# Evolution of Open Source Software Projects: A Systematic Literature Review

M.M. Mahbubul Syeed, Imed Hammouda, Tarja Systä
Department of Pervasive Computing, Tampere University of Technology, Tampere, Finland
Email: {mm.syeed, imed.hammouda, tarja.systa}@tut.fi

***Abstract*— Open Source Software (OSS) is continuously gaining acceptance in commercial organizations. It is in this regard that those organizations strive for a better understanding of evolutionary aspects of OSS projects. The study of evolutionary patterns of OSS projects and communities has received substantial attention from the research community over the last decade. These efforts have resulted in an ample set of research results for which there is a need for up-to-date comprehensive overviews and literature surveys.**

**This paper reports on a systematic literature survey aimed at the identification and structuring of research on evolution of OSS projects. In this review we systematically selected and reviewed 101 articles published in relevant venues. The study outcome provides insight in what constitutes the main contributions of the field, identifies gaps and opportunities, and distills several important future research directions.**

***Index Terms*— Open Source; Evolution; Systematic Literature Review.**

## I. Introduction

Research on Open Source Software (OSS) has gained momentum over the last decade as commercial use of OSS components continues to expand [1]. Much of the research has focused on evolutionary aspects of open source development in answer to long-term sustainability and viability concerns of community-based software projects [2].

Examples of such research include collecting experiences and building theories of OSS adoption in terms of planning, process improvement, community involvement and software maintenance [3] [4]. Often well-established theories of software evolution, such as Lehman's law [5], are studied in the context of OSS to assess evolutionary and quality characteristics such as survivability, growth potential, maintainability, and ease of adoption.

To keep track of the latest research findings in the area of OSS evolution, there is a need for comprehensive literature studies that summarize and structure the existing body of knowledge. In this article, we present a study for systematic selection, characterization and structuring literature that concerns evolution of open source projects. Our objective, and thus contribution is to produce a systematic reporting of what constitutes the key contributions, the main research gaps, and potential future directions in the field.

In order to perform the study, we have adopted a systematic literature review (SLR) approach [6] for the systematic selection and characterization of existing literature. SLR is a recommended methodology for aggregating knowledge about a specific software engineering topic or research question [7] [8], through the systematic analysis of relevant empirical studies [6]. For example, SLRs were popularly utilized to acquire, conceptualize and structure knowledge in various fields of software engineering including, dynamic analysis [9], fault prediction [10], global software engineering [11], and business process adoption [12].

To carry out this review we adopted a review protocol following the guidelines suggested in [13] and the survey process used in [9]. Keeping the research motivation in mind, we posted 11 research questions in four categories, e.g., target, approach, target group and outcome. Target refers to the different facets and dimensions of OSS projects explored; approach refers to the method, metrics, and tools used for the study; target group refers to the domain of OSS projects studied with selection motives, and finally the outcome refers to the findings and validation of the results reported in the articles. We also discuss the implications of the findings and provide recommendations for future research. The data extracted from the articles are documented under the attribute set developed for answering the research questions. This data is provided in the review website [14] and can be used by the research community to get a holistic view on OSS evolution studies.

The paper is organized as follows: In Section II we discuss the review protocol and the research questions. Answers to the research questions, and a discussion on open areas in the field of OSS and evolution are presented in Section III and IV respectively. Section V discusses validity issues related to the review process. Finally, concluding remarks are presented in Section VI.

## II. Review Methodology

Evidence-based Software Engineering (EBSE) relies on aggregating the best available evidence to address engineering questions posed by researchers. A recommended methodology for such studies is Systematic Literature Review (SLR) [6]. Performing an SLR involves several discrete tasks, which are defined and described by Kitchenham in [13]. As a starting point, SLR recommends to pre-define a review protocol to reduce the possibility of researcher bias [13]. Along those guidelines and following

This work was supported in part by the Nokia Foundation Grant and TiSE graduate school funding, Finland.


doi:10.4304/jsw.8.11.2815-2829

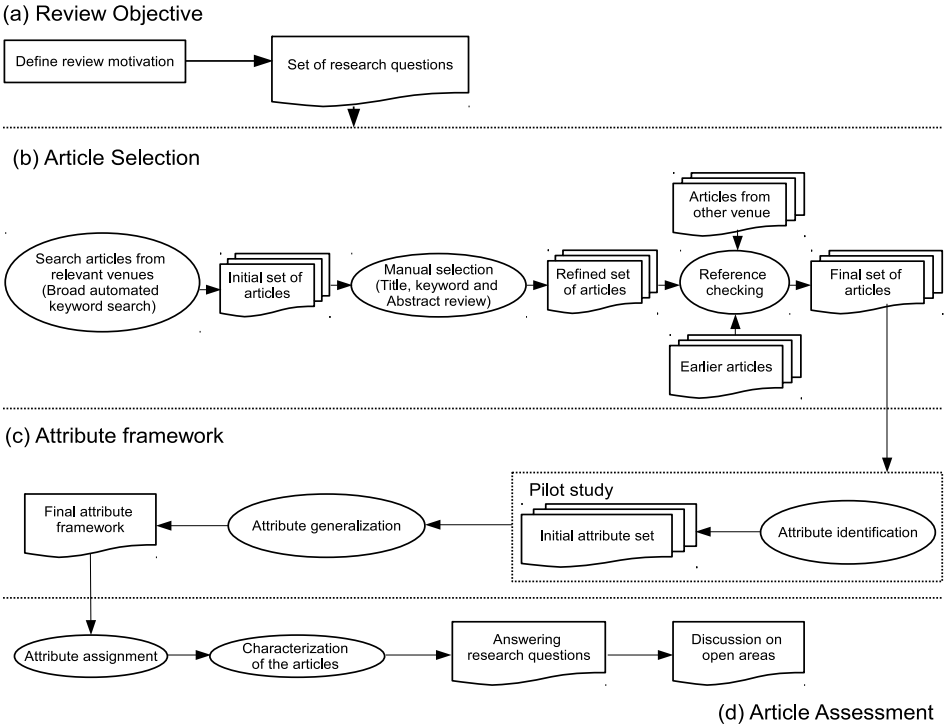


Figure 1. Overview of systematic literature review

the review process described in [9], Figure 1 shows the tasks involved in the review protocol of this study. The tasks are discussed in the subsequent subsections.

### A. Research Questions

The research questions we have defined fall within the context of OSS projects and their evolution strategies. In total we have formulated 11 questions, as presented in Table I. These questions are proposed to portray the holistic view of OSS evolution studies. This covers aspects like the focus of the study, methodological detail, case study projects, data sources, and validation mechanisms.

### B. Article Selection

This section describes the article selection process (phase (b) in Figure 1) that includes defining the inclusion criteria for article selection, an automated keyword search process to search digital libraries, a manual selection from the initial set of articles, and the reference checking of the listed articles.

**Inclusion criteria**. Along the research questions shown in Table I, we have defined the following selection criteria in advance that should be satisfied by the reviewed articles:

- Subject area of the articles must unveil strong focus on evolution of OSS projects. Authors must explicitly state the target of the study (e.g., software evolution, community evolution, co-evolution, prediction) and provide detail evidence of research methodology, data sets, and statistical detail of case study projects.
- Articles must exhibit a profound relation to OSS projects and take into consideration those aspects that are particularly attributed to the OSS community and projects. Articles using OSS as a case study are taken into account only if they satisfy the above criterion.
- Articles published in referred journals and conferences are included for the review. Similar to most SLRs, books are not considered for the review.

The suitability of the articles was determined against the above mentioned selection criteria through a manual analysis (discussed later in this section) of title, keywords, abstract. In case of doubt conclusions are checked [15].

**Automated keyword search**. Automatic keyword search is a widely used strategy in literature surveys [16] [17]. Thus we performed a broad automated keyword search to get the initial set of articles. First author of this article was responsible for the search process. Seven digital libraries were searched: IEEE Computer Society Digital Library; ACM; ScienceDirect; SpringerLink; Google Scholar; FLOSShub and Mendeley. These libraries are the popular sources for open source related research articles. All searches were based on the title, keywords and abstract. The time period for this search was from January, 2000 to January, 2013.

Knowing the fact that construction of search strings varies among libraries, we first defined search terms according to our inclusion criteria. Then to form the search strings, we combined these search terms following the guidelines of the digital library searched. The list of search terms that were used is as follows.

