International Journal on Perceptive and Cognitive Computing (IJPCC) Vol 9, Issue 2 (2023)
https://doi.org/10.31436/ijpcc.v9i2.398

Modeling the Workflow of Bug Prioritization Tasks Descriptively Using the Past Events

*Sohaib Altaf Raja, [1]Madihah Sheikh Abdul Aziz, [2]Asadullah Shah
Dept. of Computer Science, KICT, International Islamic University Malaysia, 53100 Kuala Lumpur, Malaysia

*Corresponding author: suhaibraja@hotmail.com
(Received: 3rd April 2023; Accepted: 20th May 2023; Published on-line: 28th July 2023)

Abstract—Prioritizing bugs is one of the critical decision-related tasks in managing the maintenance phase whereas it is exposed as a key challenge in handling bug reports. On the other hand, the bug triager is a prominent role to observe influencing factors for handling the bug prioritization tasks effectively. Analysis of previous bug reports shows that it is essential to handle bug prioritization tasks with the appropriate workflow. However, it is revealed that there is a research gap in modeling the workflow of prioritization tasks. The paper aims to characterize the workflow model of prioritization tasks. This research is based on a document analysis design using qualitative data from previous bug reports and other artefacts. Over 100 bug reports from large software corporations are accessed and filtered, while 20 bug reports are used for obtaining empirical data. In this study, a descriptive workflow model for prioritizing bugs is proposed by analyzing past events. This model characterizes the states of bug prioritization tasks, their statuses, and the transitions between them. Additionally, this research analyzes the industrial aspect of the proposed model and demonstrates its usefulness in providing valuable insights to the bug triager into ongoing prioritization tasks that will assist him in decision-making in prioritizing bugs retrospectively and prospectively. The finding of this research also reveals that bug reports are a valuable resource that contains significant prioritization features which is useful for illustrating the workflow of bug prioritization tasks descriptively. Thus, the implications of the model for theory and practice are discussed.

Keywords— Bug Priority, Workflow Model, Bug Handling, Decision-Making, Previous Bug Reports, Bug Triagers' Role

I. INTRODUCTION

Software projects face various challenges during the maintenance phase [1]–[6]. Prioritizing bugs comprises critical decision-related tasks that have a considerable impact on the maintenance phase and become a challenge in the management of software development projects. Bug prioritization studies evaluate various factors that affect the prioritizing of bug reports [7]–[9]. On the other hand, the role of the bug triager is prominent in observing influencing factors that impact decision-making in prioritizing bug reports and handling tasks associated with bug prioritization [9]–[11]. The literature describes the lifecycle of bug reports by illustrating the states of bugs and the transitions between these states[11], [12]. A review of the empirical data from the bug reports' history indicates several tasks connected with Triaging or Prioritization states, whereas there is a research gap in characterizing the prioritizing tasks descriptively. The transition arrows with the red color of Fig. 1 indicate this gap.

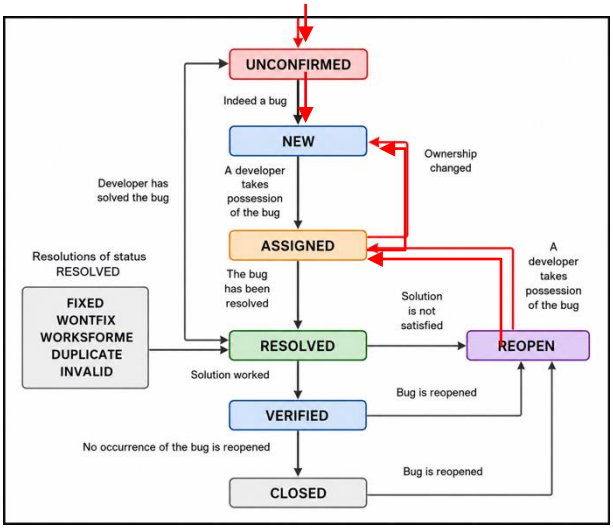

Fig. 1. Display the Lifecycle of a Bug, its States and Transitions and Switching into Different States, and the Rationale Behind these Transitions [11].

A. *Workflow of Handling Bug Report*

The literature describes the lifecycle of bugs through their reporting cycle till their closure and deployment. The bug-handling process of Atlassian [13], [14] is described by the workflow activities which are divided into various phases, further classified by different statuses, and shown by transitions between them. Fig. 1 depicts the lifecycle of an Eclipse bug report and displays several aspects of the bug-handling process [11]. It shows the transitions of bug

reports into various states as well as their switching back and forth between different states and as a result, the statuses of bug reports keep updating in the bug report. The arrow directed toward different states shows the rationale behind their transitions[11].

