2	Problematic maintenance of OSS projects	Defect management process	1
I-19	ST-8, ST-9	Non-existence of documentation	• Three practices are suggested as Infrastructure, Processes, documentation • Successful business plan based on open source technology must address the issues	2
I-20	Nil	Source code developed by ex-developers is unsupported	Not reported	0
I-21	ST-8	Configuration management	Three practices are suggested as Infrastructure, Processes, documentation	1
I-22		Users do not know how to report bugs		
I-23		Attracting contributors towards the project is difficult		
I-24	ST-10	Architectural issue	Propose the architecture refactoring activity in order to repair software architectures	1
I-25	ST-9	Poor deployment support	Successful business plan based on open source technology must address the issues.	1
I-26	ST-11	Release mismanagement issue	Consistently follow established guidelines as planned	1
I-27	ST-5	Frustration of end user with rapidly releases	• Incrementally improve the quality and performance of open source systems • Minimize human effort by applying automation judiciously • Minimize unnecessary end user overhead without compromising	1
I-28		Lack of Platform independence		
I-29		Poor support for compile time and run time configurations		
I-30		Ensuring coherency of system wide properties		
I-31	ST-4	Developed interface is not user friendly	End-user involvement in OSSD	1
I-32	Nil	No risk assessment and achievable project goals set	Not reported	0
I-33	Nil	No formal process followed in OSSD	Not reported	0
I-34	Nil	Evolution and development of OSS component is more difficult than others	Not reported	0

Table 8: Primary Studies Used in SLR

Study Ids	Title of Selected Primary Studies
SD-1	A case study of developing an IDE for Embedded Software Using open source (Dominik Ertl, Harald Krapfenbauer, 2009)
SD-2	Exploring the maintenance process through the defect management in the open source projects four case studies(Heli Lintula, Timo Koponen, Virpi Hotti, 2006)
SD-3	Towards certifying the testing process of open source software New challenges or old methodologies(Sandro Morasca, Davide Taibi, Davide Tosi, 2009)
SD-4	The (LA)TEX Project -A case study of open source software(Alexandre Gaudeul, 2003)
SD-5	Leveraging open source communities to improve the quality & performance of open source software (Douglas C. Schmidt , Adam Porter , 2001)
SD-6	Code Reuse in Open Source Software Development Quantitative Evidence, Drivers, and Impediments (Manuel Sojer, Joachim Henkel, 2010)
SD-7	Code Reuse in Open Source Software (Stefan Haefliger, Georg von Krogh, Sebastian Spaeth, 2008)(Jesper Holck, Niels Jørgensen, 2003)
SD-8	Continuous Integration and Quality Assurance a case study of two open source projects (Jesper Holck and Niels Jørgensen, 2003)
SD-9	Quality practices and problems in free software projects (Martin Michlmayr, Francis Hunt, David Probert, 2005)
SD-10	A case study Open source community and the commercial enterprise (Kevin Gary, Harry Koehnemann, John Blakley, Cheryl Goar, Holly Mann, Al Kagan, 2009)
SD-11	Software architecture relevance in open source software evolution A case study (Elisa Yumi Nakagawa, 2008)
SD-12	Code quality analysis in open source software development (Ioannis Stamelos, Angelis, Lefteris; Oikonomou, Apostolos; L. Bleris, Georgios ;, 2002)
SD-13	Evolution in Open Source Software (Michael W. Godfrey and Qiang Tu, 2000)
SD-14	Open source software development a case study of FreeBSD (Trung Dinh-Trong, James M. Bieman)
SD-15	Subversion 1.5 A Case Study in Open Source Release Mismanagement (Hyrum K. Wright and Dewayne E. Perry, 2009)
SD-16	The evolution of open source software using Eclipse metrics (Ajlan Al-Ajlan, 2009)

4.5. Findings

In this section the reported issues and reported strategies of open source software development are evaluated. The selected primary studies can view in graphical representation as per year. The research methods used in primary studies can also view. Only those research methods should be chosen that are included in EBSE.

4.5.1. Issues of Open Source Software Development

Improper requirements of OSS projects tend toward the serious issue. In OSSD requirements are defined by the project developers. No requirements related activities are performed as compare to CSS development. An improper requirement may consume more cost when changed in later stage. *Scope of team and statues of project activities are unknown* therefore development of OSS system has become difficult, and benefits of low cost in OSSD should not be achieved. As OSS *projects are not developed with active participation*, due to developer's part time participation. Therefore developers cannot perform the required task in limited time. In OSSD, developer's behavior and attitude is not serious as in CSSD. *Modified changes are not suitable* in OSS project, due to OSS systems hybrid nature. It is important to convince the OSS community to include necessary changes and to exclude extra features. It can eliminate the fear of forking in OSS project. Forking happens in OSS projects due to change of community focus, and by this way single project has divided into two branches. Some issues are generated when the source code is reused. When a component has reused then by reusing of source code in OSS projects, *dependability established between projects* to add the features and remove the bugs.

Due to dependencies in OSS project *it's hard to installs and uses* the project by end user. Some technical dependencies are created in project which generates this problem. *Reusing of source code generates some extra work*, which is wastage of time and effort. *Finding and implementing of reusable resources* take more time than from start. When a component is reused then it should be required to check all aspects of component, which takes lot of time. *Quality and security issues* are produced due to source code reusing, in OSS projects.

When a component has reused then its *adaption, integration and understanding* take more time than building from start. *Performance of OSS project* has affected by source code reusing. When OSS project components are integrated then some issues are generated as,

perform duplicate tasks, spending time on changes are wasted and source code integration difficulties issue. Testing and documentation has always ignored therefore *maintenance* becomes difficult in OSS project. With the initiation of OSS few problems are also initiated. As *attracting contributors towards the project and users do not know about bug reporting process.* OSS system and free source software's have *configuration management* issue due to high level of customization. It is quit impossible for new developer to maintain the source code when a developer leave the project. Such a source code it called unsupported code. In OSS development no design and architecture found which made easy maintenance and testing process.

Release mismanagement and *many rapid releases* frustrate the participants and end-users which tend to leave the project. During development of OSS projects their *deployment* becomes difficult. Change in adopted operating system and compiler will not convenient in OSS project development, therefore *lack of platform independence issue* has also created. When components have integrated then it's difficult to make sure a stable semantic and common standard APIs. In OSS development it's difficult to develop user *friendly interface* due to absence of end-user involvement. No risk assessment and achievable project goals set and no formal process followed in OSSD. The reported evaluation of OSS components is more difficult than others. These all are open issues and there mitigation strategies are still unreported for them.