TABLE I.
RESEARCH QUESTIONS

| Category | Research Questions | Main Motivation |
|---|---|---|
| Target | Which facets of OSS projects were explored and what statistical distribution the articles have in those facets? | To decompose the articles according to their study focus and intensity of studies in each focus area. |
| | What are the dimensions of OSS projects explored under each study facet? | To determine the specific aspect(s) of OSS projects explored in evolution studies within each facet. |
| Approach | What are the research approaches followed in the studies? | To identify the general research approach followed in evolution studies (e.g., empirical studies with quantitative or qualitative data analysis). |
| | What are the datasets or data sources of OSS projects mostly exploited in evolution studies? | To identify the data sources of an OSS project that are used for the evolution studies. |
| | What metric suits are evaluated and what tools are used for metric data collection? | To explore the metric suits used for evolution study and the popularly used tools for data extraction. |
| Target group | What is the portfolio of projects analyzed for evolution studies and what are their domains? | To determine the mode of evolution studies (e.g., horizontal or vertical) by statistically measuring the studied OSS projects and their domains. |
| Outcome | Does the concern on "OSS evolution study" follow an increasing trend? | To identify the beginning and growth of research interest in the field OSS project evolution. |
| | What contributions are made in literature to analyze the evolution of software? | To explore what results are presented to enhance the understanding of OSS projects evolution. (e.g., do evolution of OSS projects conforms to the theory of software evolution?) |
| | What contributions are made in literature to analyze the evolution of organization or community? | |
| | What contributions are made in literature to analyze the interdependency in the evolution of the software and organization? | |
| | How are the research approaches and results of the articles typically validated? | To identify the approaches employed to evaluate the research approaches and study results (e.g., internal validity, external validity, construct validity). |

Terms representing OSS: "Open source" or OSS or "Open Source Software" or "Open Source Software projects" or FLOSS or "Libre Software" or "F/OSS".

Terms representing evolution study: "evolution" or "structural evolution" or "evolution of software" or "project evolution" or "project history" or "software evolution" or "community evolution" or "co-evolution".

Automated keyword search ended up with 181 articles consisting of 46 journal articles and 135 conference articles.

**Manual selection**. Recent studies [15] [9] pointed out that (a) current digital libraries on software engineering do not provide good support for automated keyword search due to lack of consistent set of keywords, and (b) the abstracts of software engineering articles are relatively poor in comparison to other disciplines. Thus it is possible that the 181 articles identified through automated search process might contain irrelevant ones and some relevant might be missing. Due to this fact the first author performed a manual selection on these articles by reviewing the title, keywords and abstract (and in case of doubt, checking the conclusion [15]). To reduce the researcher bias in this selection process, the domain experts (second and third author) examined the selected articles against the selection criterion. Any disagreement was resolved through discussion. This process ended up with 97 articles consisting of 21 journal articles and 76 conference articles.

**Reference checking**. To ensure the inclusion of other relevant but missing articles (as mentioned above), the first author performed a non-recursive search through the references of the 97 selected articles. This process identified 4 additional conference articles.

**Final set of articles**. The article selection process finally ended up with 101 articles (21 journal and 80 conference articles). A complete list of these articles along with year and venue wise distribution can be found in our review website [14].

### C. Attribute Framework

The next step in the review protocol was the construction of an attribute framework (phase (c) in Figure 1). This framework was used to characterize the selected articles and to answer the research questions. Following is a brief description of this process.

**Attribute identification**. The attribute set was derived based on two criteria: (a) The domain of the review (i.e., evolution of OSS projects) and (b) the research questions. A pilot study was run for this step, as shown in phase (c) of Figure 1. This phase consists of a number of activities.

First, we performed an exploratory study on the structure of 10 randomly selected articles (from the pool of 101 articles). This study led to a set of eight general attributes that can be used to describe the articles and to answer the research questions. This attribute list is shown in the *Attribute* column of Table II.

Second, this list of attributes was refined further into a number of specific sub-attributes to get precise

TABLE II.
ATTRIBUTE FRAMEWORK

| Attribute | Sub Attribute | Brief Description |
|---|---|---|
| General | | Publication Type, Year of Publication |
| Study Type | | Empirical, comparative, case study, tool implementation. |
| Study Target | Software evolution<br>Community evolution<br>Co-evolution<br>Prediction | Code, architecture, bug/feature<br>Developer and user community<br>Combined evolution of software and community<br>Studies on predicting evolution of OSS projects |
| Case Study | OSS projects studied<br>Programming language<br>Project size<br>Project domain | List of OSS projects studied<br>Target programming languages of OSS projects<br>Size measure of OSS projects (in KLOC for latest release)<br>Application domain of the OSS projects covered |
| Data Source | Source code<br>Contributions<br>Communication<br>External sources | Code base, CVS/SVN<br>Change log, bug tracking systems<br>Mailing list archive, chat history<br>Sourceforge, github, ohloh. |
| Methodology | Methods<br>Metrics<br>Tool implementation<br>Tools used | Concrete methods applied<br>Type of metrics used<br>Tools implemented for the study<br>Existing tools, algorithms used for study |
| Results | Growth rate<br>Measure of evolution<br>Prediction classification<br>Summary | Defines the growth rate of an OSS project during its evolution.<br>Qualitative, Quantitative<br><br>Other findings |
| Evaluation / Validation | | Validation process for a study |

description of each of the general attributes and fine tune the findings on the research questions. To do this, we made a thorough study of the same set of articles and wrote down words of interest that could be relevant for a particular attribute (e.g., "software evolution", or "community evolution" or "co-evolution" for *Study target* attribute). The result after reading all articles was a (large) set of initial sub attributes. This data extraction task was performed by the first author of this survey.

**Attribute generalization and final attribute framework**. We further generalized the attributes and sub-attributes to increase their reusability [9]. For example, sub-attributes "mailing list archive" or "chat history" are intuitively generalized to *Communication*. This final attribute list was then examined and validated by the domain experts (second and third authors). This reduces the change of researcher bias, as neither of the domain experts had any connection with this process. The final attribute framework is shown in Table II.

### D. Article Assessment

The article assessment step consists of four distinct activities as shown in phase (d) of Figure 1. In this section we focus on the first two steps.

**Attribute Assignment**. Using the attribute framework from the previous section, we processed all articles and assigned the appropriate attribute sets to each of the articles. These attributes effectively capture the essence of the articles in terms of the research questions and allow for a clear distinction between (and comparison of) the articles under study.

The assignment process was performed by the first author of this survey. During this process, authors' claim of contribution is assessed against the results presented in the articles. For example, to validate the claim on the target of the study (e.g., software or community evolution), we assessed what relevant data sources are explored, what metrics and methods are used, and the duration and process of data collection. Also, we did not draw any conclusions from what was presented in an article if it was not explicitly mentioned. For example, we left the attribute field *study type* empty if it was not mentioned in the article.

**Characterization of the reviewed articles**. Since the attribute assignment process is subject to different interpretations, different reviewers may predict different attribute subsets for the same article [9]. As the attribute assignment process is carried out by the first author of this paper, the quality of the assignment needed to be verified to avoid reviewer bias [9]. This verification task was carried out by the domain experts who assessed the data collection table against the reviewed articles. Any disagreements were resolved through discussion. This characterization of articles is presented in our review website [14].

Next we discuss the results of this review by answering the research questions and discussing open areas in this field.

## III. REVIEW RESULTS

Given the article selection and attribute assignment (as presented in review website [14]), the next step is to present and interpret the study findings. We start with discussing answers to the research questions based on the study outcome. List of OSS projects that are studied in the review articles are provided in the website [14].

*RQ1. Which facets of OSS projects are explored*

*and what statistical distribution the articles have in those facets?*

An in-depth study on the selected articles led us to decompose the OSS evolution articles into four facets:

- Software evolution: articles under this facet explore evolutionary behavior of OSS systems and derive patterns of evolution to evaluate them against the laws of software evolution. Such studies also measure the issues that concern the commercial world. This includes for instance, study the evolutionary patterns of code complexity, maintainability, sustainability, and quality in an OSS project.
- Community evolution: articles under this facet studies how the social networks of developers and users evolve over time while building the product.
- Co-evolution: articles under this facet examine the evolution of OSS systems with the associated communities, and explore relationship between the two through different collaboration models.
- Prediction: articles under this facet deal with defining and examining prediction models to simulate the evolution of OSS projects. For instance, developing methods to support error prediction for the purpose of preventive maintenance and building quality software.

Figure 2 shows the distribution of articles (published in both journal and conferences) under each facet.