The backlog of bug reports contains the newly created bug reports as well as reopened bug reports. The new bug reports enter in backlog in an Unconfirmed state which implies that bug reports require to be validated before proceeding towards triaging and resolution tasks. The arrow directing toward a Resolved state from an Unconfirmed state shows that a bug is resolved without fixation and various statuses are associated with a bug report for describing the rationale behind the resolution which are described below[11]:

- a bug is valid, but it is a duplicate, therefore fixation is not required for it because it will be addressed automatically when its original bug is resolved. Hence it will be a dependent bug to its original bug and the bug report will be assigned a Resolved state with Duplicate status,
- a bug is not qualified as a valid bug, therefore cannot proceed for fixation. Hence Resolved state with Invalid status will be assigned to the bug report,
- a bug is valid, and fixation is not required for it because it is resolved with a workaround, or any other alternative solution worked. Hence the bug report remains in a Resolved state with a Workaround status, however, this status is not shown here in Fig. 1. This status is examined from many bug reports[i],
- for other statuses, see [11].

The arrow directing toward a New state from an Unconfirmed state indicates that a bug report is a valid bug report and various triaging and prioritizing tasks proceed before its resolution. In literature, these tasks are categorized as repository-oriented or developer-oriented tasks [2], [15]. The former tasks involve the investigation of a bug and the completion and verifiability of its information submitted in a bug report, whereas the latter tasks include the developer selection and prioritization which is indicated by the transition arrows with the red color shown in Fig. 1.

The Fig. 1 shows that the bug reports are put under different states during their lifecycle so that stakeholders could be kept informed about their resolution status, for example, in Eclipse, when a bug is not addressed appropriately and the Reopen state is assigned to it, which implies that the solution is not worked on, therefore the bug report needs re-assignment of a developer or some developer has to take possession of the bug report. In Fig. 1, this is shown by an arrow directing from the Reopen state toward the Assigned state. It is also observed from the history of bug reports that they need to be re-investigated and re-prioritized or other triaging tasks are required for their handling once they are Reopened. According to the literature, various triaging tasks are performed between the Unconfirmed and New states [11]. In Fig. 1, the transition arrows that represent the different prioritization events, and actions are depicted in red color. The figure does not, however, describe the Triaging or Prioritization states, associated statuses, and transitions between them.

The bug reports participate in the triaging process and remain in the Triaged state when they need to be investigated. Further, the right developer must be chosen for them, and the appropriate priority category must be assigned. It has been noticed that the word Triaged is also used in some bug reports to indicate the bug's state. The Triaging or Prioritization states, on the other hand, can inform the stakeholders about various triaging and prioritization events that take place in between the validation and resolution tasks, for instance, triaging and prioritization tasks that perform when a bug report moves from Unconfirmed to a New state; New to an Assigned state; and Assigned to a Resolved state. Fig. 1 demonstrates how a bug can reopen after it has been resolved, verified, and even closed. On the other hand, the history of bug reports reveals a number of tasks connected with Triaging or Prioritization states. The prioritization states, statuses, and transitions that occur between them are therefore analyzed in this paper using past events from the bug reports.

B. Bug Prioritization and the Categories

Software organizations use priority categories to label the priority levels and assign them to the bug reports [12], [15], [16]. They have to prioritize the bugs so that impactful and urgent bugs can be resolved at the earliest. At the Atlassian corporation, highest, high, medium, and low are specific labels that describe the categories of priority [14]. These categories determine the relative importance of the bugs for their resolution. Hence, the purpose of prioritization is to categorize the bug reports considering their importance to schedule their fixation in present, next, or any onward sprints.

The prioritization process is described in the literature as the decision-making task that handles the assignment of the suitable category of priority to bug reports that are waiting for their resolution in a backlog. As a result, the outcome of the prioritization process should be an appropriate prioritization decision that was taken to assign relative importance to bugs and is reflected in bug reports. Therefore, the bug triager is a key factor in decision-making whose role is to triage the bug reports, observe various decision-related aspects and choose the appropriate or better priority decision which is represented by a priority category, for instance, (1) a decision to select either the highest, high, medium, or low priority category for a specific

bug report, (2) select default priority category, (3) not to select any priority category at this stage and leave the priority attribute empty, (4) a decision to change the existing priority category and select either new category of priority or leave the priority attribute empty. The characteristics of bug reports are essential information for the bug triager to gain insights into various choices for decision-making. The priority categories must be evaluated and assigned to bug reports to determine their relative importance for fixation.

In this paper, the study of bug reports reveals that while some bug reports are resolved in accordance with the priority category that was assigned to them, others are delayed because either their priority category changed or because the priority category is still being assigned. It has been noted that some bug reports had empty priority values, indicating that the bug triager in those cases did not select a priority category and left the priority value empty. Because of this, empty would also be regarded as a decision that should be made in cases when either none of the aforementioned priority categories match the criteria or the bug triager is constrained in their ability to select a suitable priority category [14], [17].

C. *Challenges of Bug Prioritization*

Prioritizing bug reports are important decision-related activity among other bug-handling tasks [7], [9], [15] that is essential to handle with the appropriate workflow. In literature, bug prioritization is exposed as a critical challenge for handling bug reports, while plenty of research is conducted to address this challenge [7], [9], [15], [17].

