Open Source Software Hosting Platforms: A Collaborative Perspective's Review

Ghadah Alamer, Sultan Alyahya*
Information Systems Department, College of Computer & Information Sciences, King Saud University, Riyadh, Saudi Arabia

* Corresponding author. email: sualyahya@ksu.edu.sa
Manuscript submitted February 25, 2017; accepted April 27, 2017.
doi: 10.17706/jsw.12.4.274-291

Abstract: Open source software (OSS) has been gaining large attention lately. In fact, several studies have shown that the collaboration is a remarkably major factor influencing the OSS quality. In addition, there are several existing platforms providing mediums for collaborative development of the OSS projects and involve advanced features that assist in boosting the collaboration process. Such platforms which can be called as open source software hosting platforms are rich of collaborative workflows. Therefore, in this research a comprehensive investigation of the current OSS hosting platforms is held for the sake of studying their collaboration capabilities as well as pointing out any limitations related to collaboration that might hinder OSS from meeting high quality. The review has shown that the current OSS hosting platforms have some potential limitations. The identified limitations have been shared through a survey with developers working on OSS platforms and the results have generally revealed that there is a necessity to overcome these limitations.

Keywords: Collaboration, open source software, platform, software development.

1. Introduction

Software development process has been moving towards community-based Open Source Software (OSS) development rather than being completely closed [1]. OSS development is facing large growth and their existence is notable in various areas such as industry, government and education [1][2]. Therefore, there are many powerful platforms have emerged providing collaborative hosting services for OSS projects (e.g. GitHub, GitLab, Bitbucket).

OSS development is characterized by distinctive characteristics which have grabbed the attention of practitioners and researchers. Having a geographically distributed and voluntary nature, and as well as a changing and evolving community surrounding the OSS are some of the major properties of OSS development. This nature of the development process has a great and direct impact on the collaboration between the developers working on the OSS projects. Therefore, developers have been and are still working on finding out creative methods to increase productivity, efficiency, quality and reduce the complication of managing OSS projects through the adoption of better collaboration practices and version control tools.

OSS is developed in an environment that potentially might introduce several obstacles in the collaboration between developers during development. Hence, having some problems in collaboration hinders open source software from achieving the required and expected level of quality. Hironori et al. in their study [3], clearly stated that: *"In OSS development, the collaboration among developers is the key to improve software quality [3]".*

According to [4], a large number of maintained OSS projects are available, however, on the other side, there are as well large number of abandoned projects with low quality. Due to the distributed, voluntary, virtual and little strictness and systematization nature of the OSS development, some collaborative concerns have been shown up. In fact, the distribution of developers makes collaboration much more challenging [5]. The OSS development is performed through the Internet with possibly no face-to-face meetings at all. Thus, unlike in the traditional development, it is quite difficult for developers to form clear and valid impressions about each other since they are collaborating virtually [6], in contrast to the closed or traditional software development where tasks are assigned to developers and they are fully in charge of them and besides, they usually are aware of each other's abilities.

In addition, having a voluntary nature where none of the collaborating developers working on the OSS is held responsible for a certain work is also another key aspect having a high impact on the improvement of the OSS [4]. Low commitments in OSS development might seem pleasing to developers; however, it might slow down the development process or even stop it.

Having such characteristics in the OSS development requires more attentiveness towards the collaborative gaps preventing OSS from achieving good quality. In this research, the focus shall be on understating OSS development through exploring and evaluating the existing OSS hosting platforms. The research endeavors to contribute to the enhancement of the current collaboration processes between developers in currently existing OSS hosting platforms.

In fact, the main motivation behind this research is that there are no studies provide a holistic investigation on OSS hosting platforms from a collaborative perspective. Despite the fact that there are some studies that select a single platform for investigation, however, it has appeared to us that no studies have been found which investigate the collaboration processes in particular. Therefore, this research attempts to contribute to fill this gap by performing a detailed investigation on a set of OSS hosting platforms by focusing mainly on their collaborative aspects.

The remainder of this paper is organized as follows: Section 2 will provide a detailed background of OSS and will discuss the related work. Section 3 then will present an elaborative evaluation performed on the collaboration features existing in the OSS hosting platforms. Afterwards, an analysis of the limitation and challenges that the platforms are experiencing is provided in Section 4 with a survey supporting the credibility and necessity of overcoming these challenges. Finally, the paper is concluded in Section 5.

2. Literature Review

2.1. An Overview of Open Source Software (OSS)

Open Source (OS) is relatively a recent term, but the basic idea behind it is actually not. The term has been around since 1960 [7] [8]. OSS refers to as software that is free to download and use and its source code can be completely accessible, viewable and modifiable by interested developers [9].

The software industry is confronting a great transformation towards OSS model rather than closed software (proprietary software) model [1] [10]. Nowadays, many software products have OSS components integrated into them [1]. In addition, more than 70% of IT professionals in the United States prefer OSS over proprietary software [10].

OSS development includes the following characteristics:

2.1.1. Based on volunteering

OSS projects have a voluntary nature, where members contribute to an OSS project voluntarily and are not paid for their contributions.

2.1.2. Collaboration over a geographically distributed environment

OSS is developed in a geographically distributed manner [7], where members of the OSS community are

dispersedly located in remote sites with different time zones, and are collaboratively working on the OSS project.

Therefore, having this distributed nature of OSS development creates the need for coordination, as Aberdour [11] have stated that: "*OSS development must also manage a geographically distributed team,* requiring *focus on coordination tasks".* According to Nakakoji *et al.* [9], collaborative development is a substantial characteristic of any OSS.

2.1.3. Developers do not stick into a strict plan or schedule

In OSS development, developers do not usually follow a well-structured plan and schedule [7]. In fact, the original developer actually starts the OSS project without planning it. However, the entire OSS community is the one which collaboratively leads the evolution and development process [9]. In contrast to OSS development, the traditional SW development process puts large effort into planning and scheduling [11]. Moreover, in the OSS development the owner of the project does not have full power over the contributing developers [7]. Therefore, developers are not being assigned strictly to specific roles and work; rather they assign themselves with what they see falls into their interests and desires [9]. In addition, OSS might not be accurately designed at first, which means that the software may not be of advanced level or of massively high qualifications [9]. However, OSS and the whole OSS community evolve over time [7] [9]. In the OSS development process the software never reaches to an end, but it keep on improving and evolving depending on the users' desires and needs [8].