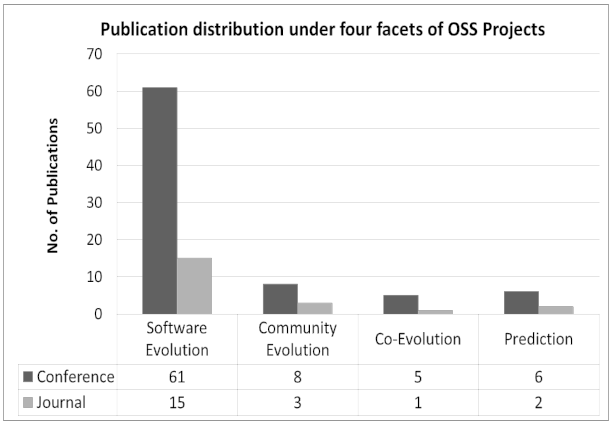


Figure 2. Article distribution under each facet of evolution study

From this figure it is evident that the facet *software evolution* got maximum attention over others. The reason of such bias distribution of articles can be defended by the fact that the development history of OSS projects is relatively new compared to its proprietary counterpart [18].

Software evolution is the most studied facet.

*RQ2. Does the interest on"OSS evolution study" follow an increasing trend?*

OSS development has appeared and diffused throughout the world of software technology, mostly in the last ten to thirteen years. During this period, a growth in interest for better understand the patterns of OSS evolution has been noticed. The increasing trend in number of publications between the year 2000 and 2012 (as shown in Figure 3) assist this claim.

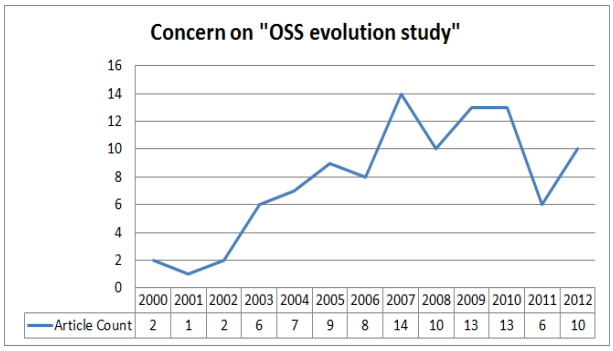


Figure 3. Concern on "OSS evolution study" over the decade

Research on OSS evolution follows an increasing trend.

*RQ3. What research approaches are followed in the studies?*

Research methodologies followed in the reviewed articles can be categorized into four distinct approaches: empirical study, case study, comparative study and tool implementation. Each of these studies use OSS project data for either quantitative analysis or qualitative analysis. Figure 4 shows the count of published articles according to this classification. As can be seen from the Figure, 75% of the studies (76 articles out of 101) followed empirical approach with either quantitative or qualitative data analysis.

| | Quantitative data analysis | Qualitative data analysis |
|---|---|---|
| Empirical Study | 72 | 4 |
| Tool implementation | 6 | 0 |
| Case Study | 4 | 6 |
| Comparative Study | 1 | 2 |

Figure 4. Distribution of articles under the classification of research approaches followed in the studies

Empirical research is the most frequent research methodology used to study OSS evolution.

*RQ4. What are the dimensions of OSS projects explored under each study facet?*

This research question gives a fine grained view on the dimensions of OSS projects explored by the evolution articles. Figure 5 provides a two dimensional view of OSS projects, e.g., code and community dimensions with their constituent parts. As can be seen from this figure,

software evolution and prediction facets mostly utilize the code dimension. Whereas the community evolution facet puts more emphasis on developer community than user community or their combination. The study on co-evolution of the code and community mostly explores the code base, bug reports, developer and user community.

| Facets | Code dimension | | | | Community dimension | | |
|---|---|---|---|---|---|---|---|
| | Code (LOC, Module, Function, package) | Architecture | Documentation | Bug | Developer community | User community | Both developer and user community |
| Sotware Evolution | 61 | 7 | 1 | 1 | # | # | # |
| Community Evolution | # | # | # | # | 11 | 1 | 1 |
| Prediction | 6 | 0 | 0 | 0 | 0 | 0 | 0 |

| | Code-developer | Code-developer and user | code-bug-developer | Code-bug-developer and user |
|---|---|---|---|---|
| Co-Evolution | 3 | 1 | 0 | 2 |

Figure 5. Dimensions of OSS projects that are explored to study a facet, # means not applicable

Code dimension (e.g., source code) is mostly studied in the articles as compared to community dimension or their combination.

*RQ5. What is the portfolio of projects analyzed for evolution studies and what are their domains?*

In general, the study of evolutionary behavior and patterns of OSS projects requires access to historical data representing their development, growth and success story. These studies thus delimited to flagship OSS projects that are large in size with a large user and developer community and belong to popular application domains. In this regard, our findings reported that most of the OSS projects studied are from the domain of Operating Systems (OS), Application Software, Integrated Development Environments (IDE), Application Servers, Libraries, Desktop Environments and Frameworks. Example projects under these domains include Linux, Eclipse, Apache, Ant, Mozilla, GNOME, KDE, and ArgoUML. These projects have more than 5 years of development and evolution history. Figure 9 in the appendix presents the domain wise classification of the studied OSS projects with the count representing their frequency of use in the evolution studies. This finding gives support to the fact that OSS evolution studies are mostly vertical and thus unable to put light on the whole population of OSS projects, as vast majority of projects are failures [19]. Only a few articles, according to our study, report horizontal studies with a large and random sample of OSS projects (ranging between 200 to 4000 OSS projects).

Large and successful OSS projects are often selected as case study projects.

*RQ6. What are the datasets or data sources of OSS projects mostly exploited in evolution studies?*

To analyze the evolutionary behavior of OSS projects, information contained in project data sources need to be explored. These data sources are termed as repositories which contain a plethora of information on the underlying software and its development processes [20] [18]. Studies based on such data sources offer several benefits: this approach is cost effective, requires no additional instrumentation, and does not depend on or influence the software process under consideration [20]. Evolution studies on the OSS projects effectively explored these repositories produced by the projects as well as the external sources. Figure 6 presents the OSS repositories in both categories and the count of articles that utilizes those repositories. According to this figure, repositories maintaining the code base (e.g., CVS/SVN, change log) are the most explored sources. This is obvious because most of the articles (as discussed in RQ1) studied either the evolutionary patterns or the prediction models for the evolution of the system. Among the external sources, SourceForge.net is the most popular repository hosting thousands of OSS projects and having the maximum number of downloads.

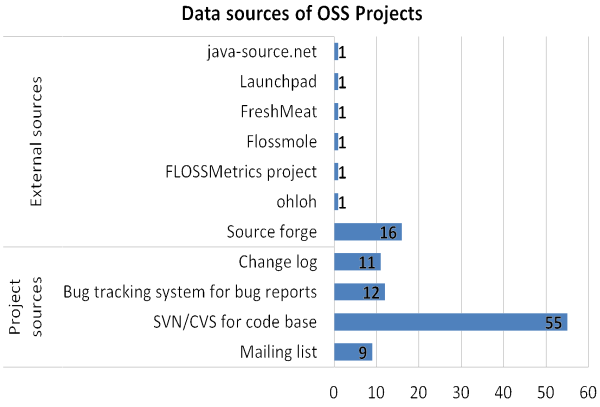


Figure 6. Data sources of OSS Projects

Repository maintaining the source code of the projects are mostly explored in the studies.

*RQ7. What metric suits are evaluated, and what tools are used for data collection and analysis?*

The reviewed articles used metrics mostly to measure the evolutionary patterns and antecedents of certain aspects of the studied projects, such as, code complexity, structural complexity, architectural patterns, predicting error proneness and maintainability, and collaboration patterns within the community. Mostly empirically validated metrics are selected for these studies. Widely used metric suites are listed in Table III.

For metric data collection, synthesis and interpretation, a number of existing tools are used. Figure 8 in the appendix provides a list of such tools used along with

TABLE III.
METRICS

| Metric Category | Example Metrics |
|---|---|
| Source code metrics | source line of code, line of code, number of functions |
| Code complexity metrics | interface complexity, Halstead suite of complexity metric, cyclomatic complexity, structural complexity |
| Object oriented metrics | Chidamber and Kemerer, L&K (Lorenz and Kidd's eleven metrics, Li's metric suite for OO programming, modularity metrics |
| Product level metrics | product size, releases, application domain, version frequency |
| Project metrics | metrics related to the OSS community structure and communication, application domain, number of developers, users, project popularity, success, application domain, no of commits, no of messages sent |

their usage area and popularity count. As most of these tools are third party applications, the accuracy of the data collection and analysis is constrained by the performance of these tools. This also puts impact on the validity of the results.