This study analyzes the prioritization workflow by examining several bug reports of Atlassian corporation which tells that the prioritization statuses of some of the bug reports kept on changing when the bug reports switch back and forth between different states. Among them, some of the bug reports are waiting for prioritization in the backlog, whereas others are engaged in the process of prioritization. Hence, the bug reports in the prioritization lifecycle move between different prioritization statuses [14].

An investigation of empirical data from bug reports reveals that bug reports remain in various prioritization statutes during their lifecycle: (1) some bug reports[ii] have no priority value; (2) some bug reports [iii] have priority values but are initially empty before being prioritized; (3) priority is assigned to some bug reports [iv] but needs to be reprioritized; and (4) some bug reports need to be reprioritized [i, v] again; (5) priority is assigned to some bug reports but then they are deprioritized [vi]. The transition between the prioritization status updates the priority value of the bug report and there are various rationales behind the prioritization statuses [14], [18].

The bug-handling process, which Eclipse and Atlassian prescribe for fixing bugs, comprises a sequence of triaging and prioritization tasks [14], [18], [19]. The model shown in Fig. 1 is generic and of a prescriptive nature because it does not describe the triaging and prioritization states. The descriptive model, on the other hand, should describe the exact workflow of each bug report, beginning with its reporting cycle, moving through its triaging and prioritization cycle, and iterating back and forth into different states until their closure and deployment.

This descriptive model should provide the bug-handling team with information about the present status of bug reports in the backlog, which can help them handle bug reports for future tasks such as triaging, prioritizing, and resolution as well as modeling the lifecycle of prioritization tasks descriptively and will give useful insights to the bug triager in decision-related tasks. Therefore, it becomes critical for the bug triager to observe the workflow of bug prioritization to monitor which bug reports are either pending or require reprioritization and hence need urgent attention. Thus, his role as a decision-maker is of utmost importance in which he needs visions into various ongoing prioritization tasks during decision-making for handling bug reports for prioritization.

Many studies model the lifecycle of bug reports [12], [20] however, there is a research gap in characterizing the lifecycle of prioritization tasks associated with handling bug reports. Previous bug reports are valuable artefacts that can be used for understanding the phenomena of bug prioritization [9], [16], [21]. This study examines the prioritizing and triaging tasks from Eclipse and Atlassian workflow processes using previous bug reports and models the workflow states of bug prioritization which illustrates the transitions of bug reports into prioritization states and statuses.

II. Literature Review

This section introduces the related studies on triaging and prioritization tasks and presents briefly issues of bug prioritization.

Empirical studies evaluate various factors to address the decisions-related challenges in handling bugs that are faced by software projects during the maintenance phase [3]–[6]. Plenty of studies report factors that impact the maintenance phase [22]–[24], while several empirical studies on bug handling present statistical findings using datasets of the past bug report to provide [4], [20], [25], [26].

Bug prioritization is an essential decision-related task to address the bug-handling issues faced by software projects. Most of the studies conducted on bug prioritization using datasets of bug reports worked on machine learning, deep learning, and NLP-based approaches, and proposed

https://doi.org/10.31436/ijpcc.v9i2.398

automated techniques [8], [9], [15], [20], [27]. Many studies use textual data such as summary and description, as well as category information such as severity, temporal information, and other factors [17], [28] whilst some demonstrate the significance of dependent bug reports linked as external sources with the bug reports, for the design of the automated techniques [8], [9], [29]. The purpose of the automated techniques is to improve the performance of bug-prioritizing tasks, however, there is still a research gap to put these techniques into practice [9], [15], [20].

Certain empirical studies evaluate the factors and establish their relationship with bug prioritization to examine their impact [7]–[9]. Almhana et al., [21] examine the changeovers in bug priority from a large data set of open-source bug reports and analyze the tasks that are performed when the transition in bug priority decisions took place. Some studies demonstrate the significance of empirical data using datasets from past bug reports by presenting the statistical findings of bug triaging and prioritization tasks [9], [16], [20].

The review of related studies on bug prioritization reports [2], [7]–[10], [15]–[17], [21] as well as looking at the prioritization tasks from previous bug reports, reveal research gaps in addressing decision-related challenges in bug prioritization due to many factors such as constraints on decision-making in prioritizing bug reports, for instance, a large backlog of bug reports; awaiting priority decisions, re-prioritization and changeovers noticed in priority decisions; the role of the bug triager as decision-maker; and lack of automation support. Literature models the lifecycle of bug reports [12], [20], however, examination of studies reveals that there is a study gap in exploring the factors that have an impact on bug prioritization and defining the ongoing tasks of prioritizing bug reports.

III. Methodology

The task of learning from the past is important in software development [30]. Due to open access to datasets of bug reports and other project archives over the decade, empirical research has gained popularity for handling bug reports and became a valuable resource for making data-driven decisions [4], [9], [20], [31], [32]. Many actions are taken during bug-triaging decisions, which are difficult to monitor during the decision-making process, but the empirical study indicates that bug reports are the platform that records the actions of software teams when they communicate [4], [9], [20], [32], [33]. As a result, the outcome of bug priority decisions, as well as the history of the activities done during the bug prioritization tasks, can be traced back, allowing decision-making-related issues in bug prioritization to be studied.