2.1.4. Development done in parallel

The OSS development is a parallel process; therefore, it is quite difficult to track the whole development progress; even though Version Control Systems (VCS) assist in doing so. In contrast, the Closed Source Software Development (CSSD) can be clearly tracked and traced [8]. In fact, the OSS's source code is improved and maintained using parallel debugging which is performed by large number of distributed developers. However, in closed source development, the source code is usually maintained and quality-checked using planned and systematic testing [7].

2.1.5. Requirement elicitation is driven by the collaborative OSS community

Since the OSS development has an evolutionary nature, the requirements keep on appearing and emerging during the development. Contrary to OSS development, the closed source software development defines the set of requirements in the very beginning through requirement elicitation [8], while in OSS development, requirements elicitation is normally driven by the collaborative community of the OSS. For instance, in OSS hosting platforms such SourceForge, users can often submit feature requests for an OSS project where this can be considered as a good approach eliminating the need for requirements elicitation. Furthermore, in OSS development, requirements go into the implementation phase directly without stopping into the detailed phase of design [8].

2.2. Related Work

Collaboration is the heart of software development; several research studies have shown that nearly 70% of the development time is used in collaborative practices [12]. The software development process includes stiff integration and relationships between software modules [4], therefore, it is critical to have large amount of communication and collaboration between developers in an efficiently managed manner usually via specialized tools [2], [4], [13]. Any problems in communication between developers will negatively affect the quality of the OSS. Hironori et al. [3] have stated that: *"in open source software development, the collaboration among developers is the key to improve software quality"*

Inefficient collaboration between developers in the OSS environment will cause problems in OSS quality [3][4]. Some of the problems that might arise are the duplication of effort whether in the removal of defects

or by developing the same type of features and in some cases there might be difficulties in allocating responsibilities [4]. The OSS development environment differs from the traditional methods of development. It mostly adopts the "freedom" concept. According to [7], the project owner finds it hard to strictly assign tasks to contributors in the OSS project development, and as well has little power over them. Therefore, some studies (e.g. [4], [14]) believe that traditional development teams might work more effectively since all members collaborate conveniently without facing any problems in communication.

The authors in [3] have studied the collaboration among developers within OSS development specifically in the bug fixing process. The study as well investigates how much developers are aware of the entire OSS modules, and how frequently they need to collaborate. The study revealed that 50% of the developers are acquainted with only one or two modules which this indicates the strong requisite for collaboration when a bug-fix has an influence on several modules. Moreover, the authors have stated that collaboration is a key factor for the enhancement of the OSS quality; however, it may sacrifice quality due to some misunderstanding resulting from collaboration and subsequently leading to bug-reopening.

Another study was done by Jarczyk *et al*. [15] which investigates the relation between the OSS project quality and the properties of the development team members. The study was done on a dataset representing OSS projects from GitHub. In addition, the quality measures considered were the popularity of the project in GitHub, and how fast the team fixes the reported issues in their project. The study has revealed that having members with focused developers rather than popular ones is preferred for the OSS project quality.

Yuya *et al.* [16] have done an investigation on what affects the success of social coding sites' (SCS) projects. The study has concentrated on three aspects which are the team structure, interactions with external developers outside the team and project content. The study has utilized analytical techniques in order to evaluate the relationships between the three aspects and the success of a project in the SCS's, and the findings have shown several correlations which can be considered by developers who would start a project in one of the social coding sites.

Moreover, a qualitative study focusing on the social interactions and collaboration in GitHub was firstly conducted by Antonio *et al*. [17]. The researchers have analyzed 18 events performed by users in GiHhub such as pull requests, forks, issues…etc. on a dataset of nearly 2.19 million users and 5.68 million repositories. The study came up with important networks describing correlations between collaborative and social attributes. The authors have finally suggested that their study poses a starting point for developing innovative strategies that can assist in enhancing collaboration.

In addition, Laura *et al*. [18] have done a study on how transparency and visibility in the OSS community is critical to boost collaboration, knowledge sharing and reputation management. The study has been conducted on GitHub users, and it has been declared that users make inferences about others from the visible networked activity information. Moreover, the study has found that such information can create inferences around several areas, which assists in collaboration and community evolvement.

The previous studies have all investigated what factors impact the success and quality of the OSS projects. The factors that were studied mainly focused around collaboration, social interactions and team structure and relationships. The studies proofed that there are correlations between such factors and the success, quality and improvement of OSS. Moreover, all studies stated that collaboration is critical in OSS development.

Furthermore, most the studies have done their study on one OSS hosting platform such as GitHub [15], [17], [18]. However, in this research a thorough qualitative analysis of a set of OSS hosting platforms is performed to study the built-in collaboration features. In addition, the authors in this research will not focus on proofing that the collaboration has its impact on OSS projects since this has been shown by large number of studies, but, it shall concentrate on finding out what shortcomings related to collaboration are present in such platforms hindering OSS project from reaching the expected quality.

3. OSS Hosting Platforms

OSS hosting platforms create enormous communities which focus on developing and maintaining OSS [19]. These platforms offer a good mean enabling efficient collaboration across developers with varying technical backgrounds who are distributed across different locations [20].

There are plenty of key elements in OSS hosting platforms which need to be defined before describing the platforms' features. In every platform there are elements which are considered as inputs into the platform which are the following:

- Repository/Project. A repository involves all the source code files of a project. A project can be considered as a frame for a repository, where a repository has a dedicated project page showing all issues, discussions, history of commits and other information pertaining to this repository [18].

- **Issues.** Issues can be bugs, feature requests, enhancements and other types of reporting related to a repository [21] [22].

- **Collaborators and contributors.** Collaborators are members of the team developing the project, and normally have write and read access permissions into it [23], while contributors do not have write access to that project/repository but can for instance report issues and send pull requests and can comment and discuss their point of views with the core developers (collaborators). Moreover, the owner (author) of the project/ repository who is by default the creator of it, has full privileges in managing and accessing it, and can as well set the level of permissions for the collaborators in the project [23].

3.1. Selected OSS Hosting Platforms

A number of open source software hosting platforms were selected for evaluation and exploration. In fact, there are not many OSS hosting platforms existing.