Empirically validated metric suites to evaluate the source code are mostly used in the articles.

*RQ8. How are the research approaches and results of the articles typically validated?*

Threats to validity in experimental studies can be categorized into three types: external, internal and construct validities [21]. External validity means to what extent the results can be generalized to the whole population (or outside the study settings). Internal validity means that changes in the dependent variables can be safely attributed to changes in the independent variables. Construct validity means that the independent and dependent variables accurately model the abstract hypotheses.

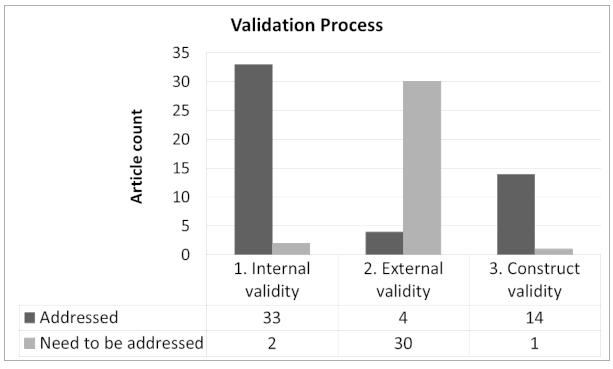


Figure 7. Validation Process of the research approaches

Our survey results concerning these validity issues reveal that 41% (42 out of 101) of the articles measured and reported the validity threats of the underlying research methodology and the research result. Figure 7 plots the number of articles that perfoms validity measure under each category. In this figure, the articles which took quantifiable measure to minimize a validity threat are counted under *addressed* field and the articles which admited the threats as a delinquent to the study are counted under *Need to be addressed* field.

Almost all the studies (30 out of 34) suffers from external validity threats and thus suffers from generalizability of the results to the population of OSS projects.

*RQ9. What contributions are made in literature to analyze the evolution of software?*

Analysis of the selected articles identifies a good sample of empirical studies which were conducted to verify the fitness of the Lehman's law of software evolution in the domain of OSS projects. These results have both conformance (either complete or partial) and contradiction with the laws of software evolution. In Table IV we provide a comprehensive summary of these studies. We believe this would provide a holistic view on the suitability of these laws in OSS domain, and will create the future pathway in deriving evolutionary patterns and laws for OSS evolution.

Other empirical analysis on OSS projects reveal several stimulating properties/characteristics of the system evolution. In Table V we summarizes these findings according to their primary focus of study.

*RQ10. What contributions are made in literature to analyze the evolution of organization or community?*

OSS projects typically come with a highly distributed community of developers and users. Members of such community share a common interest in the project. They often interact with one another to share knowledge, and collaborate in contributing and developing the project [50]. Communities are the core of OSS projects, and for the successful evolution of a project, it should have a large number of developers (authors and contributors) [51]. Other studies to date, report distinct properties of OSS communities. Such as, necessity of a critical mass of the core developer team [52], motivation of joining a project [50], change of roles of the community members through contributions [53], social dynamics and pattern of the community [54] [55]. These results are summarized according to the study focus in Table VI.

*RQ11. What contributions are made in literature to analyze the interdependency of the software and organization evolution?*

The successful evolution of OSS projects depends on the co-evolution of the community (both developer and user) and the software [5]. Related research in this track identifies that the number of contributors grows with the growth in product size [30]. This observation is replicated in [65], with evidence that the increase of introduced packages and reported bugs are highly coherent with the increase of contributors and active users, respectively. Also, the increase in documentation and modularization

TABLE IV.
FITNESS OF LEHMAN'S LAW IN OSS PROJECTS

| Ref | Growth Rate | Brief Description |
|---|---|---|
| [3] | Super-linear growth. | Super-linear growth pattern in system level is due to accumulated linear growth of the large set of driver subsystems. Contradict with Lehman's 4th law. |
| [22] | Super-linear growth. | The project has doubled in size in terms of SLOC and number of packages in every 2 years. |
| [23] | Super-linear growth. | Contradicts Lehman's 1st and 3rd law. |
| [24] | Linear or Super-linear growth. | Majority of the large projects grow linearly with few having a super linear growth. Both contradict with Lehman's law. |
| [25] | Average growth in size is 17%, while decreasing in structural complexity on average 13%. | Contradicts Lehman's second law. |
| [26] | Super-linear growth. | The growth is due to increase in time interval between subsequent releases in recent past. |
| [27] | Super-linear growth (for large project). Linear or Sub-linear growth (for small project). | Large projects are large in size (in LOC), more active in number of revisions, and have more programmers than those are not. |
| [28] | Linear and Super-linear growth. | Linear pattern was noted for NOF, NOC, LOC and size measures. Super-linear growth for plug-ins and downloadable source code. |
| [29] [30] [2] [31] | Linear growth. | Conformance to all six laws of Lehman considering the growth in size, coupling and complexity. |
| [4] | Linear growth. | • Conformance to Lehman's 6th and 2nd law (growth pattern measured in source code level, file level, module level and complexity).<br>• Conformance to Lehman's 1st and 5th law holds (according to changing rate and growing rate of handled uncommented line of code).<br>• Contradicts with Lehman's 6th law (for file level growth). |
| [3] | Linear or Sub-linear growth. | 16 out of 18 studied systems follow a growth pattern linear or close to linear. |
| [32] [33] | Not mentioned. | Conformance to Lehman's 1st, 2nd, 3rd and 6th law. |
| [32] [33] | Not mentioned. | Contradicts with 4th,5th,7th and 8th law. |
| [34] | Linear growth. | A declining at linear rate was noticed at module level. |
| [35] | Linear growth. | Follows a pattern of stagnated growth with decrease in size at some points. |
| [36] | Linear growth. | • The evolution pattern do not consistently conform to Lehmans laws.<br>• Decrease in structural complexity (e.g., McCabe cyclomatic complexity).<br>• Increase in calculation logic (e.g., Halstead complexity increased).<br>• Increase in modularity. |

levels are obliged to rising number of developers and their contribution in a project [59]. In this regard, more presence of users in the community drives more changes in the code base.

## IV. AVENUE TO FUTURE RESEARCH

The final step of the survey (see Figure 1) consists of formalizing the tacit knowledge acquired through the study of the review articles in order to distill further research directions. To conduct this step, we analyzed the results reported in section III to identify gaps, commonalities and contradictions, and look for most and least frequently used research approach in each facet.

### *ON SOFTWARE EVOLUTION*

Understanding the most common study facets (as displayed in Figure 2) gives the impression that *open source software evolution* is the most widely investigated field. Research under this facet has produced good sample of analytical results which are available for further examination, assessment and comparison [5]. Our review on this direction suggests following research directions,

**Law's of evolution for Open Source Software.** The most common study topic under this facet is to evaluate the fitness of Lehman's law of evolution to open source software. These results are summarized in Table IV. A closer scrutinize to the table data reveals that the results have both facsimile and contradicting to the Lehman's law. For instance, the growth rate of OSS, presented in column 2 of Table IV varies between super-linear (i.e., greater than linear) and sub-linear (i.e., less than linear). This has both conformance and contradiction with the second and sixth law of evolution.

Comprehension of these results suggest that the laws and theory appear to be breaking down through non-conforming data and findings (Table IV). Thus Lehman's laws of software evolution which is primarily based on the

TABLE V.
EVOLUTION OF THE SOFTWARE

| Focus of Study | Results | Reference |
|---|---|---|
| Code Complexity Evolution | Project size increases with an improvement on some quality measures, e.g. decrease in complexity of the system. | [37] |
| | The procedure and file level complexity remains unchanged. | [38] |
| | • Decline trend in structure complexity (e.g., McCabe cyclomatic complexity).<br>• Increase in calculation logic (e.g., Halstead complexity).<br>• Increasing trend in system's modularity. | [39] |
| | Increase in coupling, interface complexity and cyclomatic complexity. | [2] |
| | System level decay in terms of the underlying structure due to the addition of new folders and functionalities. | [40] |
| | Increase in complexity for large files as they undergo frequent changes. | [41] |
| | Decrease in modularity due to major architectural and implementation changes. | [42] |
| | There exists positive correlation between error probability across classes and the code bad smells. | [43] |
| Code Quality Improvement | Modularity of the software system increase due to periodic refactoring and cleanup activities after major changes. | [42] |
| | Periodical refactoring is needed to prevent system decay and stagnation, and to improve architectural quality. | [41] [43] |
| | The development strategies and practices of OSS projects support to maintain the reliability and quality of the software. | [44] |
| | OSS is less entropic than proprietary applications having low unit maintenance costs. | [45] |
| Code cloning | Code cloning (or code duplication) does not have a large impact on post-release defects (quality). | [46] |
| | Presence of duplicate code in the software does not make it more difficult to maintain. | [47] |
| Documentation | The documentation process often start with an initial upfront maintenance effort (to create the initial documentation or writing a book), which is then updated according to the changes in the project. | [45] |
| Sub-project Evolution (projects under a project) | • Existing sub-projects might be merged or removed.<br>• New sub-projects might be introduced.<br>• Sub-projects might follow different trend models in the growth, complexity and changes. E.g., Eclipse. | [28] |
| OSS Dynamics | • SOC (Self Organized Criticality) occurs during the evolution of OSS.<br>• SOC can be used as a conceptual framework for understanding OSS evolution dynamics. | [48] |
| | • The evolution dynamics of OSS may not follow SOC.<br>• The past of an OSS project does not determine its future except for relatively short periods of time. | [49] |

study of the large close source systems, is not sufficient to justify or account for the evolutionary pattern and behavior of the open source software. As none-the-less these laws did not consider the community dimension of the OSS projects which is an integral part of sustainable evolution of the open source software.