Plenty of research studies on bug handling tasks employ datasets from bug reports as a subject for different statistical assumptions and suppositions. This study is based on a document analysis design. In this study, bug reports from closed-source projects of Atlassian Corporation [34] and other open-source projects, such as those of Mozilla, Eclipse, and Red Hat [35]–[37], serve as the main source of empirical data. The secondary sources of empirical data, such as artefacts outlining the triaging and prioritizing process, are defined by various large software corporations [14], [38]–[43]. These artefacts are used to analyze qualitatively the bug report data.

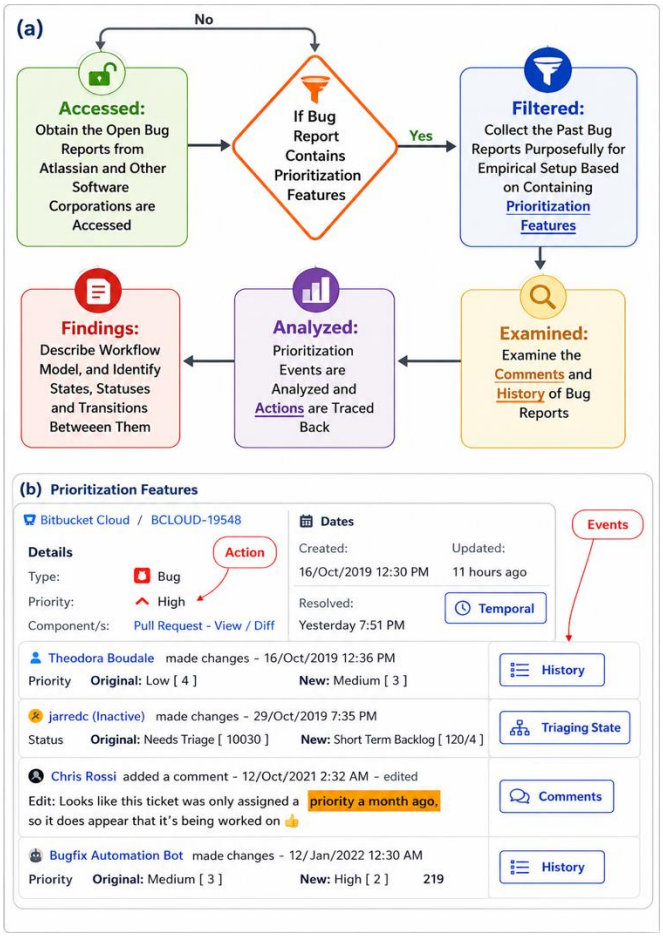

Fig. 2. (a) Illustrates the Data Collection Process of Bug Reports Based on Prioritization Features Contained in it. (b) Display the Different Prioritization Features Present in Bug Reports.

Atlassian, [44] is an Australian software corporation that develops a family of products that are serving over 200000 customers in over 190 countries. The Atlassian product family includes Atlassian-owned and operated products, including Jira, Confluence, web apps, and mobile apps. Jira and Bugzilla are bug-tracking systems, designed to help teams of software developers, project managers, and other software development teams to handle bugs. It gives a

platform to many open-source and commercial software projects. Bugzilla provides custom filters to access the bug reports, whereas Jira provides both the custom filters and JQL to access the bug reports. The data in this study is collected from open and closed-source projects managed in the Bugzilla and Jira bug-tracking systems.

Bug reports [i-viii] are the primary unit of observation in this study that is collected from different software corporations. The sampling technique used in this study is purposive. The dataset of bug reports selected for this study is based on the prioritization features that are examined mainly from the comments and history section of bug reports. It is examined in this study that the prioritization decisions taken are reflected in many bug reports. It is also observed that some bug triager communicates the priority decisions and discusses their rationales in the comments[i]. Therefore, to analyze these features, the comments and history section of bug reports are examined because they contain rich past data. The datasets of past bug reports are examined, and the prioritization events are analyzed from comments, history, and other features of past bug reports which indicates that bug reports contain events that are performed and actions that are taken by the bug triager during the bug prioritization tasks. Therefore, these actions are traced back by examining the outcome of priority decisions.

For empirical setup, bug reports are accessed and filtered in two steps. The first step involves accessing the more than 100 opened bug reports, and the second step includes filtering them based on several prioritization features that should be present in their contents, such as one or more priority decisions; reprioritizations; comments containing discussion about the prioritization; assignment and reassignment of priority shown in the history section; and temporal data related to prioritization. The methodology is shown in the flowchart in Fig. 2. The process of collecting and analyzing data is shown in Fig. 2 (a), while examples of prioritization features found in bug reports are shown in Fig. 2 (b), which is used to evaluate bug reports in an empirical setup. In this figure, the rectangles in blue that have short descriptions are used to tag the prioritization features. In the figure, it demonstrates how events can be analyzed from history, comments, temporal information, and triaging tasks while using this information. For instance, the reason why the current priority is high can be traced back to past events, which indicate that this action was taken on January 12, 2022, and analyzed from comments made on October 12, 2021. These events are used to identify the present and past statuses of bug reports as well as transitions between them, whereas the triaging tasks are used to identify the prioritization states.