The above reported issues of open source software development have found from the empirical studies to reduce the failure rate of OSSD. As success of OSSD tends the industry to adopt this methodology, and it is possible that industry frustrates and leaves this methodology due to its increasing failure rate.

4.5.2. Strategies of Open Source Software Development

Virtuous communications support in the whole OSSD process can overcome the communication issues. With proper communication activities status and team size can be accountable. The changes adopted or leading towards adoption should be communicated to every participant. Defect management process has proposed for the maintenance of OSS projects. In this process bugs are reported by user to developer. In OSSD testing activities are ignored. Testing issue can mitigated by reiterative testing of complete OSS development process.

In OSSD, a user friendly interface can be generated by the involvement of end user. OSS development has long term development time period, therefore change in technology happened. Due to which users are frustrated and platform independency like issues have created. This can be mitigated by proposed strategies. Incrementally improve the quality and reduce human effort and end user involvement. If a participant's choice of project task has announced the problem of duplicate work can be removing. The strategy of hiring champions as sponsors can be mitigating the code reusing issue. Some practices are suggested to mitigate the issue of *attracting contributors towards the project; users do not know about bug reporting process and configuration management. These practices are divided into three categories'* as infrastructure, processes and documentation. Bud tracking system CVS, automatic builds and mailing list are included in infrastructure. Process like join, release, branches, peer review, testing and quality assurance are suggested to include in project. Documentation of OSS project is almost ignored; it is suggested to build two type of documentation as code style and code commit.

If a project has an infrastructure as: process, documentation as suggested then quality issues can be mitigated. A successful business plan for open source software projects can resolve the issues of requirement gathering and testing of OSS project. When these issues are resolved then project deployment should also become easy. Design and architectural issues may eliminated by repairing the architectures by proposed refactoring action. Release process of OSS projects should be managed by consistently following the plans.

4.5.3. Selection of Research Methodology

The selected primary studies have three type of research methodologies used in systematic literature review. These methodologies are included the survey, interview, and case studies. Due to geographical distribution of OSSD, it is quite impossible to conduct an experiment. Therefore till no experiments exists. Expert opinions relevant with our research phenomena also not exist. Therefore we have found these methodologies as shown in graph.

Figure 4

4.5.4. No Selected Evidences Per Year

In this study we have 17 evidences included. Maximum no. of research publications are 6/17 exist in 2009 and minimum no. of research publications are 1/17 which exist in many years i.e. 2000, 2001, 2002, 2004, 2005, 2006, and 2010. In the year 2003 and 2008 research publications are 2/17 exist. This graphical representation shows us that no. of empirical studies published in the area of open source software development is very low. This will indicate us about the need of more empirical studies year to explore the area that shows the issues of open source software development.

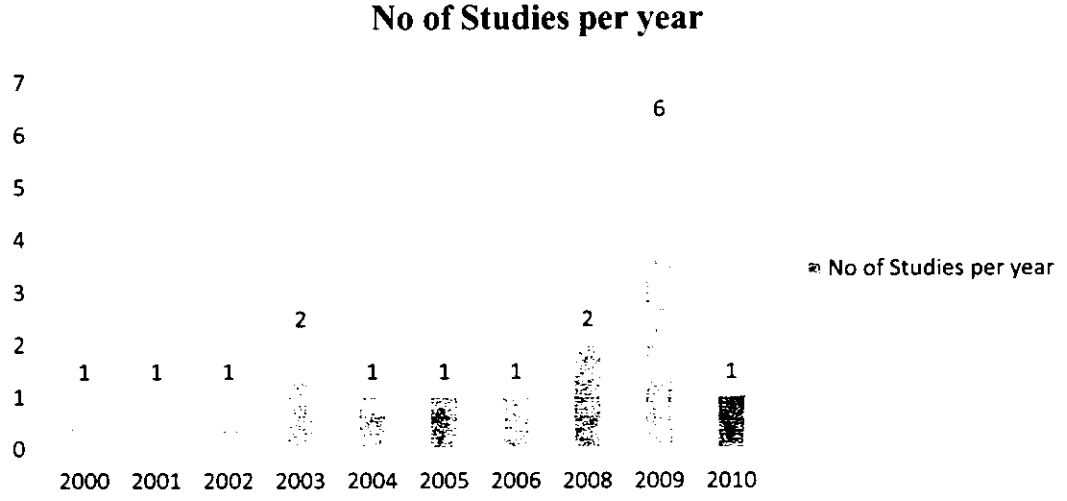


Figure 5

4.5.5. Validation of Systematic Literature Review

Outcomes of systematic literature review are unbiased. There are many kinds of validity exists which has used to define the chance of bias in the SLR. We have described two types of validities that are: internal and external validity.

The conducted research has accomplished by a single researcher; therefore chances of personal biasness should be happened. Due to this problem, thesis supervisor has counter check all outcomes of every stage during SLR. The conducted SLR has completed as planned in review protocol. Only schedule mismatch found due to some difficulties.

The conducted systematic literature review has solid external validity. Sufficient empirical studies have found in the literature that supports the result of SLR. But still some problems are exist, which produce bias i.e. date of searching and terms used for search may be imperfect. Due to accessibility issue, grey literature has ignored. For this type of validity, a review protocol has piloted and critically evaluated by supervisor, co supervisor and an external authority.

Table 9: Structure of SLR

Section	Subsections	Scope
Title	Issues Analysis of Open Source Software Development	All the reported issues of OSSD and their reported mitigation strategies had focused
Executive summary	Context	Lack of cumulative empirical knowledge about the issues of open source software development was the major concern of this study. Therefore SLR is most suitable method to conduct this study. All reported issues had identified and mitigated except the open issues that has not mitigation in literature. Then these issues had been categories not the basis of their nature.
	Objectives	
	Methods	
	Results	
	Conclusions	
Background		History of OSS from 1950 to 2010 has been discussed. When name of OSS had approved
Review questions	RQ 1 and RQ 2	In both questions reported issues of OSSD and their reported mitigation strategies had focused
Review Methods	Data sources and search strategy	Few electronic databases has used as described before. The retrieved selected studies should be included after applying inclusion and QAC. Then data had been extracted in designed forms and also synthesized as planned in review protocol
	Study selection	
	Study quality assessment	
	Data extraction	
	Data synthesis	
Included and excluded studies	Included criteria	Only studies which provides evidence in favor of review question had extracted
	Exclusion criteria	Studies which do not provides evidence in favor of review question had been excluded
Results		Evidence based reported issues had identified and also mitigated by reported strategies, but some of identified issues are still not mitigated.
Conclusions		13 categories have developed on the bases of 34 issues. These categories have developed on the nature of identified issues.