To deal with this problem, a viable route would be to examine the underlying ontologies for software evolution [5] considering the OSS specific characteristics, and then re-assess the laws of software evolution to fit in OSS domain.

**Metric set for software evolution**. Software evolution studies mostly utilize metrics that are empirically validated in prior studies (as presented in Table III). These metrics are derived for closed source projects, and are primarily used to verify the Lehman's law of software evolution. Though these metrics provide valuable insight to OSS evolution, they do not consider the community dynamics. Thus an empirically validated set of metrics in favor of explicit representation of the community is required to complement the existing metric set.

**Predicting the future.** Prediction of OSS projects is one area that is least popular among the study facets (Figure 2). Yet future research should focus on developing reliable prediction models and methods supporting error prediction, measuring maintenance effort and cost of OSS projects. Because, the commercial organizations, for instance, requires such prediction models to assess an open source component for adoption [66].

**Study the existence of SOC.** Another direction of research would be to study the notion of SOC (Self Organized Criticality) in OSS projects. SOC dynamics articulate that the current state of a project is determined (or at least, heavily influenced) by events that took place long time ago. Existential exploration of SOC in the domain of OSS projects reveals contradictory results (Table V). Thus future research can take further step in validating the existence of SOC and its implication on the evolution of open source software.

TABLE VI.
ORGANIZATIONAL (OR COMMUNITY) EVOLUTION

| Focus of Study | Contribution | Reference |
|---|---|---|
| Community Formation | A group of core developers quickly emerges at the center of the community at the early stage of the project and becomes the key contributors. | [54] |
| | The OSS community evolution follows small world property with a strong community structure and modularity. Initially it has fairly dynamic nature which gradually settles down into fixed groups. | [56] |
| | During the evolution, the communication between core and peripheral developer's decreases, and sub-community of developers forms. | [57] |
| | The growth of OSS community follows "rich gets richer" phenomenon. This means a healthy sized community often attracts new developers. | [58] |
| Community Structure and Activity | • 57% of the studied projects has only one or two developers.<br>• Only 15% has more than 10 developers. This category constitutes only flagship projects. | [51] |
| | • About 83% projects have only one or two stable developers.<br>• 16% of the projects attains the size of the core team. | [59] |
| | Developers play different role ranging from core developers to passive users. These roles are implicit, and are mainly defined and determined according to the level of contributions made to the project. | [53] |
| | Developer's role changes through accumulated contributions over a period of time. Changes in developer roles and community structure are significantly associated with the quality of contributions, but not with contribution quantity. | [53] [57] |
| | A Pareto distribution on the size of the developer community was identified. That is a vast majority of the OSS projects fail to take off and soon become abandoned. Probable reasons for such failure include projects inability to attract developers to attain a critical mass of developers, and insufficient communication and collaboration. | [60] [54] |
| Community Migration | • The migration of old developers from previous release to a new one is very high.<br>• Developer's code maintenance activity increases with increase in experience.<br>• The community has a natural 'regeneration' process for its voluntary contributors. This process increases the probability of code adoption that are left by the outgoing developers. | [32] |
| | Core developers should promote community regeneration process by creating an organizational ecosystem. The ecosystem will provide the openness of the system, process and communication which will attract others to join. | [61] |
| Sustainability | • OSS development is prominently a community-based model.<br>• The community must be a sustainable community for the long term survivability of the project. | [62] [50] |
| | • A stable and healthy core team in the community is essential for the sustainability of OSS projects.<br>• Core developers introduce less structural complexity and remove existing structural complexity from the code. | [2] |
| | Sustainability can be credited to the following motivating factors: (a) people are benefited from participating in a thriving OSS community, (b) improve technical competence as a developer, and (c) projects have well-defined modular design and cheap means of communication. | [63] |
| | It is important to maintain a balance composition of all the different roles in a community for sustainability. For example, to an extreme, if most of the community members are passive users then the system will not evolve. | [64] |
| Evolution of Sub-communities (Communities of sub-projects) | • Ecology of the sub-communities are formed around the sub-projects.<br>• Sub-projects are often governed by a common governance, e.g. Apache.<br>• Members in sub-communities often collaborate due to mutual task dependencies.<br>• Sub-communities also compete for the project resources. | [50] |

*ON COMMUNITY EVOLUTION*

Study on the community evolution identifies several key properties (reported in Table VI), which lay the foundation for further research in this direction. We propose the followings to be investigated.

**Community building**. Studies reported that the majority of OSS projects failed to attract members to attain the critical mass. Only few flagship projects are able to attract developers. Factors influencing the motivation to join a community has been studied (e.g., [62] [50]), and several phenomena are proposed. For instance, rich gets richer phenomenon. Yet it is not identified what exclusive properties initiate the community building process at the nebula stage of the project. Following research questions can be considered relevant,

- Why some projects are able to attract contributors during the nebula stage of the project, while most of them can not?
- What formation of the community refers to a balance one, and how the community structure changes towards a balance structure during its evolution?

- Can a visible pattern be identified within the domain of OSS projects for the above two cases?

**Migration of responsibility and sustainability**. It has been reported that migration of developers from one release to the next is high and that the developers take more responsibility as they gain experience. Yet it is a common phenomenon in open source domain that developers freely join or leave the project. And when a developer leaves, his responsibilities must be assigned to someone else. For instance, the codebase maintained by a outgoing developer should be taken care of by others. Else it will be abandoned and discarded from subsequent releases. Thus it will be beneficial to explore the followings,

- How responsibility migrates among the developers? Does this migration follow preferential-attachment?, i.e., is the responsibility handed over to the developers who are in close connection to the outgoing developer.
- What impact such migration has on the project evolution?

*ON CO-EVOLUTION*

It is turned out from our review that the understanding of co-evolution of the code and the community in OSS projects has received little attention in literature (Figure 2). As a consequence, the community dimension and corresponding communication channels (e.g., mailing archives, bug tracking systems) are explored seldom, as can be seen from Figure 5 and Figure 6 respectively. Study on co-evolution in OSS projects, however, is becoming increasingly popular. Because, in such projects the code evolution is dependent on the contribution of community members, and that a successful evolution of the code is required for the survival of the community. The following research directions can be considered relevant.

**Exploring socio-technical congruence**. In the OSS projects contributions made by the community members not only drive the system evolution but also redefine the role of these contributing members and thus change the social dynamics of the OSS community [53]. In this connection, it will be very interesting to investigate the phenomenon *socio-technical congruence* in OSS projects. Socio-technical congruence which is a conceptualization of Conway's law [67] states that there should exists a match between the coordination needs established by the technical domain (i.e., the architectural dependency in the software) and the actual coordination activities carried out by project members (i.e., within the members of the development team) [67]. This concept was already explored in closed source projects, and reported a high correlation with software build success, quality, and faster rate of modification [68]. Thus socio-technical congruence plays a pivotal role in conceptualizing the co-evolution in a project. Surprisingly, this notion as a research area has not been given much attention among open source researchers. Although it is identified and reported as a desired property for collaborative development activities like OSS projects [69]. Considering the lack of focus in this direction, we propose the following to investigate.

- Does the essence of socio-technical congruence as a conceptualization of Conway's law holds for OSS project? Can it be stated as an implicit characteristics or property of successful OSS project?
- What quantitative approach/method can be utilized to verify the existence of socio-technical congruence in OSS projects? What repositories can be used for this purpose?
- What correlation can be derived between socio-technical congruence and the quality/sustainability of OSS projects?