Table 1 shows the list of OSS hosting platforms that were selected for evaluation. Platforms highlighted with light gray were excluded due to them being shut down. In addition, in order to wipe off confusion, it should be clarified that there is a difference between source code hosting platforms in general and OSS hosting platforms.

As shown in Table 1, 12 OSS hosting platforms were selected, and 2 were excluded which are Google code and JavaForge. Google code has announced on March, 2015 that it has been shut down, leaving a message admitting that there are other better open source project hosting platforms which have gained ascendancy such as GitHub and Bitbucket [24]. Moreover, JavaForge has as well announced on March, 2016 that it will be shut down permanently [25].

Table 1. Selected OSS Hosting Platforms for Evaluation.

Ref.	Platform
[23]	GitHub
[26]	GitLab
[27]	Bitbucket
[28]	Codeplex
[29]	SourceForge
[30]	Alioth Debian
[31]	Launchpad
[32]	Ourproject.org
[33]	Tigris.org
[34]	GNU savannah
[24]	Google Code
[25]	JaveForge

Among the 10 platforms that were selected, GitHub is the largest one hosting the largest number of OSS projects [19], [20], [22], [35]. Other platforms such as Bitbucket, Gitlab, Codeplex and SourceForge are also popular platforms. According to a survey which was conducted on OSS developers, in order to learn about their preferences around OSS hosting platforms [36], the results have shown important statistical information which is illustrated in Fig. 1.

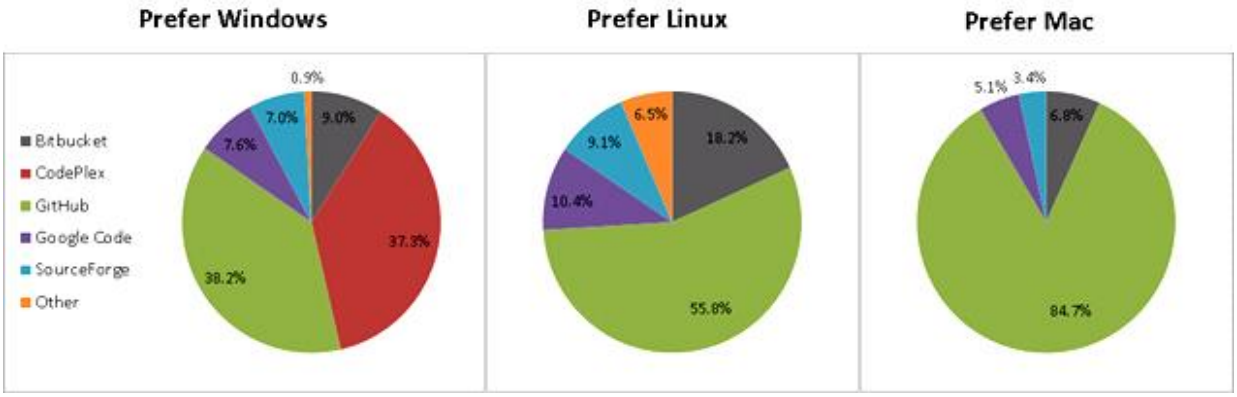

Fig. 1. Survey results showing OSS hosting platforms preferences of OS developers [36].

The results of the survey have revealed the following:

- GitHub has the highest preference among others.
- Codeplex was also preferred in the second place by windows users; however, it is not preferred by Linux and Mac users.
- Bitbucket was a second preference after GitHub.
- Google code was the third preference.
- SourceForge was the fourth preference.

The results of this survey were a motivation to select all OSS platforms mentioned in this survey, in addition to other platforms which might be less in popularity and adequacy.

3.2. Collaboration Features into the OSS Hosting platforms

An exploration of the collaborative features in the 10 platforms is presented in this section. It has been made by registering in and experiencing the platforms, watching video and written tutorials, reading papers and surfing developers' views and experiences about the platforms. As a result of the exploration, there are several features that are integrated into most of the OSS hosting platforms which support community engagement, allowing developers to collaborate effectively and therefore, facilitating the development process of OSS. These features can be categorized into the following types:

- Repository/Project Management. OSS hosting platforms integrate good management solutions into them. Platforms such as GitHub, SourceForge, Launchpad, Bitbucket and Gitlab all allow creating multiple teams and assigning each with certain repository access permissions. Thereby, instead of assigning permissions to every single member, creating teams with different permission and then inviting members into those teams is an available management approach. The owner of a project can create unlimited number of teams, name them and assign them with certain access permissions [21][23][26][27][29][31]. The following is a simple clarification of how teams can be organized and managed:

- Developers' team: have write and read access permission to repositories.
- Reviewers' team: have read access permissions to repositories.
- Owner's team: have extensive access permission among all repositories.

In addition, all the work including issues and pull requests need to be organized properly. Few platforms have offered project management features providing developers with the ability of managing their work on a repository. GitHub for instance, offers a project management tool which is a new feature that GitHub has recently announced. These tools permit developers to create columns representing phases such as TODO, in progress or done. These phases are customizable, and issue and pull request can be dragged and dropped directly into these phases using a Kanban-style interface [23].

- Social Features. There are plenty of OSS hosting platforms which incorporate social networking features, for that, these platforms might also be called social coding platforms. Social functionalities have made a remarkable change in collaborative software development [19]. Moreover, the involvement of social features has simplified interactions between developers on a larger scale. GitHub and Bitbucket in particular are platforms with powerful social features [23][27]. Through these platforms, a developer:

- Has his/her own personal profile with basic personal information.
- Follows other developers with similar interests, and can be followed by other developers in the community.
- Watches other developers' activities and having the ability to be notified about them. Notifications are used to trigger engagement, and save time spent on browsing for developers every time he/she feels curios about their progress.
- Starring interesting projects in order to keep track of their updates [19], [20].

These following and followers mechanism, have an impact in building reputation. The more followers a developer has the higher interest there is on their work [20]. In support of this fact, Dorota [20] stated that: *"The more followers a developer has, the larger the group potentially interested in their work, creating a potential for even greater influence".*

In addition, other OSS hosting platforms might have fewer interactions between their users, due to minimal social media functionalities incorporated into them. Among all OSS hosting platforms only Github and Bitbucket have the following and followers mechanism between developers, while Codeplex allows developers to follow projects instead of developers [23], [27], [28]. However, this does not mean that the other platforms do not involve any social interactions. In fact, all OSS platforms provide their registered users with their own profile but the amount of information presented in a profile differs from one platform to another.