IV. Findings

During the handling of the bug reports, it is significant for the bug triager to visualize the workflow of bug prioritization to observe which bug reports need immediate consideration. This study examines previous bug reports, analyzes the workflow of bug prioritization, and illustrates the transitions between prioritization states and statuses in bug reports.

A. *Workflow of Handling Bug Reports for Prioritization Tasks*

An examination of empirical data from bug reports shows that bug reports stay in two states either they are waiting for prioritization or are involved in the process of prioritization. They switch between unprioritized, reprioritized, prioritized, deprioritized, and cannot-be-prioritized statuses during their lifecycle, as a result, the priority value of bug reports keeps on updating, for instance,

- the priority value of some bug reports remains empty,
- some bug reports are prioritized once,
- some bug reports are prioritized more than once,
- some bug reports need to be reprioritized,
- while some bug reports were given specific priority levels, their priority was later changed to empty.

Table 1 describes the workflow model of prioritization according to which the triaging and prioritization tasks are coordinated; some are dependent, whereas others are interdependent tasks. Fig. 3 illustrates it diagrammatically and it shows the collaboration between the two prioritization states. Fig. 4 shows the synchronization between various lifecycle states of bug reports and prioritization states. The digit in a bracket shows the number of bugs moving in a bug-handling pipeline under a specific state. It is supposed in Fig. 4, that twelve bugs are newly reported though many existing bugs are in the pipeline for triaging and resolution tasks. Among newly reported bugs, some bug reports are waiting-for-prioritization state, whereas other bugs are in-process-of-prioritization state. This equation also holds for bug reports that are in the triaging backlog which implies that some triaged bugs are waiting to be prioritized while some are engaged in the process of prioritization. Fig. 3 demonstrates that the bug reports can be either moved to a prioritization state or remain in waiting for the prioritization state, however, it goes through different statuses. It displays that nine bug reports are waiting for prioritization, and the other eleven bug reports are moved to the prioritization state.

The transition between prioritization states is triggered by an event that takes place when a bug triaging team chooses certain bug reports from the backlog for the prioritization process, and as a result of it, the status and

https://doi.org/10.31436/ijpcc.v9i2.398

value of priority change. Fig. 3 illustrates that when the bug reports switch to the process-of-prioritization state, their initial priority statuses and values may change to new priority values. Table 1 illustrates the initial and new values. It shows that prioritization statuses and priority values change in many situations after the transition takes place in the prioritization states. The first part of Fig. 3 shows the transitions between the two prioritization states, the middle part illustrates their existing statuses, and the last part shows that priority values are updated when bug reports switch between the prioritization states.

B. Possible Statuses

Table 1 describes the transitions between the various states and statuses. The first column of the Table shows the two possible prioritization states and the second column show the possible statuses of the bug report and their transitions, for instance, a bug report can switch from one state to another and between the following possible statuses.

1) *'Unprioritized' Status:* The bug report with 'unprioritized' status indicates that the decision to assign any specific priority category was not taken and the priority attribute is left empty. Therefore, the bug report with 'unprioritized' status may need to be either prioritized, remain in unprioritized status, or cannot be prioritized during the process of prioritization.

2) *'Reprioritized' Status:* The bug report with 'reprioritized' status implies that the bug report is assigned a specific priority category more than once. Therefore, the bug reports with the 'reprioritized' status can be either reprioritized again, deprioritized, or cannot be prioritized during the process of prioritization.

3) *'Prioritized' Status:* The bug report with 'prioritized' status means that a specific priority category is already assigned once to the bug report. Therefore, the bug reports in this state can be reprioritized, deprioritized, or cannot be prioritized during the process of prioritization.

4) *'Cannot be Prioritized' Status:* The bug report with 'cannot be prioritized' status implies that they cannot be fixed now. Therefore, the 'cannot be prioritized' status is assigned and hence they cannot be prioritized during the process-of-prioritization state.

5) *'Deprioritized' Status:* The bug reports with 'deprioritized' status implies that priority is initially assigned to the bug reports but later it is removed, and the priority value is now changed to a null value. Therefore, a 'deprioritized' status is assigned to these bug reports.