**Sub-project evolution with their community**. Large open source projects often encompass many sub-projects. Such as, sub-projects in Eclipse, GNU, Linux, and Apache. Often ecology of sub-communities formed around these sub-projects, which are governed by a common governance [50]. Study on the formation and evolution of sub-projects and their communities have revealed many key characteristics, which are listed in Table V and Table VI, respectively. Yet the interdependency in evolution between the two and their impact on the overall project evolution remain untouched. The following would be worth to investigate.

- Does there exist a correlation between the evolution (growth, complexity, change) of the sub-projects and their associated sub-communities? Does the community change with the change in the sub-project?
- How does a community form around a newly added sub-project?
- What attributes of a sub-project attract new developers to join?
- What happens to the sub-community when a sub-project is deleted or merged to other sub-project?
- What dependencies lead to inter project communication?
- What kind and level of communication and collaboration takes place between sub-communities?
- Does there exist a correlation between the project evolution and the sub-project evolution?

*ON RESEARCH METHOD*

A number of issues related to the research approach can be improved to increase the acceptability of the reported results. We pointed out the followings,

**External validity of the results**. Empirical study is the most popular research approach employed in evolution studies (Figure 4). These studies, however, are horizontal in nature (as reported in RQ5) considering only flagship OSS projects. Due to this approach of studying OSS projects, the reported results suffer from

generalizability threat, as reported in Figure 7. Yet to make these finding applicable and hold for the extended region of OSS projects, explicit measure should be taken. An interesting route to deal with this is to categorize the findings (current or future) according to the project domain, or similar organizational structure and practices, or similar product size and complexity. This will reveal the broader picture which can then be compared and possibly merged for proposing a more general evolutionary pattern and behavior for OSS.

**Framework for the data collection and representation**. As discussed in RQ6, OSS projects often produce large volume of data representing their development and evolution history. Research to date, explores the repositories that maintain these data, a list of which is provided in Figure 6. However, data collection and representation in these repositories vary significantly from project to project. Furthermore, data from the same source may have different formatting (e.g., emails are often free of format even in listing the senders credentials). Due to these facts, it is a challenging task to collect relevant data following a standard format from OSS repositories. In this context, researchers often employ their own means to collect and represent data for research. This reduces the compatibility and comparability of the reported results even if they use same data sources. Taking these issues in consideration, a framework for uniform data collection and representation can be developed to make the results cohesive and comparable to each other.

## V. Threats to Validity

Carrying out a survey is mostly a manual task. Thus most threats to validity relate to the possibility of researcher bias [9]. To minimize this, we adopted guidelines on conducting SLR suggested by Kitchenham [13]. In particular, we documented and reviewed all steps we made in advance, including selection criteria and attribute definitions.

In what follows, the description related to validity threats pertaining to the article selection, the attribute framework, and the article characterization is discussed.

### A. Article Selection

Following the advice of Kitchenham [13], the inclusion criteria is set at the time of defining the review protocol, and the criteria are based on the research questions. This reduces the likelihood of bias. Articles satisfying this selection criterion are considered. For collecting relevant articles we first performed automated keyword search and then performed manual selection. The first step condenses the selection bias whereas the latter ensures the relevance of the selected articles. Finally, a non recursive search through the references of the selected articles is performed. This increases the representativeness and completeness of our selection. To further minimize the selection bias and reviewer bias, domain experts (second and third author) verified the relevance of the selected articles against the selection criteria.

### B. Attribute Framework

The construction of the attribute framework may be the most subjective step [9]. Thus we take the following steps to acknowledge this fact: the attribute set is derived based on the research questions and domain of study. Then a pilot study is carried out to further refine the attribute framework. Furthermore, the representativeness of the framework is examined by domain experts (second and third author).

### C. Article Assessment

Similar to the construction of the attribute framework, the process of assigning the attributes to the research articles is subjective and may be difficult to reproduce [9]. We address this validation threat through an evaluation process where domain experts assess the collected data against reviewed articles.

## VI. Discussion

In this paper we have reported a systematic literature review (SLR) on the evolution studies of Open Source Software projects. To carry out this study we adopted a review protocol following the guidelines presented in [13] and [9]. A set of 101 articles (21 journal and 80 conference articles) were selected for the review. Through a detailed reading of a subset of the selected articles, we derived an attribute framework that was consequently used to characterize the articles in a structured fashion.

We also posed a set of research questions in advance that are investigated and answered throughout the study. The attribute framework was sufficiently specific to characterize the articles in answering the research questions. The set of articles and collected data under this attribute framework is presented in our review website [14]. Nonetheless, an elaborated discussion on the validity of the review process is also presented.

The characterization of the reviewed articles will help researchers to investigate previous studies from the perspective of metrics, methods, datasets, tool sets, and performance evaluation and validation techniques in an effective and efficient manner. We also put an elaborated discussion on the most significant research results. In summary, this article provides a single point reference on the state-of-the-art of OSS evolution studies which could benefit the research community to establish future research in the field.

Related works in this track carried out a literature review on open source software evolution [70]. This study explores the software evolution, mostly emphasizing on research methods, metrics, and data analysis. Contrast to this, our review provides a holistic view of the evolution of OSS projects, concerning all the facets studied to date.

Yet our reported result pertaining to the conflicting reporting of Lehman's law of software evolution confirmed the findings in [70]. This suggests that future work in the area of OSS evolution should explore more to unify the findings through comprehensive study on the the open areas discussed in this paper.

## REFERENCES


[1] R. Grewal, G. Lilien, and G. Mallapragada, "Location, location, location: How network embeddedness affects project success in open source systems," in *Management Science*, vol. 52, no. 7, 2006, pp. 1043–1056.

[2] S. Suh and I. Neamtiu, "Studying software evolution for taming software complexity," in *Australian Software Engineering Conference*, 2010, pp. 3–12.

[3] G. Robles, J. Amor, J. Gonzalez-Barahona, and I. Herraiz, "Evolution and growth in large libre software projects," in *IWPSE'05*, 2005, pp. 165–174.

[4] C. Roy and J. Cordy, "Evaluating the evolution of small scale open source software systems," in *CIC 2006*, 2006, pp. 123–136.

[5] W. Scacchi, "Understanding open source software evolution: Applying, breaking, and rethinking the laws of software evolution," in *Applying, Breaking, and Rethinking the Laws of Software Evolution*. John Wiley and Sons Inc, 2003.

[6] B. A. Kitchenham, R. Pretorius, D. Budgen, O. P. Brereton, M. Turner, M. Niazi, and S. Linkman, "Systematic literature reviews in software engineering- a tertiary study," *IST*, vol. 52, no. 8, pp. 792–805, 2010.

[7] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software," in *Engineering Technical Report EBSE-2007-01*, 2007.

[8] M. Petticrew and H. Roberts, "Systematic reviews in the social sciences: A practical guide," in *Blackwell Publishing*, 2005.

[9] B. Cornelissen, A. Zaidman, A. Deursen, L. Moonen, and R. Koschke, "A systematic survey of program comprehension through dynamic analysis," *TSE*, vol. 35, no. 5, pp. 684–702, 2009.

[10] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Systems with Applications*, vol. 36, no. 4, pp. 7346–7354, 2009.

[11] D. Łmite, C. Wohlin, T. Gorschek, and R. Feldt, "Empirical evidence in global software engineering: a systematic review," *ESE*, vol. 15, no. 1, pp. 91–118, 2010.

[12] A. Pourshahid, D. Amyot, A. Shamsaei, G. Mussbacher, and M. Weiss, "A systematic review and assessment of aspect-oriented methods applied to business process adaptation," *JSW*, vol. 7, no. 8, pp. 1816–1826, 2012.

[13] B. A. Kitchenham, "Procedures for performing systematic reviews," in *Technical Report TR/SE-0401, Keele University, and Technical Report 0400011T.1, National ICT Australia*, 2004.

[14] M. M. Syeed, "http://reviewossevolution.weebly.com/," 2013.

[15] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *JSS*, vol. 80, no. 4, pp. 571–583, 2007.

[16] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp, "Motivation in software engineering: A systematic literature review," *IST*, vol. 50, no. 9-10, pp. 860–878, 2008.

[17] T. Dyba and T. Dingsyr, "Empirical studies of agile software development: A systematic review," *IST*, vol. 50, no. 9-10, pp. 833–859, 2008.

[18] D. Atkins, T. Ball, T. Graves, and A. Mockus, "Using version control data to evaluate the impact of software tools," in *ICSE*, 1999, p. 324333.