Some platforms provide valuable information on a developer's profile, while others present very basic information which does not help in forming an impression about a developer. Through a good developer's profile, a person can easily obtain a quick idea on how active a developer is and what are his\her interests. A contribution graph in a developer's profile is a powerful indictor which summarizes all contributions and activities done by the owner of the profile from when the profile is originated until the present time. Only GitHub and Bitbucket use contribution graphs in their developers' profiles [23] [27], while other platforms might show all or most recent activities of the developers on their profiles as a stream with no graph. On the other hand, some platforms do not show any recent activities in a developer's profile and are satisfied with very basic information.

- Code Review. Having an integrated code review feature into the platform is crucial which is missed in some OSS hosting platforms such as Codeplex, Aliothdebain and SourceForge [28] [29] [30]. Code review assists in providing better OSS quality. In some cases a developer would like to ensure the validity of a piece of code, for a certain feature or a bug fix, before merging it into the main codebase. Using the code review practice allows developers to crowdsource reviews from core and external developers [37]. A code review allows project owners, collaborators and reviewers to initiate a discussion about the code. Reviewers then can provide valuable feedback and suggestions, and the core developers then decide whether to consider

their feedback and commit changes [37][38].

- Pull Requests. Since almost all OSS hosting platforms currently use distributed version control system (DVCS) such as Git version control system, a new approach for collaborating on distributed software development has emerged which is called pull requests (PR) [39]. In pull requests, rather than pushing changes into a repository, developers actually pull those changes from other repositories and merge them into theirs [39]. Contributions to source code are normally done using pull requests [38]. Code review can be done using pull requests. However according to GitHub and other platforms which also follow GitHub's workflow, there are two pull request models. The first model type is used for code review. Meaning that project owners and collaborators can create a pull request to review and open up a discussion about proposed changes in the source code and can request this pull request review form a certain reviewer. Moreover, reviews can be obtained from people outside the development team as well [23][38].

On the other hand, the second pull request model is used when a contributor, who has only read access into the repository, would like to cooperate in its improvement. The contributor proposes some changes on the repository after forking it, and asks the maintainers of the project to take them into consideration and merge them to their main repository [23]. The project's maintainers then decide whether to merge the contributor's work into their work or reject it. In both models a discussion is held, and precious knowledge, experiences and opinions can be shared through the discussion. The first model is used to review a code developed by the project maintainers, where the second model is initiated by contributors and sent to maintainers to be reviewed. Both are intrinsic practices which exist in most widely known OSS hosting platforms supported with a neat graphical user interface.

- Issue Tracking. An issue tracker allows contributors and collaborators to collaborate on issues, and offers better management of bugs, defects, feature requests, tasks, enhancements… etc. [21] [22]. However, some platforms use bug trackers which mainly focus on bug reports, but the majority of the platforms use issue trackers allowing tracking of issues other than bugs [22]. For instance, Launchpad uses a bug tracker where GitHub, Bitbucket, Codeplex and more use issue trackers. Through issue trackers contributors can easily create issue reports, tag them under a certain category (bug, feature request, enhancement, question…etc.), search and filter them, assign them to certain developers or even open up a discussion about a certain issue if they wish. On the other hand, collaborators can manage the status of issues by updating them or marking them as closed or reopening them in case there was a need to reconsider some issue [21][23][22].

As a result of the exploration, a detailed feature matrix is created as shown in Table 2 demonstrating the availability of several kinds of capabilities and collaborative features in these platforms.

From Table 2 we can observe also the following:

- All platforms allow visitors to search for publicly available OSS projects and download their source code.
- All platforms except two allow users to filter OSS projects according to specified criteria.
- All platforms allow project owners to add collaborators into their OSS projects.
- GitHub and Gitlab are the only platforms showing a contribution (activity) graph on their developers' profile.
- In all platforms, developers can subscribe to a project to receive notifications about all its updates.
- All platforms have issue trackers except two which use a separate tracker for bugs, where developers can easily collaborate around, and hence, issues and bug reports can be easily created managed, assigned and discussed.
- Some platforms offer a pull request as a flow for code contributions, and others allow submitting files of patches.
- Four out of ten platforms have code review as one of their capabilities.

Table 2. Detailed Feature Matrix of OSS Hosting Platforms.

● Indicates that the platform has the whole feature
○ Indicates that the platform has part of the feature
Cells highlighted with gray indicate that the feature can be done by or shown to visitors. No registration or signing in required.

		OSS Platforms									
OSS Hosting Platforms Features		GitHub	GitLab	BitBucket	Codeplex	SourceForge	Alioth Debian	Launchpad	Ourproject.org	Tigris.org	GNU Savannah
Repository/Project Management	Create a repository/project	●	●	●	●	●	Project creation restricted to admins	●	Project must be approved first	No longer accepting new projects proposals	Projects must be approved first
	Delete a repository/project	●	●	●	●	●		●			
	Upload source code to a repository:	●	●	●	●	●		●			
	1 Via drag and drop files	●									
	2 Creating & editing code online directly in the browser	●	●	●							
	3 Import a repository from other OSS hosting platforms	●	●	●		●					
	4 Via a command-line	●	●	●	●	●	●	●	●	●	●
	Search for other users' repositories in the OSS community	●	●	●	●	●	●	●	●	●	●
	Filter repositories/ projects by several criteria	●	●	●	●	●	●		●	●	
	Invite a collaborator to a repository/project	●	●	●	●	●	●	●	●	●	●
	Set repository/ project access permissions	●	●	●	●	●	●	●	●	●	●
	Subscribe to a project or work item– receive notifications	●	●	●	●	●	●	●	●	●	●
	Permit private repositories/projects	●	●	●				●			●
	Access and download repository/project source code	●	●	●	●	●	●	●	●	●	●
	Show the history of all commits for a repository via UI	●	●	●	●	●	●	●	●	●	●
	Post project openings to find additional team members				●		●				●
	Search for project openings to join a project				●		●				●
	Requesting to join a project using UI (not via email)				●		●	●	●	●	●
	Create many groups/teams for a project	●	●			●		●			
	Invite members into groups/teams of the project	●	●	●		●		●			
	Set projects access permissions for each groups/teams	●	●	●		●		●			
	Organize work items in phases (TODO, in progress, done…etc.)	●	●				●		●		
	Review projects				●						
Social Features	Personal profile with a bio, followers, following…etc.	●		●	●						
	Follow other developers in the OSS community	●		●							
	Follow projects / repositories				●						
	Starring projects / repositories	●	●								
	Show trending developers (popular developers)	●		●			●	●			
	Show trending projects (popular projects\ repos)	●	●	●	●	●	●	●			
	Rate projects				●	●					
	Peer rating						○				
Code Review	Request a code review - review piece of code before merge	●	●	●				●			
	Select reviewer	●	●	●				●			
	Discuss the code being reviewed	●	●	●				●			
	Merge reviewed code to the main or master branch via UI	●	●	●				●			
Pull Requests	Create a pull request (merge request) in support of UI	●	●	●	●	●		●			
	Show number of forks for each repository/ project	●	●	●	●	●					
	Accept or decline a pull request via UI	●	●	●	●	●		●			
	Merge a pull request to the main or master branch via UI	●	●	●	●	●		●			
	Discuss pull requests in pull requests section - comments	●	●	●	●	●		●			
	Filter pull requests (opened, closed, merged)	●	●	●	●	●		●			
	Assign pull requests to collaborators.	●	●	●	●	●		●			