CHAPTER NO: 5

CONCLUSION

5. Conclusion

This chapter concludes the systematic findings of systematic literature review. The recommended guidelines and future directs for researchers and practitioners have documented. These guidelines will overcome the issues of open source software development.

5.1. Discussion

In this systematic literature review only evidence-based reported issues and reported strategies of open source software development are investigated. Most convenient electronic databases included: IEEE, Springerlink, ACM, Science direct, ISI Web of Knowledge and googlescholar.com. Validity threats are also effectively addressed.

Source Forge has estimated that in March 2009 approximately out of 158669 registered projects 17% OSS projects are categories as stable projects, and only 1.52% projects are included in mature category. Failure rate of OSS projects tends to mitigate the issues of OSSD. This SLR has identified the reported issues and their mitigation strategies of OSSD.

These issues are raised in the whole development cycle in OSSD. This critical analysis has performed systematically and identifies issues and mitigation strategies from project initiation to project release. 34 identified issues are the major causes of OSS projects failure. These identified issues are categories according to nature of issues. 13^{th} categories of OSSD issues are developed. These categories have cumulate 34 issues as table 5.

The reported issues had categorized as following:

Requirement issues, communication issues, code reusing issues, integration issues of OSS components, testing issue, maintenance issue, documentation issue, quality problems, architectural issue, deployment issue, releasing issues, miscellaneous and evaluation issue.

These reported categories contain whole OSS development issues. Some of the issues have remained uncovered. Some reported issues may be reported after 15-Sep 2010. Those issues are not included in this report of SLR. Some social issues also found as culture issue and social norms, which are out of scope in this report. This report has contained only those issues of OSSD that are reported on the basis of evidence. Issues without evidence, as time zone differences in OSS were rejected in study selection procedure.

To mitigate the reported issues of OSSD lot of reported strategies have found. These strategies can resolve these issues. Good communicational support for upcoming actives during OSSD may resolve the communication issues. Maintenance issues can be resolved by adopting the defect management process. This process has good bugs reporting system which can eliminate the bugs. All the issues have mitigation strategies except few of them. These are open issues whose mitigations strategies are still missing.

5.2. Recommendations

In this section some of important recommendations have been pointed. These recommendations have required attention for researchers and practitioners.

5.2.1. Future Directions

There are several future directions for the OSS research community. These are still not in consideration, and needed to explore. These future directions are as following:

- The reported strategies required reporting frequency of mitigation which tends to inform; at what extent this problem can be solved by this strategy. This will be done by replication case studies. By replicated case studies it's easy to report the frequency of mitigation.
- Few articles have identified the open issues of open source software development. These issues have needed to resolve by evidence.
- Some of the issues of OSSD are still not reported by evidence. Therefore they are not included in our report. By reporting issues and mitigation strategies the rate of OSS project failure will be reduced.
- No formal process has exist that can be follow in OSSD. If a process can be introduced and implemented in the environment of OSSD then it may be possible to remove all issues of OSSP's.

References

[1] Ajlan Al-Ajlan. (2009). The Evolution of Open Source Software using Eclipse Metrics. 2009 International Conference on New Trends in Information and Service Science, (pp. 211-218).

[2] Alexandre Gaudeul. (2003). The (LA)TEX project: A case study of open source software. Proceedings of the 2003 Annual Meeting, (pp. 132-145).

[3] Andreas Bauer, Markus Pizka. (2002). The Contribution of Free Software to Software Evolution. Proceedings of the Sixth International Workshop on Principles of Software Evolution (IWPSE'03), (pp. 1-10).

[4] Audris Mockus, Roy T. Fielding, James Herbsleb. (2000). A Case Study of Open Source Software Development:The Apache Server. ICSE , (pp. 263-272). Limerick, Ireland.

[5] Brian Fitzgerald. (2004). A Critical Look at Open Source. (pp. 92-94). IEEE Computer Society.

[6] Conlon, M. P. (2007). An Examination of Initiation, Organization, Participation, Leadership, and Control of Successful Open Source Software Development Projects. Information Systems Education Journal, 1-13.

[7] Cristina Gacek, Tony Lawrie, Budi Arief. (2004). The many meanings of Open Source. IEEE Computer Society , 34 - 40 .

[8] David Bretthauer. (2001). Open Source Software: A History. UConn Libraries Published Works.

[9] Dominik Ertl, Harald Krapfenbauer. (2009). A Case Study of Developing an IDE for Embedded Software Using Open Source. Fourth International Conference on Software Engineering Advances, (pp. 191-196). Vienna, Austria.

[10] Douglas C. Schmidt , Adam Porter . (2001). Leveraging Open-Source Communities To Improve the Quality Performance of Open-Source Software. First Workshop on Open-Source Software Engineering , (p. 11 pages).

[11] Elisa Yumi Nakagawa. (2008). Software Architecture Relevance in Open Source Software Evolution: A Case Study. Annual IEEE International Computer Software and Applications Conference, (pp. 1234-1239).

[12] Greg Madey, Vincent Freeh, Renee Tynan. (2002). The Open Source Software Developement Phenomeno: Analysis Base on Social Network Theory. Eighth Americas Conference on Information Systems, (pp. 1806-1813).

[13] Heli Lintula, Timo Koponen, Virpi Hotti. (2006). Exploring the Maintenance Process through the Defect Management in the Open Source Projects Four Case

Studies. Proceedings of the International Conference on Software Engineering Advances (ICSEA'06). Finland.

[14] Hyrum K. Wright and Dewayne E. Perry. (2009). Subversion 1.5: A Case Study in Open Release Mismanagement. Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009. FLOSS '09. ICSE Workshop on .

[15] Ioannis Stamelos, Angelis, Lefteris; Oikonomou, Apostolos; L. Bleris, Georgios ;. (2002). Code quality analysis in open source software development. Blackwell Science Ltd Information Systems Journal.

[16] Jesper Holck and Niels Jørgensen. (2003). CONTINUOUS INTEGRATION AND QUALITY ASSURANCE: A CASE STUDY OF TWO OPEN SOURCE PROJECTS. Australasian Journal of Information Systems.

[17] Jesper Holck, Niels Jørgensen. (2003). Continuous Integration and Quality Assurance a case study of two open source projects. Australasian Journal of Information Systems, 40-53.