[19] C. B. K. Beecher, A. Capiluppi, "Identifying exogenous drivers and evolutionary stages in floss projects," *The Journal of Systems and Software*, vol. 82, no. 5, pp. 739–750, 2009.

[20] J. Cook, L. Votta, and A. Wolf, "Cost-effective analysis of in-place software processes," *TSE*, vol. 24, no. 8, p. 650663, 1998.

[21] D. Perry, A. Porter, and L. Votta, "Empirical studies of software engineering: A roadmap," in *The Future of Software Engineering, Finkelstein A (ed.). ACM Press: New York NY*, 2000.

[22] J. Gonzalez-Barahona, G. Robles, M. Michlmayr, J. Amor, and D. German, "Macro-level software evolution: a case study of a large software compilation," *Journal Empirical Software Engineering*, vol. 14, no. 3, pp. 262–285, 2009.

[23] M. Godfrey and Q. Tu, "Evolution in open source software: A case study," in *ICSM*, 2000, pp. 131–142.

[24] A. Capiluppi, J. Gonzlez-Barahona, I. Herraiz, and G. Robles, "Adapting the staged model for software evolution to free/libre/open source software," in *IWPSE '07*, 2007, pp. 79–82.

[25] D. Darcy, S. Daniel, and K. Stewart, "Exploring complexity in open source software: Evolutionary patterns, antecedents, and outcomes," in *HICSS '10*, 2010, pp. 1–11.

[26] G. Robles, J. M. Gonzalez-Barahona, M. Michlmayr, and J. Amor, "Mining large software compilations over time: Another perspective of software evolution," in *MSR '06*, 2006, pp. 3–9.

[27] K. Stefan, "Software evolution in open source projectsa large-scale investigation," *Journal of Software Maintainance and Evolution: Research and Practice*, vol. 19, pp. 361–382, 2007.

[28] T. Mens, J. Fernndez-Ramil, and S. Degrandsart, "The evolution of eclipse," in *International Conference on Software Maintenance (ICSM)*, 2008, pp. 386–395.

[29] A. Bauer and M. Pizka, "The contribution of free software to software evolution," in *Sixth International Workshop on Principles of Software Evolution*, 2003, pp. 170–179.

[30] A. Capiluppi, "Models for the evolution of os projects," in *ICSM '03*, 2003, pp. 65–74.

[31] S. McIntosh, B. Adams, and A. Hassan, "The evolution of ant build systems," in *MSR'10*, 2010, pp. 42–51.

[32] Y. Lee, J. Yang, and K. Chang, "Metrics and evolution in open source software," in *QSIC'07*, 2007, pp. 191–197.

[33] G. Xie, J. Chen, and I. Neamtiu, "Towards a better understanding of software evolution: An empirical study on open source software," in *ICSM'09*, 2009, pp. 51–60.

[34] S. Ali and O. Maqbool, "Monitoring software evolution using multiple types of changes," in *ICET'09*, 2009, pp. 410–415.

[35] A. Capiluppi and J. Ramil, "Studying the evolution of open source systems at different levels of granularity: Two case studies," in *IWPSE*, 2004, pp. 113–118.

[36] M. M. Simmons, P. Vercellone-Smith, and P. Laplante, "Understanding open source software through software archeology: The case of nethack," in *30th SEW*, 2006, pp. 47–58.

[37] K. Stewart, D. Darcy, and S. Daniel, "Observations on patterns of development in open source software projects," in *5th WOSSE*, 2005, pp. 1–5.

[38] A. Capiluppi and J. Ramil, "Studying the evolution of open source systems at different levels of granularity: Two case studies," in *IWPSE'04*, 2004, pp. 113–118.

[39] M. Simmons, P. Vercellone-Smith, and P. Laplante, "Understanding open source software through software archeology: The case of nethack," in *SEW '06*, 2006, pp. 47–58.

[40] A. Capiluppi and T. Knowles, "Software engineering in practice: Design and architectures of floss systems," in *Open Source Ecosystems: Diverse Communities Interacting, IFIP Advances in Information and Communication Technology*, vol. 299/2009, 2009, pp. 34–46.
[41] A. Capiluppi and J. Ramil, "Change rate and complexity in software evolutions," in *WESS'04*, 2004.
[42] R. Milev, S. Muegge, and M. Weiss, "Design evolution of an open source project using an improved modularity metric," in *OSS'09*, 2009, pp. 20–33.
[43] W. Li and R. Shatnawi, "An empirical study of the bad smells and class error probability in the post-release object-oriented system evolution," *Journal of Systems and Software*, vol. 80, no. 7, pp. 1120–1128, 2007.
[44] K. Hayashi, K. Kishida, K. Nakakoji, Y. Nishinaka, B. Reeves, A. Takasbima, and Y. Yamamoto, "A case study of the evolution of jun: an object-oriented open-source 3d multimedia library," in *ICSE'01*, 2001, pp. 524–533.
[45] B. Dagenais and M. Robillard, "Creating and evolving developer documentation: understanding the decisions of open source contributors," in *FSE'10*, 2010, pp. 127–136.
[46] N. Bettenburg, W. Shang, W. Ibrahim, B. Adams, Y. Zou, and A. Hassan, "An empirical study on inconsistent changes to code clones at the release level," in *WCRE '09*, 2009, pp. 85–94.
[47] K. Hotta, Y. Sano, Y. Higo, and S. Kusumoto, "Is duplicate code more frequently modified than non-duplicate code in software evolution?: An empirical study on open source software," in *IWPSE-EVOL '10*, 2010, pp. 73–82.
[48] W. Jingwei, R. Holt, and A. Hassan, "Empirical evidence for soc dynamics in software evolution," in *IEEE International Conference on Software Maintenance (ICSM 2007)*, 2007, pp. 244–254.
[49] I. Herraiz, J. Barahona, and G. Robles, "Determinism and evolution," in *Proceedings of the 2008 international working conference on Mining software repositories (MSR'08)*, 2008, pp. 1–10.
[50] M. Weiss, G. Moroiu, and P. Zhao, "Evolution of open source communities," *OSS*, vol. 203, pp. 21–32, 2006.
[51] A. Capiluppi, P.Lago, and M. Morisio, "Evidences in the evolution of os projects through changelog analysis," in *Proceedings of the 3rd Workshop on Open Source Software Engineering (ICSE03)*, 2003, pp. 19–24.
[52] A. mockus, R. Fielding, and J. herbsleb, "Two case studies of open source software development: Apache and mozilla," *ACM Trans. Software Engineering and Methodology*, vol. 11, no. 3, pp. 309–346, 2002.
[53] K. Nakakoji, Y. Yamamoto, Y. Nishinaka, K. Kishida, and Y. Ye, "Evolution patterns of open-source software systems and communities," in *IWPSE*, 2002, pp. 76–85.
[54] K. Ngamkajornwiwat, D. Zhang, A. Koru, L. Zhou, and R. Nolker, "An exploratory study on the evolution of oss developer communities," in *HICSS*, 2008, p. 305.
[55] Q. Hong, S. Kim, S. Cheung, and C. Bird, "Understanding a developer social network and its evolution," in *27th ICSM*, 2011, pp. 323–332.
[56] Q. Hong, S. Kim, S. Cheung, and C.Bird, "Understanding a developer social network and its evolution," in *27th ICSM*, 2011, pp. 323–332.
[57] R. Chang, S. Yang, J. Moon, W. Oh, and A. Pinsonneault, "A social capital perspective of participant contribution in open source communities: The case of linux," in *HICSS*, 2011, pp. 1–10.
[58] M. Weiss, G. Moroiu, and P. Zhao, "Evolution of open source communities," *OSS*, vol. 203, pp. 21–32, 2006.
[59] A. Capiluppi, P. Lago, and M. Morisio, "Characteristics of open source projects," in *CSMR '03*, 2003, p. 317.
[60] F. Hunt and P. Johnson, "On the pareto distribution of sourceforge projects," in *Proceedings of Open Source Software Development workshop*, 2002, pp. 122–129.
[61] Y. Yunwen and K. Kishida, "Toward an understanding of the motivation open source software developers," in *ICSE '03*, 2003, pp. 419–429.
[62] S. Shah, "Motivation, governance, and the viability of hybrid forms in open source software development," in *Management Science*, vol. 52, pp. 1000–1014.
[63] J. Gutsche, "The evolution of open source communities," *Topics in Economic Analysis and Policy*, vol. 5, no. 1, 2005.
[64] G. Robles, J. M. Gonzalez-Barahona, and M. Michlmayr, "Evolution of volunteer participation in libre software projects: Evidence from debian," in *1st OSS*, 2005, pp. 100–107.
[65] Y. Wang, D. Guo, and H. Shi, "Measuring the evolution of open source software with their communities," *ACM SIGSOFT Software Engineering Notes*, vol. 32, no. 6, pp. 1–7, 2007.
[66] M. Syeed, T. Kilamo, I. Hammouda, and T. Systa, "Open source prediction methods: a systematic literature review," in *Proceedings of 8th. OSS, Springer*, 2012.
[67] S. Bendifallah and W. Scacchi, "Work structures and shifts: An empirical analysis of software specification teamwork," in *11th ICSE*, 1989, p. 260270.
[68] N. Nagappan, B. Murphy, and V. Basili, "The influence of organizational structure on software quality: an empirical case study," in *ICSE*, 2008, p. 521530.
[69] T. Browning, "Applying the design structure matrix to system decomposition and integration problems: a review and new directions," in *IEEE Transactions on Engineering Management*, vol. 43, no. 3, 2001, p. 292306.
[70] H. Breivold, M. Chauhan, and M. Babar, "A systematic review of studies of open source software evolution," in *APSEC*, 2010, pp. 356–365.