TABLE I

DISPLAY THE WORKFLOW, THE PRIORITIZATION STATES OF BUG REPORT AND, THEIR TRANSITIONS INTO VARIOUS STATUSES

Prioritization States	**Existing Priority Status**		**New Priority Status**	
	Initial Status	**Initial Value**	**New Status**	**New Value**
The backlog of the Bug Report is Waiting-for-Prioritization (9)	Unprioritized	Empty (3)		
	Prioritized	High (1)		
	Reprioritized	Highest (1), Medium (4) – *Already Assigned Priority More than Once*		
The Backlog of the Bug Report is In-Process-of-Prioritization (11)	Unprioritized	Empty (5)	Unprioritized	Leave Empty (1)
			Cannot be Prioritized	Leave Empty (1)
			Prioritized	Low (2) – *Assign Default Priority*
			Prioritized	Highest (1) – *Assign Appropriate Priority*
	Prioritized	Low (1)	Reprioritized	High (1) – *Change and Assign New Priority*
	Reprioritized	High (1) – *Already Assigned Priority Once*	Reprioritized	Medium (1) – *Change and Assign New Priority*
		Medium (2) – *Already Assigned Priority More than Once*	Reprioritized	High (1), Low (1) – *Change and Assign New Priority*
		Highest (1) – *Already Assigned Priority Once*	Deprioritized	Leave Empty (1)
		Medium (1) – *Already Assigned Priority Once*	Cannot be Prioritized	Leave Empty (1)

https://doi.org/10.31436/ijpcc.v9i2.398

C. *Existing and New Statuses*

The bug reports in existing priority status display their initial priority status and value when they are either in a 'waiting for prioritization' status or moved to a 'process of prioritization' state. In Table 1, the second column shows three initial statuses in which bug reports may remain that are 'unprioritized', 'prioritized', and 'reprioritized' statuses. The third column shows the initial priority value of bug reports after which they can move to any other possible statuses, for instance, a bug report can be empty, or already a priority is assigned to it once or more times.

The fourth and fifth column displays the new priority status and value of the bug reports. It demonstrates how the transition that occurs when the bug reports move between the prioritization states causes the prioritization statuses to change to 'unprioritized', 'prioritized', 'reprioritized', 'cannot be prioritized', and 'deprioritized' and how this updates the priority values.

D. *Bug Reports in 'Waiting for Prioritization' State*

The second column of Table 1 illustrates that nine bug reports in the backlog are waiting-for-prioritization among which three are in unprioritized status, one is in prioritized status and five are in reprioritized status. The waiting-for-prioritization state indicates that the prioritization process of these bug reports is pending. The unprioritized status of bug reports indicates that it has never been prioritized, in which case its priority value is empty, or that it has been prioritized once or more, in which case its priority value has been assigned, but it has moved to the deprioritized status later, where it has changed to an empty value.

Reprioritized status implies that bug reports are already prioritized once or maybe more and are currently waiting to be reprioritized. It is shown in the third column that the three bug reports with unprioritized statuses have empty priority values, one bug report with prioritized status has a high priority value, and among the five bug reports with reprioritized statuses, the existing priority of one bug report is the highest and the other four bug reports are medium.

The fourth and fifth column is empty because the prioritization process of these bug reports is pending and therefore, the value of priority will not update and remain the same.

E. *Bug Reports in the 'Process of Prioritization' State*

The table displays that eleven bug reports moved to the 'process of prioritization' state among which five bug reports are in 'unprioritized statuses' and no priority is assigned before, one bug report is in prioritized status whereas the remaining five bug reports are in reprioritization status and priority values are already assigned to them.

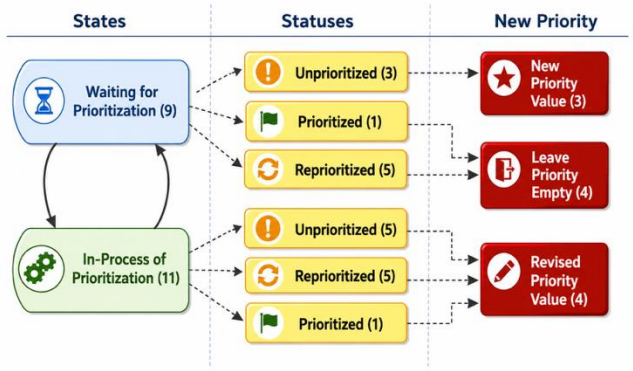

Fig. 3. Illustrates the Switching between Prioritization States, Statuses of Bug Reports, and Priority Values.

1) Unprioritized' Status: Table 1 shows that five bug reports with unprioritized status moved to the 'process-of-prioritization' state. The existing status is shown in the second column whereas the existing value of priority is shown in the third column. The fourth column shows that among five bug reports, one bug report remained in unprioritized status which implies that this bug report is moved to the waiting-for-prioritization state, whereas one bug report is invalid and therefore, cannot be prioritized in the future and is closed. On the other hand, among the three bug reports, low priority is assigned to two bug reports, whereas the highest priority is assigned to one bug report, which is displayed in the fifth column.

2) *'Prioritized' Status:* Table 1 displays that one bug report with prioritized status moved to the 'process of prioritization' state that needs reprioritization. The third column shows that low priority is assigned to one bug report. The fourth column shows the status of the bug report is updated to reprioritized status, whereas high priority is assigned to it which is displayed in the fifth column.