[18] Kareem Ullah, Shahbaz Ahmed Khan. (2011). A review analysis of open source software development. JATIT, 98-108.

[19] Kevin Gary, Harry Koehnemann, John Blakley, Cheryl Goar, Holly Mann, Al Kagan. (2009). A Case Study: Open Source Community and the Commercial Enterprise. 2009 Sixth International Conference on Information Technology: New Generations, (pp. 940-945).

[20] Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, Yunwen Ye. (2002). Evolution Patterns of Open-Source Software Systems and Communities. International workshop on principles of software Evalution . Boulder: ACM.

[21] Manuel Sojer, Joachim Henkel. (2010). Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments. Journal of the Association for Information Systems, 868-901.

[22] Marina Marinela Gerea. (2006). Selection and Evaluation of Open Source Components. Thesis.

[23] Martin Michlmayr, Francis Hunt, David Probert. (2005). Quality Practices and Problems in Free Software Projects. Proceedings of the First International Conference on Open Source Systems, (pp. 24-28). Genova,.

[24] Michael W. Godfrey and Qiang Tu. (2000). Evolution in Open Source Software:A Case Study. Software Maintenance, 2000. Proceedings. International Conference on , (pp. 131-142).

[25] 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.3A, 303-312.

[26] Sandro Morasca, Davide Taibi, Davide Tosi. (2009). Towards Certifying the Testing Process of Open-Source Software: New Challenges or Old Methodologies? ICSE'09 Workshop, (pp. 25-30). Vancouver, Canada.

[27] Stefan Haefliger, Georg von Krogh, Sebastian Spaeth. (2008). Code Reuse in Open Source Software. MANAGEMENT SCIENCE, 180–193.

[28] Timpo Koponent. (2007). Evaluation of Maintenance Processes in Open Source Software Projects Through DVMS.

[29] Trung Dinh-Trong, James M. Bieman. (n.d.). Open source software development a case study of FreeBSD. Proceedings of the 10th International Symposium on Software Metrics (METRICS'04).

[30] Vijay K. Gurbani, Anita Garvert, James D. Herbsleb. (2006). A Case Study of a Corporate Open Source Development Model. ICSE'06. Shanghai, China.

Appendix

Review Protocol

Document Version Control:

Table 1 : Updates of Review Protocol

Document Status	Version	Date	Changes in old version
Review Protocol	1.0	20-08-2010	None
Review Protocol	1.1	24-08-2010	PICOC model
Review protocol	1.2	29-08-2010	Search string and keywords
Review protocol	1.3	02-09-2010	Data extraction and synthesis form
Review protocol	1.4	09-09-2010	Data analysis and synthesis form
Review protocol	1.5	14-09-2010	Final changes after pilot study

Introduction

According to B.Kitchenham [01] "A systematic review is a defined and methodical way of identifying, assessing, and analyzing published primary studies in order to investigate a specific research question. A systematic review can also discover the structure and patterns of existing research, and so identify gaps that can be filled by future research".

Purpose of evidence base software engineering (EBSE) is to improve the decision making about the software development and maintenance. The existing evidences of research with practical experiences are used for this purpose. "This aim is decidedly ambitious, particularly because the gap between research and practice can be wide. EBSE seeks to close this gap by encouraging a stronger emphasis on methodological rigor while focusing on relevance for practice. This is important because rigor is necessary in any research that purports to be relevant" [02].

Development of Review Protocol

According to B.Kitchenham [03], "A review protocol specifies the methods that will be used to undertake a specific systematic review. A predefined protocol is necessary to reduce the possible researcher bias". The components of a review protocol include all the elements of the review plus some additional planning information [03].

1. Background
2. Research questions
3. Identification of research literature
4. Study selection criteria and procedure
5. Study quality assessment checklist and procedure
6. Data extraction strategy
7. Synthesis of extracted data
8. Dissemination strategy
9. Research plan

Keywords: Open source software development, traditional software development, open source software project.

1. Background

In this background we are describing the meanings and history of OSS. When OSS was initiated? and how it becomes a complete software development methodology. We have also explored the basic differences of TSD and OSSD. This brief background will help us to increase the understandability about OSSD.

The word open source is representing the software development process that depends on the physically dispersed developers, which are connected by internet. Accessibility of source code is the main characteristic of open source software projects. Purpose of open source software systems is to provide the independencies in order to use the system for any purpose [4]. In OSSD, system is developed by an individual or team of software developers. A prototype system is released on the Internet, which can be accessed freely and system's source code can be read, modified and redistributed [5].

When someone talks about open source, then many opinions come in the mind like these are open software, with open collaboration, open process, open release, open deployment and open environment. When open software is released then it allows the further source code distribution, redistribution and licensing. In open collaboration the involvement depends upon the discussion groups, virtual meeting rooms, and shared assets of development. Open process means the external view of development and the product release, which may include the external auction of projects and coordination for resources. Open deployment that focuses on the new products releases. Open environment that directed toward the use of open source products in system development. Free software advocate Richard Stallman and free software foundation describe that OSSD directed towards complete software development methodology [6].

There are some basic differences between TSD and OSSD. A large number of volunteers have participated in OSS systems development. The work is not assigned to the participants in OSS systems. No system level design exists in OSS development. No project planned, scheduled and no list of deliverables exists. These differences propose a tremendous case of physically distributed development. Therefore software developers are doing their assigned work at random locations, and no face to face meetings. The coordination among developers during OSSD is done by email and bulletin boards [7]. In OSSD there is an originator, who initiates the project, and invites others to participate in the project. The originator of project

makes the code available for the developers in order to precede the development process. So that anyone can be participate into system development. But the owner of the project (originator) is free to decide about the contributions to be included or not for official release of the project [8].

The feature of open source development includes the scheduling, code quality, unstable code, planned evolution, testing and preventive maintenance. There is little pressure of schedule to complete the project. Most of the developers are part time participators and they also do their full time jobs. Due to part time participation the development cycle may be long therefore the open source software is unaffected from time to market pressure. Without owner satisfaction, the open source software project cannot be released.

There is no fix code quality and standard in OSSD. It's very difficult to insist on particular standards. Often an unstable code is submitted by the developers as a contribution to the project. In Linux this problem is solved by synchronized development paths. There is a development release path and stable release path. When the new features are added then they are transferred into the stable release path. In OSD the requirement of active participation is not essential. Especially parallel debugging is used for maintaining code quality except systematic testing and other planning prescriptive approaches [9].