Issue/bug Tracking	Uses issue tracker (bug, feature request…etc.)	●	●	●	●	●	●		●	●	
	Uses bug tracker							●			●
	Change issue/ bug statuses by an owner or collaborators	●	●	●	●	●	●	●	●	●	●
	Filtering issues by their types (bug, feature request…etc.)	●	●	●		●	●		●	●	
	Filtering issues/ bugs by other criteria (closed, opened, vote...etc.)	●	●	●	●	●	●	●	●	●	●
	Discuss issues/ bugs in the issues/bugs section by commenting	●	●	●	●	●	●	●	●		●
	Mention users in a comment/involve user into a dissuasion	●	●	●	●	●	●	●	●		●
	Prioritize issues/ bugs	●	●	●	●	●	●	●	●	●	●
	Rate or vote for issues/ bugs	●	●	●	●	●		●		●	●
	Assign specific issues to collaborators	●	●	●	●	●	●	●	●	●	●
	Feature request by submitting forms and files of patches.						●		●	●	●
	Manage dependencies between work items (e.g. issues, merge requests)	○	○	○				○		○	○
Others	Graph summarizing contributions and activities	●	●								
	Contributions/activity stream	●	●	●	●	●		●			●
	Dashboard for the developer's personal account	●	●	●	●	●	●	●	●	●	●
	Wikis	●	●	●	●	●			●	●	
	Discussion boards				●	●	●	●	●	●	
	Mailing lists				●	●	●	●	●	●	●

- Only Codeplex, Aoithdebian and GNU savannah allow developers to post project openings in order to find new members to join their teams, and allow other developers to search for those posts to find projects to join in.
- SourceForge and Codeplex allow users to rate and review projects.
- All platform except three, show the most trending projects.
- Only four platforms out of ten, show the most trending and popular developers in their community.
- All platforms except three provide wikis which are normally used for collaborative documentation.

4. OSS Hosting Platforms Limitations Analysis

4.1. The Limitations

After exploring and evaluating some existing OSS hosting platforms, it has appeared that there are several limitations that have been found in them. Despite all the highly services that they offer, they are missing some collaborative capabilities that are required during the development of an open source software. Moreover, resolving these limitations shall contribute to the quality enhancement of the produced open source software. The following are some limitations which have been shaped as questions. Furthermore, a clear objective for each limitation has been provided demonstrating the potential value from solving it.

Limitation *1*

How can an OSS project owner reduce the impact of the time zone differences resulting from the development team ***members being distributed? And how can project owners find developers nearby to join the team?***

The success of open source software strongly depends on how effectively the team members are collaborating. A study was conducted by Daning et al. [40] on how the geographical distribution of the development teams affects the success of the OSS project and the productivity of the team as well. The study has analyzed a large set of real OSS projects and has investigated the correlation between the rate of the geographic differences between the team members and the OSS project's rating score representing the project's success. The study has revealed that the distances between the team members has a negative impact on the OSS project's success. The authors have also stated in their study that members located in places with

the same or with just little differences in their time zone are likely to collaborate more effectively. Hence, the closer the open source software team members are, the better the collaboration is, and thus, the higher the possibility it is of the OSS project's to be successful.

Moreover, the study [17] has as well revealed that the geographic distribution has a clear impact on collaboration. The study was done on GitHub users and it has shown that users like to interact with other users in the platform who are located nearby. Additionally, the study has declared that small teams usually involve collaborators who are located around specific location instead of being scattered around the world.

Therefore, from these findings it is important to consider the geographical distribution of the OSS development team since it can negatively affect the collaboration process. From the OSS hosting platforms that have been evaluated in the previous section, the platforms that allow project owners to manage and invite members into their teams usually do not provide capabilities that can help them in finding members that are located in places nearby. There are no suggestions that can assist the project maintainers in performing a good selection of team members with regard to their geographical locations. Indeed, considering this issue is requisite, since it has a direct influence on OSS projects, and as well gives teams the chance to hold face-to-face meetings which are one of the rarely performed practices in OSS development.

Objective. Improving the process of managing teams and members' invitation into OSS teams which collaborate around an OSS, by allowing project maintainers to establish a team involving members located at the same place, or at places with little differences in their time zones. The geographical distribution of team members is an issue that cannot be completely solved, but its effect can surely be minimized to a large degree. As have been previously mentioned, the OSS development lacks face-to-face meetings. Through this improvement, team members will be able to collaborate efficiently due to little or no time zone difference between them, or even have the higher chances to hold face-to-face meeting if necessary. In addition, members located close to each other tend to normally have the same culture and language, which this as well has its impact on OSS development.

Limitation *2*

How can project owners find team members with the suitable technical or programming skills that the open source software project needs in order to increase the probability of delivering high quality OSS?

Having developers with good technical skills or experience in the domain where the OSS project is working on is more likely to produce OSS of higher quality. Since the whole OSS development is performed collaboratively through the Internet, it is hard to figure out which developer better suits the project and which does not. Apparently, through the exploration of the set of OSS hosting platforms in this paper, the platforms that enable project owners to create and manage teams and invite members into their OSS project do not offer any assistance in finding volunteer developers with the required skills that better suits the OSS project to collaborate around it.