**M.M. Mahbubul Syeed** received his B.Sc degree in Computer Science and Information Technology from Islamic University of Technology, Bangladesh in September, 2002 and his M.Sc degree in Information Technology from Tampere University of Technology, Finland in April, 2010. He is currently working towards his Ph.D. degree and working as a researcher in the same university. His current research interest includes study of Open Source Software ecosystem.

**Imed Hammouda** is currently an associate professor at Tampere University of Technology (TUT) where he is heading the international masters programme at the Department of Pervasive Computing. He got his Ph.D. in software engineering from TUT in 2005. Dr. Hammouda's research interests include open source software, software architecture, software development methods and tools, and variability management. He is leading TUTOpen - TUT research group on open source software. He has been the principal investigator of several research projects on various open initiatives. Dr. Hammouda's publication record includes over fifty journal and conference papers.

**Tarja Systä** is a professor at Tampere University of Technology, of Pervasive Computing Department. Her current research interests include software maintenance and analysis, software architectures, model-driven software development, and development and management of service-oriented systems.

## APPENDIX

| Application domain and usage of the Tools | Tools |
|---|---|
| **Content analysis software**<br>- Parse the source code and extract the dependencies between program entities<br>- Measure the abstractness and instability of a system's packages<br>- Recover changes from the CVS repository<br>- Code clone detection<br>- Measuring code complexity<br>- Test coverage measurement<br>- File comparison<br>- Inheritance and attribute data analysis<br>- Source code analysis, such as, measuring total size, LOC, no of uncommented LOC, number of global functions, variables and macros. | Word count (LIWC), SLOCCount, Numlines, Ctags, CVSchangelog, D-CCFinder, Diff, SVNKit library, JDEpend4Eclipse plugin, Jdepend, PBS tools, cvs2cl, SimScan, CloneDR, Scientific Toolworks' Understand, Emma, JarJarDiff, Jhawk, codeSurfer, RSM , Analizo, analizo-utils package, softChange, CIL merger tool, Linguistic Inquiry |
| **Data detection tool**<br>- Token-based, line-based, PDG-based<br>- Source code extraction (from source code repositories, mailing lists and bug trackers) | CCFinder, CCFinderX, Simian, Scorpio, exuberant ctags<br>Stripcmt, JavaCC parser generator, Utilities, Kenyon, Doxygen, LDX, CodeVizard, ASTdiff, CTSX, CVSAnaly2, MIstats, Bicho, Weka |
| **Metric extraction tool**<br>- OO metrics, Complexiety metics, product and project metrics | Jhawk tool, STAN, Metrics, Borland Together tool<br>Understand for C |
| **Simulation tool** | NetLogo |
| **Social network analysis** | Ucinet, Louvain algorithm, OSSNetwork |
| **Statistical tool** | SPSS, R (free statistical package) |
| **Visualization tool** | Herdsman, Least-Squares Fitter (LSF), DOT |

Figure 8. Tools (OSS and Proprietary) used for evolution studies

| OSS Domain | Domain wise Study frequency | OSS Projects ( Study Frequency) |
|---|---|---|
| 2D Java game engine | 1 | EasyWay (1) |
| 2D graphics drawing | 2 | JHotDraw (2) |
| 3D modeling | 1 | ThreeCAM (1) |
| Application software | 21 | Mplayer(1), Gimp(3), gnumeric(1), Barcode Library (1), Zlib(1), Gnuparted(1), Weasel(1), Dailystrips (1), Edna(1), Motion(1), Rblcheck (1), Xautolock(1), Bubblemon (1), Disc-Cover(1), FOP(1), Freenet(1), Jetspeed2(1), Jmol(1), TV-Browser(1) |
| Application suite | 1 | Pentaho (1) |
| Bittorrent clinet | 1 | Azureus (1) |
| Bug Tracking System | 1 | Mantis (1) |
| Business intelligent and reporting engine | 2 | JasperReports (2) |
| Code standard checker | 1 | Checkstyle (1) |
| Compiler | 3 | GCC (2), Jasmin (1) |
| Database engine | 4 | DatabaseToUML(1), QMailAdmin (1), SQLite (2) |
| Desktop Enviornment | 9 | GNOME(4), KDE(4), mono (1) |
| Distributed File system | 5 | Arla (5) |
| Document viewer | 2 | Evince (2) |
| File server (MP3) | 1 | Mutt (1) |
| FLOSS repositories | 4 | RubyForge(1), Savannah(1),SourceForge(2) |
| Framework | 9 | OsCache(1), Spring Framework(3), Hibernate(2), Shoes(1), VLC(1), Django(1) |
| FTP client | 2 | FileZilla (2) |
| Game | 5 | Tyrant(2), FreeCol (1), Nethack(1), GameScanner(1) |
| GIS | 1 | Grass (1) |
| Http Client | 1 | jakarta-commons (1) |
| IDE | 14 | Kdevelop(1), Squeal(1), Eclipse(9), Swig(1), JDT core(2) |
| Instant Masseging (IM) | 7 | Gaim(1), OpenYMSG (1), aMSN(1), Miranda(2), Ayttm(1), Pidgin(1) |
| Internet and networking | 5 | Kdenetwork(1), NatMonitor(1), Tritonn(1), Newsstar(1), Hamachi-GUI (1) |
| IRC client | 1 | Konversation (1) |
| Java Application Generator | 1 | JAG (1) |
| Java's type safe nature | 1 | Guice (1) |
| Library | 14 | CDK(1), kdelibs (1), Wine(2), DBI(1), SwingWT(1), JFreeChart(1), Ant(5), Jun(2) |
| Mail Client | 6 | Evolution(2), Calamaris(1), Ximian Evolution(1), Columba(2) |
| Office suit | 3 | Koffice(2), OpenOffice(1) |
| OS | 33 | Linux(13), Ubuntu(1), BSD(4), kdebase(1), Debian(5), Fedora(1), FreeBSD(4), NetBSD(3), OpenBSD(1) |
| peer to peer data streaming sw | 1 | Ktorrent (1) |
| Platform (Blogging) | 1 | WordPress (1) |
| Programming language | 3 | PHP(1), Ruby(2) |
| Protocol | 2 | OpenSSH (2) |
| Relational database management | 8 | SQuirrel SQL Client(1), HSQLDB(2), PostgreSQL(4), Firebird (1) |
| SCM software | 2 | Subversion(1), CVS(1) |
| SDK | 1 | KSDK (1) |
| Server (Application) | 24 | Jboss(3), Bind(2), Vsftpd(2), SendMail(3), Apache(7), AdServerBeans(1), aolserver(1), cherokee(1), fnord(1), lighttpd(1), monkeyd(1), weborf(1). |
| Source code analyzer | 1 | PMD (1) |
| SSL implementation | 1 | OpenSSL (1) |
| Support system | 2 | GNUWingnu(1), SRA-PostgreSQ(1) |
| Testing | 1 | EclEmma(1) |
| Text editor | 6 | WinMerge(1), Jedit(4), VIM text editor(1) |
| Tool suit | 4 | Samba(2), Quagga(2) |
| UML modeling tool | 9 | ArgoUML(9) |
| Web browser | 7 | Mozilla(5), galeon(1), Mem´oria Virtual(1) |

Figure 9. OSS Projects analyzed for evolution studies