Open source software originated in 1950s and 1960s, when the software and hardware both were sold together. During 1950s and 1960s the source code of software was freely available without any restriction in IBM and DEC user group. In 1969 first version of UNIX was written by Ken Thompson, UNIX source code was accessible freely till in seventies. TEX software is developed by Donald Knuth. In 1979 UNIX was commercialized. Eric Allmann developed a system of communication between computers over ARPANET. In 1980 the software commercialization increased [10].

In 1983 GNU Manifesto for free software was published by Richard Stallman to establish the free software foundation. In 1986 Perl (Practical Extraction and report Language) a flexible programming language was developed by Larry Wall. CGI (common Gateway Interface) script has written with Perl. CGI is the connection (or interface) between a form on a Web page and the Web server. In 1987 Linux which is the version of UNIX was released with complete source code by Andrew Tanenbaum for the PC, Mac, Amiga, and Atari ST. In 1991 new version of UNIX 0.02 was developed by Linus Torvald, which is called LINEX. In 1993

Free BSD version 0.1 was released. Debian Linux that is the new Linux distribution was developed by Lan Murdock. Red Hat Linux that is leading Linux distribution is created by Marc Ewing in 1994. Apache group created the web server in 1995 that is leading HTTP server today. Netscape released the Mozilla with source code in 1998. Some of the major Software vendors decided to port their product to Linux. In 1999 users of Linux had reached to 7.5 Million [10].

From 1970-2000 the OSS was in progress without a name during these three decades. Many of the successful and broadly used software packages were developed in this period [11]. A survey was conducted in Japanese software industry to know the existing position of OSSD in 2001. The survey was conducted by SRA a well reputed company. From 1987 (FSF) is supported by SRA. In different countries SRA highlighted the OSS projects of different categories and evaluated the present business and governmental support for OSSD [8].

It was reported by Gartner that growing rate of Linux is fastest, for server market. "Approximately 50% of the web sites run on Apache web server". Yet the OSSP are facing lot of problems. It is reported by Source forge portal, which is a major host of Open Source projects, that in March 2009 out of 158669 registered projects only 17% projects were stable and 1.52% projects were reached at mature status. In OSSD the failure and success rates depend upon the open source community. In OSD the developers are physically distributed and hardly ever meet face to face. They communicate through email and bulletin boards [12].

Growing rate of OSS projects is increasing quickly. It has been estimated that OSS systems have 500,000 users, which are increasing with the rate of 700 per day. The OSS projects are also increasing with the rate of 60 new project per day [13][14]. It is estimated by Gartner group that 85% of enterprise companies have adopted the OSS systems and remaining 15% are moving to adopt in the next few years [15]. A survey was conducted about the usage of OSS, which reports that in 13 European countries the usage rate of OSS was 78%. Survey was also conducted in US and reported that 87% of organizations are using the OSS [16].

Godfrey and Tu reported that Linux has very huge line of code (two millions). Its growth rate is "super-linear". It is analyzed that the growth rate becomes low when commercial systems have become larger. Godfrey and Tu's suggest that "OSS systems have a growth rate that is much greater than that of traditional systems" [17]. Stallman reports that "worst threat to the

free/open source software community comes from the use of software patents instead of copyright as a means of protecting intellectual property rights" [11].

2. Need of SLR

According to Andreas Bauer open source software development might not be the "silver bullet" of software engineering because; it also has some shortcomings and troubles [20]. F/OSS has faces many "serious challenges" that have directed the software engineering community towards ultimately failure [21]. Even with lot of technological successes, OSSD has also faces lot of "fundamental challenges". OSS systems have been criticized due to these challenges [22]. It would be tremendously profitable to improve the design and architectural support for OSSD and limitations of the OSSD methodology [23].

By these facts and figures we can see the importance of systematic literature review of issue analysis of OSSD. Research and evidence shows that OSSD is very successful to produce the high quality software, but still there are some issues which result in a failure. By resolving these issues we can get success.

3. Research Questions

RQ1) What are the reported issues in open source software development?

RQ2) What strategies are reported to deal with these issues?

3.1 Research Questions Structure

Petticrew and Roberts suggest using the PICOC (Population, Intervention, Comparison, Outcome, and Context) criteria to frame review question [19].

Table 2 : PICOC Criteria of Research Questions

PICOC	RQ1	RQ2
Population	Open Source Software Projects	
Intervention	Open Source Software Development	
Comparison	------	------
Outcome	Reported Issues	Reported Strategies
Context	Industry, Practitioners	

4. Identification of Research Literature

Identification of research literature includes the following activities:

- Source selection

- Search terms
- Documenting the search

4.1 Source Selection

The following electronic sources will be searched for the collection of evidence in literature because most relevant papers exist in these databases and these are the major databases of software engineering [01].

- IEEE
- ACM Digital Library
- ScienceDirect
- SpringerLink
- ISI web of knowledge
- Google scholar

Other sources of evidence include the following material [03]:

- Reference lists from relevant primary studies and review articles
- Contacting experts and researchers working in the area and asking them if they know of any unpublished work

4.2 Search Terms

Search terms will be applied on the title and abstracts of papers. The strategy used to construct search terms is as follows [18].

a) Drive major terms from the questions by identifying the population, intervention, and outcome;
b) Identify alternative spellings and synonyms for major terms. If any terms were identified via consultation with expects in the field and/or subject librarians;
c) Check the keywords in any relevant papers we already have downloaded;
d) Use the Boolean OR to incorporate alternative spellings and synonyms and
e) Use the Boolean AND to link the major terms from population, intervention, and outcome.

Results of a):

Table 3 : Major Terms from PICO Criteria

PICO	Major Terms
Population	Open Source Software projects
Intervention	Open Source Software Development
Comparison	----- ----- -----
Outcome	Issues, Strategies

Result of b): Synonyms of major terms and alternate spellings.

Table 4 : Synonyms of Major Terms & Alternate Spellings

Major term	Synonyms
Open Source Software Development	open source software development, Free software development, Free/OSSD, Free/Libre
Issues	Challenges, Drawback, limitation, disadvantage, risk, problem, cons, flaw, lack

Result of c): Keywords in relevant papers already we have downloaded.

- Paul J. Adams, Andrea Capiluppi and Cornelia Boldyreff (2009) keywords: Free/libre and open source software development, Issues of communication, issues of productivity, issues of coordination.

Result of d): Use of the Boolean OR to link the major terms from population, Intervention and outcome.

1) {open source software development projects} OR {Free software development} OR {Free/OSSD} OR {Free/Libre}
2) {Drawback} OR {limitation} OR {disadvantage} OR {risk} OR {problem} OR {issues}
   {Challenges} OR {cons} OR {flows} OR {lacks}

Result of e): Use the Boolean AND to link the major terms from population, intervention and outcome.