Objective. The main objective is to improve the process of managing OSS teams by helping OSS project owners find members for their teams who have skills that comply with the nature of their projects. Having the capability of offering wise suggestions to project owners on who to invite to their teams shall effectively help in forming a good team for the OSS project to collaborate around. Furthermore, for instance, depending on the programming language used in an OSS project, a set of developers who have this programming language skills and experience can be suggested and prioritized for project owners to invite.

Limitation *3*

How can OSS development teams discover and manage dependencies between elements of work, such as issues, on ***the open source software project?***

Some issues (e.g. features, defects, enhancement), tasks, pull requests/merge requests or work items in

general must be completed before others. For instance, in order to develop a specific feature, a bug must be fixed in advance. Dependencies between any elements of work should be fully managed. Managing dependencies will direct team members on scheduling their work.

Even though some platforms as demonstrated in Table 2, can assist in managing dependencies, for instance Launchpad offers some management of dependencies between merge proposals, and Github, Gitlab and bitbucket have been offering some practices, such as using references, that might indirectly somehow help in managing dependencies to some extent. However, these capabilities offer basic management of dependencies and requires improvements in order to for it to be more effective. Hence, there is a need for managing dependencies using efficient collaborative mechanisms that are dedicated for tracking dependencies of all types and between any type of work items and are effective even within large OSS projects.

Objective. The goal is to help collaboratively in identifying, managing and tracing any dependencies between elements of work and keep the affected OSS team members informed. By doing so, this shall help in the affordance of effective collaboration and guidance around the OSS, and thus, producing an open source software of better quality.

Limitation *4*

How to assist in increasing commitments from volunteer developers in the OSS community?

Due to the voluntary nature of the OSS development, contributors to the OSS projects cannot be considered responsible for any part of it. This reliance on volunteer developers might lead to lower quality of the OSS [4]. By exploring the current OSS hosting platforms, it has appeared that work elements on the OSS project (e.g. issues, code reviews and pull requests) can be assigned to developers and to specific millstones. Millstones are used to define deadlines for accomplishing tasks. However, the assignees can decide to postpone these tasks to an unknown time or even simply choose to ignore them. Developers are not held accountable for exceeding deadlines since they are volunteers. Therefore, having some mechanisms that can help in encouraging developers to accomplish their tasks before reaching the deadline is necessary, which this is missing in current OSS hosting platforms.

Objective. The goal is to develop a mechanism which can help encouraging developers collaborating on an OSS project to fulfill their commitments by accomplishing tasks that they are assigned to within the deadline. For instance, once a developer performs a task within a specific deadline, he is granted a point. Volunteers usually care about their reputation; therefore, having such mechanism indirectly assists in making the voluntary environment stricter. Through such mechanism, developers in the OSS community can see how punctual other developers are from their points.

Limitation *5*

How can project owners/developers in the OSS community have a clear impression about other developers in the community before working with them?

OSS team members are geographically distributed; therefore, having face-to-face meetings is almost imposable. Thus, since team members only collaborate virtually through the Internet usually using text, it is very difficult for members in the open source community to be acquainted with one another's personality such as trustworthiness, easy or hard to work with, abilities… etc. [6]. Out of all OSS hosting platforms that have been explored, there is not any way to form an accurate impression about other developers. Even though there are some platforms such as GitHub and Gitlab use metrics which are displayed using graphs on a developer's account showing contributions that he has made over time, however, they only show the number of contributions and when they have been done without showing how good they are, or even other personal aspects of a developer. Indeed, forming accurate impressions about other members is a critical issue which needs to be considered before picking peers in the OSS community to work with on a project. In fact, in Table 2 it appears that only one platform (i.e. Alioth Debian) has a peer rating feature, however, it requires

enhancements to be able to allow a more accurate impression formation about others in the OSS community. The platform shows fixed factors to rate a developer according to, where a person might need to know other personal aspects than the ones mentioned. In addition, a developer might rate another by selecting from a scale, but on the other hand, some explanation are needed that why a developer deserves such rating. In addition, not all ratings are reliable; therefore, the results of the peer ratings might lead developers to misjudge each other. Hence, peer rating should be more accurate and not restricted by choosing from a limited number of options.

Objective. The aim is to define a good method that can help in peer impression formation. After working with peers in the OSS community, the developers should be able to rate and assess each other using practices that can enable them be more precise in their ratings. Computer-based judgments are not always accurate, but are supplementary. Having people evaluating and rating others is more accurate and valid than having them rated automatically according to some calculations that the platform does. Therefore, this shall encourage developers to collaborate with other developers which they feel they are aware of.

4.2. Evaluation of Limitations

After analyzing the limitations and challenges which have been deduced from evaluating the OSS hosting platforms, in this section the set of limitations are evaluated and assessed. In order to obtain accurate judgments on how beneficial it is overcoming the limitations in such platforms, a survey has been conducted.

The survey has been distributed to users of OSS platforms. The users are developers ranging from experts to beginner developers. The survey was distributed to developers from around the globe and has not focused on one region. In addition, the survey has been distributed through emails to approximately more than 400 developers, and 40 developers have responded. The survey asked about how beneficial it is to add a process or a capability which overcomes the encountered limitations. Moreover, the survey used a scale from 1 to 5, were 5 is very beneficial as a response option for each question. The following illustrate the findings resulting from the survey:

- It has appeared that the question concerning *Limitation 1* has received diverse responses. See Fig. 2.

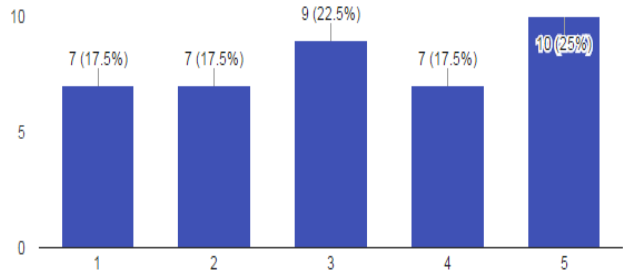

Fig. 2. Summary of responses for limitation 1.

- The question concerning *Limitation 2* has as well received diverse responses, however, most responded with 4 and 5 from the scale indicating the requisite of overcoming such limitation (Fig. 3).