3) *'Reprioritized' Status:* Table 1 displays that five bug reports with reprioritized status moved to the 'process of prioritization' state that needs reprioritization and has different priority values. The third column shows that three bug report is already assigned priority once among which one is now at the highest priority, one is at the medium priority and one is at the high priority, whereas two bug reports are already assigned priority more than once which is now at medium priority. The fourth and fifth columns show that these bug reports are reprioritized and assigned new priority values. Fig 3 shows that a new priority value is assigned to three bug reports, the priority value is left empty in four bug reports, while the priority value is updated in four bug reports. The fourth column of Table 1 displays the new statuses of the four bug reports that have empty priority values, showing that one bug report has been given the

https://doi.org/10.31436/ijpcc.v9i2.398

status of "unprioritized," two bug reports have been given the status of "cannot be prioritized," and one bug report has been given the status of "deprioritized."

V. IMPLICATIONS OF THE STUDY

The implications of the study to both theory and practice are analyzed which are discussed in this section. The suggested workflow is useful as it contributes to theoretical knowledge as well as is valuable for the software industry. From theoretical aspects, it characterizes the workflow states and statuses to demonstrate the phenomena of bug prioritization, which can be further input for future research in this area. For example, different bug prioritization tasks can be explored that comprise various events that may take place during the prioritization of bug reports and actions that are taken as a result.

A. *Collaboration between different lifecycle states of bug reports and prioritization states*

The literature models different workflows for handling bug reports. In Fig. 1, the lifecycle of a bug report and transitions into various bug-handling states is illustrated, while the arrow directed toward different states shows the rationale behind their transitions[11]. In this study, it is examined from various bug reports that several bug triaging and prioritization are performed after the bug is validated, assigned, resolved, verified, closed, and reopened. However, there is a research gap in characterizing the workflow for prioritization. Therefore, the proposed workflow for prioritization can be integrated with the various existing bug-handling models.

Fig. 4 displays the lifecycle of bug reports and their interaction with prioritization states. It illustrates that bug reports either remain in the waiting-for-prioritization state or in-process-of-prioritization state during or after their reporting cycle, triaging cycle, resolving, closing, or reopening of bug reports. The numbers of bug reports shown in italics are hypothetical and are just shown for illustrating the life cycle of bug reports and their synchronization with prioritization workflow, for instance,

- twelve bug reports are newly reported and either remain in the waiting-for-prioritization state or in-process-of-prioritization state,
- eighteen bug reports are in the triaging backlog and either remain in the waiting-for-prioritization state or in-process-of-prioritization state,
- eight bug reports are resolved, and twenty-four bugs are closed, whereas the arrow directing towards Reopen state shows that some bugs are reopened after being resolved and closed,
- six bug reports are reopened bugs and the arrow directing from reopen towards triaging state shows that some bugs may again need triaging and prioritization.

The two-directional arrows between the bug reporting cycle and the prioritization states highlight the fact that some bug reports are in a waiting-for-prioritization, due to which their priority values are left empty. This fact is also evident from bug reports, which show that most bugs are not assigned a priority when they are reported. Some bug reports, on the other hand, are newly created and move to the 'process of prioritization' state during their reporting cycle, and the priority value is assigned to them. This phenomenon is also evident from several Atlassians' bug reports where many bugs are given low priority during the reporting cycle. In Fig. 4, a collaboration between triaging cycle and prioritization states is shown in which some bug reports are in a waiting-for-prioritization state because of either delay in triaging process or other bug handling constraints. However, bug reports that are triaged and have no constraints, can move in-process-of-prioritization state. Fig. 1 shows that some bug reports reopen after being verified or closed, while it is analyzed from the history of bug reports that some bugs reprioritize after they reopen. Thus, the bug reports that cannot be verified or closed after being fixed, can be moved again into reprioritization status where they either will be in a waiting-for-prioritization or in-process of-prioritization state.

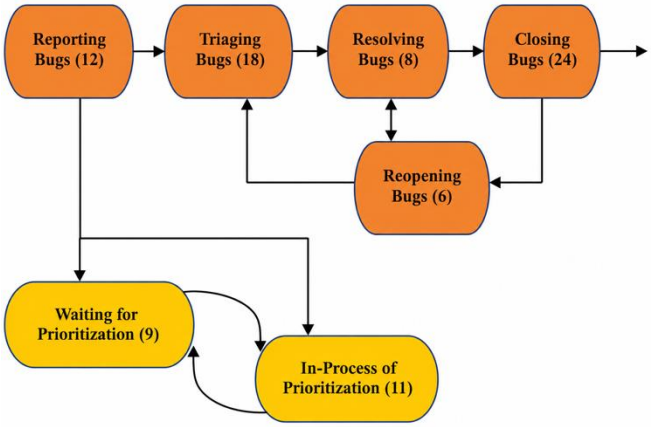

Fig. 4. Illustrates the Lifecycle of Bug Reports in and its Interaction with Prioritization States.

B. *Software Industrial Aspects*

From an industry perspective, this workflow is valuable to observe bug prioritization tasks that are being performed during the lifecycle of a bug. The software industry

recognizes the significance of sprint planning, daily stand-up, review, and retrospective meetings which are important ceremonies of the Scrum development process. This research demonstrates the advantages of the suggested workflow for the bug triager during the scrum meetings, to visualize the prioritization statuses for effectively handling bug reports. The bug triager can visualize the workflow of bug prioritization in sprint planning, daily stand-up and review meetings, and analyze the various prioritization statuses of bug reports. The bug triager can observe the workflow of bug prioritization and examine the prioritization statuses of various bug reports from the backlog of the previous iteration which are waiting for prioritization and analyze the rationale behind the prioritization process in a retrospective meeting to make plans for prioritizing the bug reports in the future.