Separate search string has been designed for each data source to extract results. The following is a general query and then specific queries are listed below in table 2.4 Appendix A.

4.3 General Query

({open source software development} OR {free software development} OR {Free/OSSD} OR {Free/Libre}) AND ({Drawback*} OR {limitation*} OR {disadvantage*} OR {risk*} OR {problem*} OR {con*} OR {drawback*} OR {flaw*} OR {lack*})

4.4 Documenting the Search

As the process of performing a systematic review must be transparent and replicable [01] therefore the search process should be well documented. The search process of this review will be documented in the following way.

Table 5 : Search Process Documentation [01]

Data source	**Documentation**
Digital library	Name of database: IEEE Search strategy for the database: • Only English papers • Search criteria is metadata only Date of search: 15-09-10
Conference proceedings	Title of proceedings Name of conference (if difference) Journal name (if published as part of a journal)
Other source	Data search/contacted URL Any specific condition pertaining to the search

The individual research paper record will be stored in an electronic database created in Microsoft access.

Table 6: Individual Study Form

Study ID	
Paper Title	
Author(s)	
Journal / Proceeding	
Volume	
Issue	
Pages	
(Title/Abstract based relevance)	
Publication year	
Search Date	
Search Resource	

For recording the integrated results of all data sources, we developed a form that will maintain the count of total studies found in each data sources.

Table 7 : Integrated Search Results

Basis of selection	No. of papers found in each database					
	IEEE	ACM	SPRINGERLINK	SCIENCE DIRECT	ISI WEB OF KNOWLEDGE	Total
Abstract						
Full text						

4.5 Endnote

Endnote X2 is used to manage the record of research papers. We save the record of all primary studies in Endnote X2 in the following categories.

- Accepted Studies
- Rejected Studies
- Duplicate Studies

5. Study Selection Criteria and Procedures

5.1 Inclusion Criteria

Research papers will be included on the basis of reading their titles and abstracts.

- Research papers relevant to open source software development issues will be included
- Primary studies written in English language will be considered
- Only case / field studies, experiments and experience/industrial reports will be included for gathering evidence from literature
- Research papers which are based on the expert opinion will be included
- Research papers related to the topic, which do not provide evidence, will be included as weak evidence

5.2 Exclusion Criteria

The following type of papers will be excluded.

- Textbooks and web pages will be excluded
- Primary studies of simple open source software will be excluded
- If a paper is published in several conferences or journals then the most complete version, on the basis of studies discussed in the article, will be included

5.3 Study Selection Process

Search strings will be used which are already identified. Primarily titles and abstracts of papers will be studied and then relevant article will be identified. If paper title and abstract is not relevant then it will be excluded. Those papers having relevant titles then their abstracts will be studied and if any paper does not provide evidence then those papers will be included as week evidence. The results against each string will be recorded. The results will be tabulated in the following way [03]:

- Number of papers per source
- Number of candidate papers per source
- Number of selected papers per source

Relevant and irrelevant studies will be selected or rejected and then consulted with the supervisor. A list of rejected papers will be maintained with reasons for rejecting the papers.

6. Study Quality Assessment Checklists and Procedures

Study qualified assessment criteria is a detailed criterion for filtering the research papers further more. Each paper will be assessed after full reading. Here we have been designed separate quality assessment checklists for case/field studies, experiment/industry reports and experiments.

Table 8 : Quality Assessment Criteria (Qac) Checklist [01][18]

ID	QUALITY ASSESEMENT CRITERIA	FEEDBACK	SCORE
Common criteria for all papers			
1.	Is the study relevant to the topic?	Y/N/P	
2.	Does the study review the related work for the problem?	Y/N/P	
3.	Are the finding systematically reported and sufficient evidence reported to justify the relationship between evidence and conclusion?	Y/N/P	
4.	Is the objective of study clearly defined?	Y/N/P	
5.	Is the design of study clearly defined?	Y/N/P	
6.	Does the study clearly define the nature of open source software projects?	Y/N/P	
7.	Does the study clearly state the issues of open source software development?	Y/N/P	
8.	Is it clear which open source software development issue is resolved by the reported strategy?	Y/N/P	

The scale for evaluating the quality of all these questions will be:

- Y (YES) = 1
- P (PARTIAL) = 0.5
- N (NO) = 0

6.1 Study Quality Assessment Procedure

Studies which will pass the inclusion criteria then a detailed quality assessment will be done on those selected studies. Candidate primary studies will be obtained after applying a detailed quality assessment. Full reading of selected candidate primary studies will be required. After a full reading, then data will be extracted.

7. Data Extraction and Synthesis

The objective of this stage is to design data extraction form to accurately record the information researchers obtained from the primary study. To reduce the opportunity for bias, data extraction forms have been defined and piloted when the study protocol is defined [03].

7.1 Required data

The information required for each primary study which is selected after applying quality criteria and inclusion criteria, has been mentioned in the data extraction forms. The data will be extracted by the one researcher and checked by another researcher or supervisor.

Table 9 : Data Items Extracted For Each Study [18]

Data item	Value	Additional notes
Title of study		
Year of publication		
Name of database		
Type of study		Case study / field study / Experiment / industry Experiment report
Aim of paper		
Is the research question mentioned clearly		Yes /No
What issues are resolved?		Issue reported and focus
Major contribution		Significance /importance of study
What strategies are reported against the open source software development issues in paper?		Strategies means technique/ method/ tool/methodology/ Framework
What measures are used for comparing		Criteria for comparisons

results?		
How results are represented?		Graphical/tabular/descriptive
Are results valid?		
Is research question answer properly? If not then what is missing?		
Does strategy answer the issue? If yes to what extent?		
Additional Notes		
Further work		

Table 10 : Data Item Extracted from All Research Papers [01]

Data item	Value
Data extractor	
Application domain(s)	
No. of all relevant studies in all database	
No .of studies extracted from other sources	
Total no of relevant studies	
No .of duplicate studies	
No of studies left after duplicate study	
No of studies' relevant to open source software development	
No .of studies left after applying inclusion/ exclusion criteria	
No. of studies left after applying quality assessment criteria	

7.2 Data Extraction Process

The data extraction form will be filled by the data extractor and then it is checked by the data checker. The data checker will be the supervisor or other researcher. The data checker will sure the accuracy of data extracted from the studies. If any disagreement found, then the form will be revised. Separate forms will be mentioned for each study selected after study selection and quality assessment criteria.