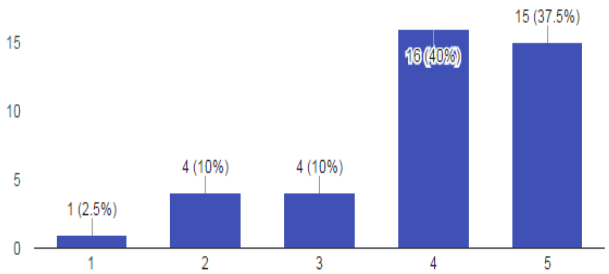

Fig. 3. Summary of responses for limitation 2.

- The question regarding *Limitation 3* has received responses that were mostly around 4 and 5 from the scale. See Fig. 4.

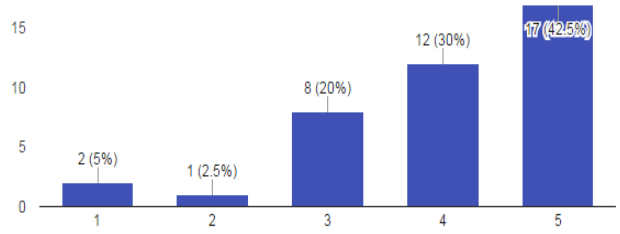

Fig. 4. Summary of responses for limitation 3.

- The question regarding *Limitation 4* has received slightly different responses; however, most responses show a desire for overcoming such limitation. See Fig. 5.

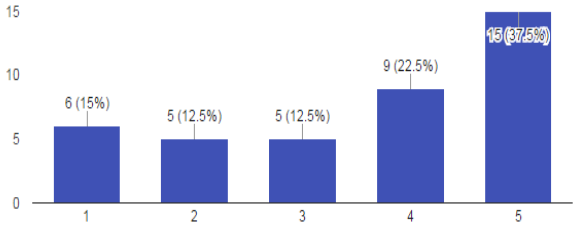

Fig. 5. Summary of responses for limitation 4.

- Eventually, the question concerning *Limitation 5* has received responses showing different point of views, but they are mostly centered on 3, 4 and 5 from the scale. See Figure 6.

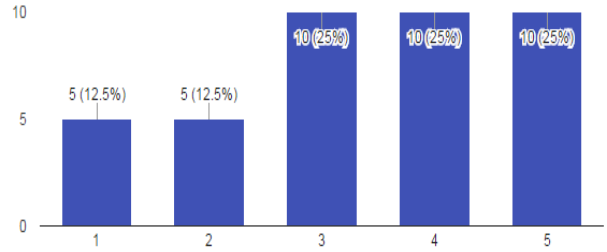

Fig. 6. Summary of responses for limitation 5.

Regardless of the different responses showing the differences in the developers' point of views, which this is normal in any survey, it can be generally concluded that resolving the previously analyzed challenges is desirable by users of the OSS hosting platforms. It has appeared that Fig. 2 which is concerned about *Limitation 1* shows that the developers have different views, which this relatively goes against the studies [17] and [40] that show that there is a strong correlation between team distribution and collaboration. From the researchers' own view, these relative mismatching between the results and the studies might be due to the respondents' unawareness of such issue. Nevertheless, the respondents who are against this limitation is around one-third only.

Moreover, regarding Figure 3 which is concerned about *Limitation 2*, it has been shown that 77.5% of the respondents (40% responded with 5 and 37.5% responded with 4) show high demand for overcoming *Limitation 2*. Meaning that, having the ability to find developers with skills and experience which match with the OSS projects' needs is very beneficial.

Furthermore, Figure 4 shows that 42.5% have responded with 5 from the scale indicating that managing

dependences between elements of work is very beneficial. Besides, 30% have responded with 4 from the scale indicating as well a high requisite for overcoming *Limitation 3*.

Moreover, as shown in Figure 5, it can be inferred that 60% of the respondents (37.5% responded with 5 and 22.5% responded with 4) think that having a mechanism which can increase commitments in the voluntary OSS environment is beneficial. The rest have different responds ranging between 1, 2 and 3, and this might be due to some developers preferred to be completely free from any constraints. However, there is a large percentage indicating the need to overcome *Limitation 4*.

Finally, Fig. 6 representing question 5 about *Limitation 5*, shows different point of views where most responses are centered around 3, 4 and 5 in the scale where each have received 25% of responses. Regardless of the other responds, it can be generally concluded that there is to some extent a need for a method which can assist in forming accurate impressions about other developer in the OSS community such as peer rating.

5. Conclusions

Open source software is a software paradigm which has been and is still growing at an exponential rate. In addition, there are great predictions for a brighter future for such paradigm. The OSS development has as well gained the attention of many researchers. Due to the nature of the OSS development process which has many variations compared to its corresponding traditional closed source software development process, many concerns have raised. OSS development is well-known for its distributed and voluntary environment.

In this research, an elaborative review about OSS development and its hosting platforms has been provided. Moreover, the research has deeply discussed the issues that OSS development includes, and has mainly focused on the most critical factors directing and affecting the collaboration during the OSS development process. Furthermore, OSS hosting platforms offer a large space for collaborative development over OSS projects, and are supported with advanced features and as well as a massively large community.

Large investigation has been performed in this research on the existing OSS hosting platform and some important limitations have been deduced. Additionally, for the sake of assessing the significance of overcoming the limitations, a survey has been distributed to a large target group of developers who are familiar with OSS hosting platforms. The survey has generally revealed that there is a necessity to overcome such limitations. Even though the OSS hosting platforms have offered good features to support the whole OSS development process, there are still some areas where there are opportunities for improvements in the collaboration aspect of these platforms, and subsequently increasing the possibility of delivering successful and high quality OSS.

As future plan, the authors attempt to work intensely on solving the limitations and challenges that have been deduced from this part of the research. Furthermore, it is planned to produce a collaborative approach that helps developers in the OSS community collaborate and therefore, develop open source software of high quality.

References

[1] Deshpande, A., & Riehle, D. (2008). The total growth of open source. *IFIP – The International Federation for Information Processing Open Source Development, Communities and Quality*, 197–209.

[2] Villarrubia, A., & Kim, H. (2015). Building a community system to teach collaborative software development. *Proceedings of the 2015 10th International Conference on Computer Science & Education*.

[3] Hayashi, H., Ihara, A., Monden, A., & Matsumoto, K.-I. (2013). Why is collaboration needed in OSS projects? a case study of eclipse project. *Proceedings of the 2013 International Workshop on Social Software Engineering*.