8. Data Synthesis

The aim of data synthesis is to integrate and summarize the data collected from the studies. Data synthesis form will be used to provide such kind of information. The following table assigns the issue ID to the reported issues in different papers.

Table 11 : Assigning IDs to the reported Issues

Issue ID	Description	Additional Notes
I1	Winwin	Winwin
---	---	---

The following table assigns the strategy ID to the strategies reported in different papers.

Table 12 : Assigning IDs to the reported Strategies

Strategy ID	Description	Additional Notes
ST1	Winwin	Winwin
---	---	---

The following table provides a list of open source software development issues reported in the different papers.

Table 13 : Data Synthesis Form

Study ID	Quality Score	Database	Year of publication	Issue ID	List of issues
SD1	1	IEEE	2007	I1	Winwin
---	---	---	---	---	---

The following table provides cumulative information about the total no of studies which reported each open source software development issue. The table corresponds to the RQ.1.

Table 14 : Counting Study IDs Covering Reported Issues

Issue ID	Study ID that cover the issue	Total no of study IDs
I1	SD1,SD2	2
---	---	---

The following table provides the cumulative information about the total number of studies which reported strategies for open source software development issue. This table corresponds to RQ.2.

Table 15 : Counting Study IDs Covering Reported Strategy

Issue ID	Strategy ID	Reported strategy	Study cover this strategy	Total no of study IDs
I1	ST1	Clarify the goal	SD1,SD2	2
---	---	---	---	---
			Grand total	2

9. Data Analysis

Summary of each study's information will be represented in the tabular form in a particular order (the most recent study will come first). The number of studies will also be counted.

Table 16 : Layout of Answer of RQ.1

Issue ID	Reported issue	Reference
I1	Winwin	Winwin
---	---	---

Table 17 : Layout of Answer of RQ.2

Issue ID	Strategy ID	Reported strategy	Reference
I1	ST1	Winwin	Winwin
---	---	---	---

10. Dissemination Strategy

The results of systematic review will be convincing for both software engineers group and researchers. The results of systematic review will be written in "*Technical Report*" format. The report will be reviewed by the internal SIG committee of "International Islamic University Islamabad" and external committee as well. A short version of report will be send to practitioners for their comments. We will try to publish it in journal and/or conference.

11. Research Plan

Table 18: Project Time Table

ID	TASK NAME	START	FINISH	DURATION
1	Evaluate the review protocol	15-09-2010	14-10-2010	4W
2	Identification of research	15-10-2010	03-12-2010	7W
3	Study selection criteria	04-12-2010	21-01-2011	7W
4	Study quality assessment	22-01-2011	11-03-2011	7W
5	Data extraction	12-03-2011	29-04-2011	7W
6	Data synthesis	30-04-2011	27-05-2011	4W
7	Specifying dissemination mechanisms	28-05-2011	10-06-2011	2W

8	Formatting the main report	11-06-2011	25-06-2011	2W
9	Evaluating the report	26-06-2011	10-07-2011	2W

References

[1] B.Kitchenham, "Procedure for performing systematic review", joint Technical Report, Computer Science Department, Keele University (TR/SE-0401) and National ICT Australia Ltd, 2004.

[2] Tore Dyba, "Evidence-Based Software Engineering for paractitioners" Simula Research Laboratory and SINTEF Information and Communication Technology.

[3] B.Kitchenham, and S.Charter, "Guidelines for performing systematic literature reviews in software engineering", Version 2.3, Keele University, EBSE Technical Report, EBSE-2007-01,July 9,2007.

[4] Cristina Gacek, Tony Lawrie, and Budi Arief, "The many meanings of Open Source".

[5] Ioannis Stamelos, Lefteris Angelis, Apostolos Oikonomou & Georgios L. Bleris, "Code quality analysis in open source software development".

[6] Walt Scacchi, "Free/Open Source Software Development: Recent Research Results and Methods" Institute for Software Research, Donald Bren School of Information and Computer Sciences, University of California, Irvine.

[7] Audris Mockus Roy T. Fielding James Herbsleb, "A Case Study of Open Source Software Development: The Apache Server".

[8] Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, Yunwen Ye , "Evolution Patterns of Open-Source Software Systems and Communities".

[9] By Michael W. Godfrey and Qiang Tu, "Evolution in Open Source Software: A Case Study". Software Architecture Group (SWAG), Department of Computer Science, University of Waterloo.

[10] Alexander Hars and Shaosong Ou, "Working for Free? – Motivations of Participating in Open Source Projects". IOM Department Marshall School of Business University of Southern California.

[11] David Bretthauer, "Open Source Software: A History" University of Connecticut, dave.bretthauer@uconn.edu.

[12] Chintan Amrit, "Exploring the Impact of Socio-Technical Core- Periphery Structures in Open Source Software Development". Faculty management and Governance, Department of Information Systems and Change Management, University of Twente.

[13] S. Elliott and Walt Scacchi, "Free Software Development: Cooperation and Conflict in a Virtual Organizational Culture Margaret".

[14] S. Elliott and Walt Scacchi, "Free Software: A Case Study of Software Development in a Virtual Organizational Culture by Margaret".

[15] Sandro Morasca, Davide Taibi, Davide Tosi, "Towards certifying the testing process of open source software: new challenges or old methologies?".

[16] Brian Fitzgerald, Lero, "Open Source Software Adoption: Anatomy of Success and Failure". Irish Software Engineering Research Centre and University of Limerick, Ireland.

[17] Trung Dinh-Trong and James M. Bieman, "Open Source Software Development: A Case Study of FreeBSD". Software Assurance Laboratory, Computer Science Department, Colorado State University

[18] B.A.Kitchenham, E.Mandes and G.Travassos, "protocol of systemtic review of within-and , cross-company estimating models" Version 14, 2006, URL:http: www.dur.ac.uk

[19] Petticrew, Mark and Helen Roberts. Systematic Reviews in the Social Sciences: A Practical Guide, Blackwell Publishing, 2005, ISBN 1405121106

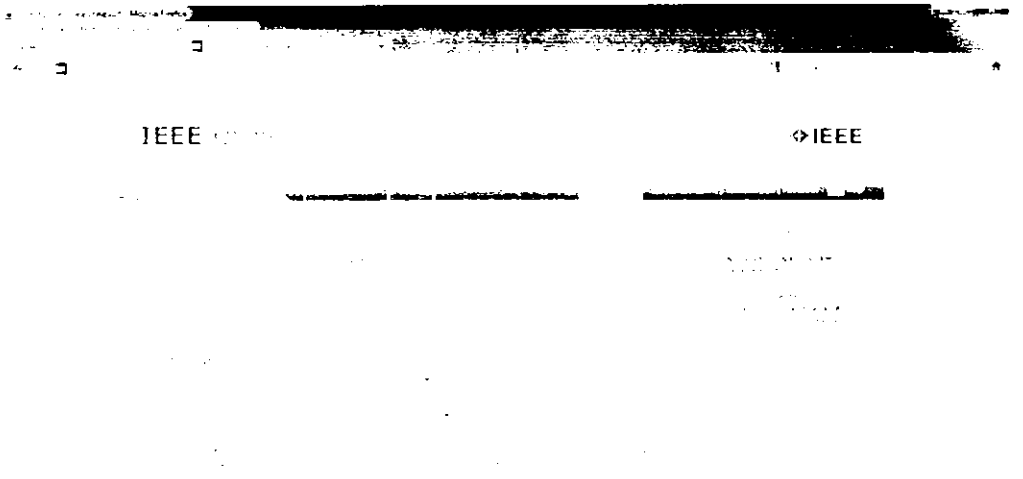
[20] Andreas Bauer, Markus Pizka. (2002). The Contribution of Free Software to Software Evolution. Proceedings of the Sixth International Workshop on Principles of Software Evolution (*IWPSE'03)*, (pp. 1-10)

[21] A Critical Look at Open Source by Brian Fitzgerald, University of Limerick

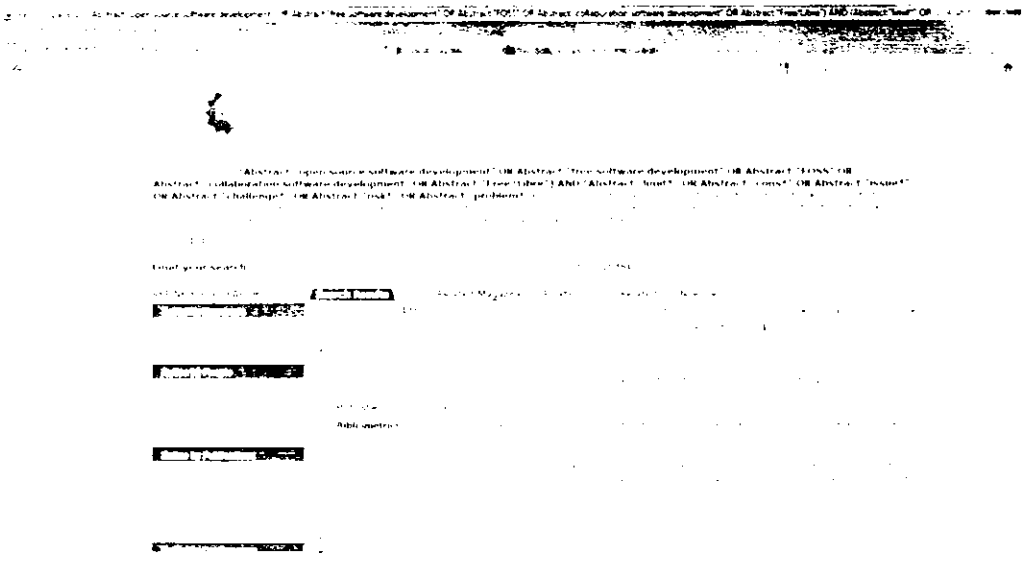
[22] Morten Sieker Andreasen, Henrik Villemann Nielsen, Simon Ormholt Schrøder, Jan Stage. (2006). Usability in Open Source Software Development: Opinions and Practice. Information Technology and Control Vol.35, No.3A, 303-312.

[23] Vijay K. Gurbani, Anita Garvert, James D. Herbsleb. (2006). A Case Study of a Corporate Open Source Development Model. ICSE'06. Shanghai, China.

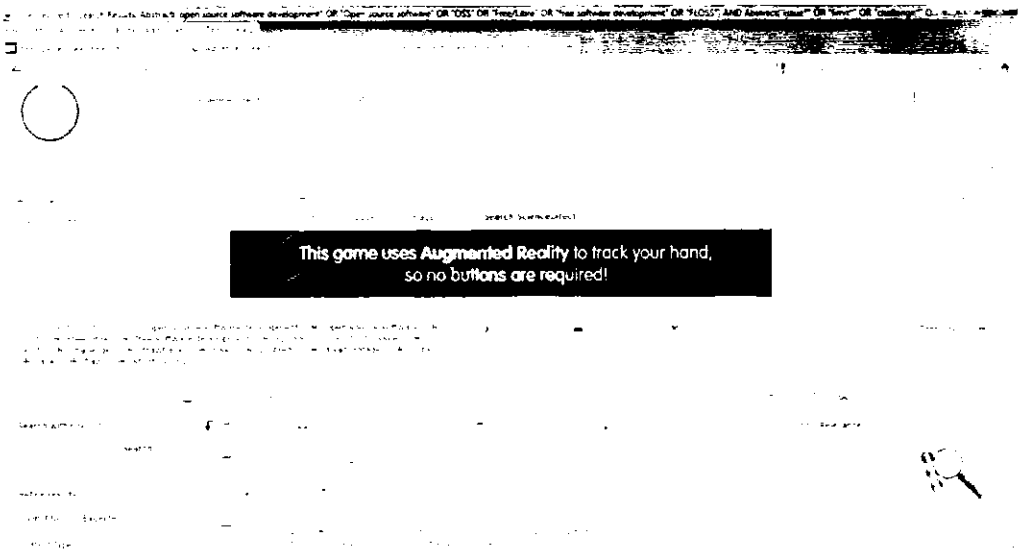
IEEE:



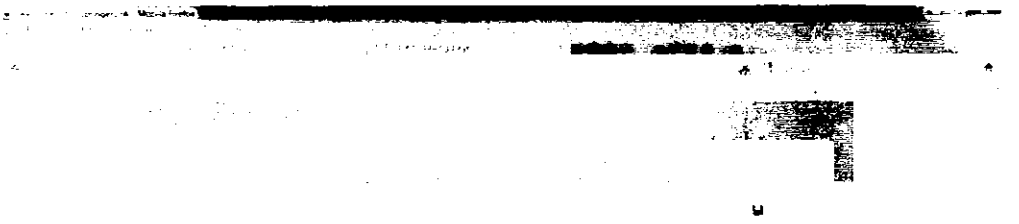
ACM:



SCIENCE DIRECT:



SPRINGERLINK:



International Islamic University