[4] Farooq, S. U., & Quadri, S. M. K. (2012). Quality practices in open source software development affecting

quality dimensions. *Trends in Information Management*, *7(2)*.

[5] Lanubile, F., Ebert, C., Prikladnicki, R., & Vizcaino, A. (2010). Collaboration tools for global software engineering. *IEEE Software*, *27(2)*, 52–55.

[6] Bosu, A. (2013). Modeling modern code review practices in open source software development organizations. *Proceedings of the 11th International Doctoral Symposium on Empirical Software Engineering*.

[7] Tu, G. Q. (2000). Evolution in open source software: A case study. *Proceedings International Conference on Software Maintenance*.

[8] Potdar, V., & Chang, E. (2004). Open source and closed source software development methodologies. *Proceedings of the International Conference of Software Engineering*.

[9] Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., & Ye, Y. (2002). Evolution patterns of open-source software systems and communities. *Proceedings of the International Workshop on Principles of Software Evolution*.

[10] Singh, A., Bansal, R., & Jha, N. (2015). Open source software vs proprietary software. *International Journal of Computer Applications*, *114(18)*, 26–31.

[11] M. Aberdour, "Achieving Quality in Open-Source Software," *IEEE Software*, vol. 24, no. 1, pp. 58–64, 2007.

[12] Salameh, H. B., & Jeffery, C. (2014). Collaborative and social development environments: A literature review. *International Journal of Computer Applications in Technology*, *49(2)*, 89.

[13] Ankolekar, A., Herbsleb, J., & Sycara, K. (2003). Addressing challenges to open source collaboration with semantic web. *Proceedings of 3rd Workshop on Open Source Software Engineering, the 25th International Conference on Software Engineering* .

[14] Thayer, R. H., McGettrick, A. D., & Budgen, D. (1993). *Software Engineering: A European Perspective*. Los Alamitos, CA: IEEE Computer Society Press.

[15] Jarczyk, O., Gruszka, B., Jaroszewicz, S., Bukowski, L., & Wierzbicki, A. (2014). GitHub projects, quality analysis of open-source software. *Lecture Notes in Computer Science Social Informatics*.

[16] Yoshikawa, Y., Iwata, T., & Sawada, H. (2014). Collaboration on social media: Analyzing successful projects on social coding.

[17] Lima, A., Rossi, L., & Musolesi, M. (2014). Coding together at scale: GitHub as a collaborative social network. *Proceedings of the ICWSM*.

[18] Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. (2012). Social coding in GitHub. *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*.

[19] Jiang, J., Zhang, L., & Li, L. (2013). Understanding project dissemination on a social coding site. *Proceedings of the 2013 20th Working* Conference *on Reverse Engineering*.

[20] Celińska, D. (2016). Who is Forked on Github? Collaboration among Open Source Developers.

[21] Qusef, A., Albadarneh, I., & Albadarneh, A. (2015). GitBull: Source code hosting web application. *Proceedings of the IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*.

[22] Bissyande, T. F., Lo, D., Jiang, L., Reveillere, L., Klein, J., & Traon, Y. L. (2013). Got issues? Who cares about it? A large scale investigation of issue trackers from GitHub. *Proceedings of the 2013 IEEE 24th International Symposium on Software Reliability Engineering*.

[23] Build software better, together. Retrieved, from: https://github.com/

[24] Bidding farewell to Google Code. Retrieved, from: https://opensource.googleblog.com/2015/03/farewell-to-google-code.html/

[25] Javaforge shutdown. *Javaforge shutdown*. Retrieved from: https://codebeamer.com/cb/wiki/938267/.

[26] Code, test, and deploy together with GitLab open source git repo management software. *GitLab*.

Retrieved from: https://about.gitlab.com/
[27] The Git solution for professional teams. Retrieved from: https://bitbucket.org/product/
[28] CodePlex. Retrieved from: https://www.codeplex.com/
[29] Find, Create, and Publish Open Source software for free. Retrieved from: https://sourceforge.net/
[30] Alioth: Welcome. Retrieved from: https://alioth.debian.org/
[31] Launchpad. Retrieved from: https://launchpad.net/
[32] Welcome. Retrieved from: [Online]. Available: https://ourproject.org/
[33] Open Source Software Engineering. Retrieved from: http://www.tigris.org/
[34] Welcome. Retrieved from: http://savannah.gnu.org/
[35] Brindescu, C., Codoban, M., Shmarkatiuk, S., & Dig, D. (2014). How do centralized and distributed version control systems impact software changes? *Proceedings of the 36th International Conference on Software Engineering*.
[36] Survey results: Open source developer preferences. *CodePlex Weblog*. Retrieved from: https://blogs.msdn.microsoft.com/codeplex/2010/11/27/survey-results-open-source-developer-preferences/
[37] Beller, M., Bacchelli, A., Zaidman, A., & Juergens, E. (2014). Modern code reviews in open-source projects: which problems do they fix? *Proceedings of the 11th Working Conference on Mining Software Repositories*.
[38] Yu, Y., Wang, H., Yin, G., & Ling, C. X. (2014). Reviewer recommender of pull-requests in GitHub. *Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution*.
[39] Gousios, G., Pinzger, M., & Deursen, A. V. (2014). An exploratory study of the pull-based software development model. *Proceedings of the 36th International Conference on Software Engineering*.
[40] Hu, D., Li, X., & Zhang, X. (2016). The impacts of geographic dispersion on OSS project success: Face-to-face vs. virtual collaboration. *Proceedings of the Thirty Seventh International Conference on Information Systems.*

Ghadah Alamer has received her bachelor's degree in information systems, college of computer and information sciences with first class honors from Princess Nora bint Abdulrahman University, KSA. She is currently in her last semester as a master's student in information systems, college of computer and information sciences in King Saud University, KSA. Her current research interests are in CSCW (computer-supported cooperative work), distributed development and software project management.

Sultan Alyahya received his PhD degree in Computer Science from Cardiff University, UK, in 2013. He also received his MSc degree in Information Systems Engineering from the same university in 2007. The BSc degree was obtained with honors in information systems from King Saud University. Dr. Sultan is currently an assistant professor at the College of Computer and Information Sciences, King Saud University. His main research interests are in the fields of Software Project Management, Agile Development and Computer Supported Co-operative Work (CSCW).