The bug tracking and management system is used to handle various aspects of bug reports, for instance, creating news bug reports, and triaging, searching, and tracking bugs. Hence these bug-tracking systems are essential tools to handle bug reports and facilitate their resolution. The workflow model of prioritization tasks can be integrated with any bug tracking and management tool using Machine Learning approaches. Thus, the bug triager can visualize the ongoing prioritization tasks and customize his dashboard of the bug tracking tool which will give him insights and notify him with alerts. However, analyzing the adoption of this model and its integration with any bug-tracking tool will be part of future research.

C. *Decision-making Approaches*

The proposed workflow model of bug prioritization will be useful for a retrospective and prospective decision-making approach. For a retrospective decision-making approach, the bug triager can observe the prioritization workflow of past bug reports during the retrospective and review meetings, which is useful for lesson learning. He can find the bug reports waiting for prioritization and analyze the rationale behind the prioritization process, improving future processes. For the prospective decision-making approach, he can select the backlog of open bug reports and monitor the prioritization of bug reports during the sprint planning, and daily stand-up and analyze the various prioritization statuses.

VI. Conclusion and Future Direction

Literature reports many decision-related challenges in prioritizing bug reports due to various reasons including constraints in prioritizing bug reports, awaiting priority decisions, re-prioritization, and changeovers in priority decisions whereas the role of bug triager is critical in handling the bug prioritization tasks. The study analyzes and discusses the implications of the study for both theory and software industrial aspects. It proposes a descriptive workflow model of bug prioritization that characterizes the states of bug prioritization tasks, their statuses, and the transitions between them. The utilization of the suggested model for the software industry is also examined according to which this model can be integrated into any bug-tracking and management tool that will provide meaningful insights to bug triager into ongoing prioritization tasks and describes their synchronization with other bug-handling tasks. Further, this model can be used in different formats of Scrum and other triage meetings to assist him in decision-making in prioritizing bugs retrospectively and prospectively.

The paper examines the datasets of previous bug reports from Atlassian, Eclipse, Mozilla, Apache, and Red Hat projects and qualitatively analyzes the prioritization tasks from the previous events which reveal that bug reports contain events that are performed and actions that are taken during the bug prioritization tasks. The actions can be traced back by examining the outcome of priority decisions whereas past prioritization events can be traced back by analyzing the comments and the history of past bug reports which demonstrates the decision-making of bug prioritization. Some prioritization features are explored in this study to describe the workflow model of prioritization tasks, whereas other prioritization features can be explored in future work. Therefore, it can be said that bug reports are a valuable resource for outlining the workflow of bug prioritization tasks that can assist the bug triager in decision-making.

This model can be expanded in the future by looking at bug reports from other open-source projects and close-source projects to explore additional prioritization statuses that are not covered in this workflow model. Future research work can explore many events and the actions associated with the different prioritization tasks. A framework should be designed as part of future work that can adopt this model for industrial aspects in order to incorporate the suggested model. Therefore, empirical research should be carried out to apply the suggested model in different Scrum meeting formats and gather feedback.

Acknowledgement

The authors hereby acknowledge the review support offered by the IJPCC reviewers who took their time to study the manuscript and find it acceptable for publishing

Conflict of Interest

The authors declare that there is no conflict of Interest.

References

[1] J. D. Herbsleb and A. Mockus, “Formulation and Preliminary Test of an Empirical Theory of Coordination in Software Engineering,” 2003.

[2] J. Anvik, "Reducing the Effort of Bug Report Triage: Recommenders for Development-Oriented Decisions," vol. 20, no. 3, 2011, doi: 10.1145/2000791.2000794.
[3] R. P. Ghozali, H. Saputra, M. A. Nuriawan, Suharjito, D. N. Utama, and A. Nugroho, "Systematic Literature Review on Decision-Making of Requirement Engineering from Agile Software Development," *Procedia Comput Sci*, vol. 157, 2019, doi: 10.1016/j.procs.2019.08.167.
[4] T. M. Hesse, V. Lerche, M. Seiler, K. Knoess, and B. Paech, "Documented decision-making strategies and decision knowledge in open-source projects: An empirical study on Firefox issue reports," *Inf Softw Technol*, vol. 79, pp. 36–51, Nov. 2016, doi: 10.1016/j.infsof.2016.06.003.
[5] J. A. O. G. Cunha, H. P. Moura, and F. J. S. Vasconcellos, "Decision-making in Software Project Management: A Systematic Literature Review," *Procedia Comput Sci*, vol. 100, 2016, doi: 10.1016/j.procs.2016.09.255.
[6] F. Mendes, E. Mendes, N. Salleh, and M. Oivo, "Insights on the relationship between decision-making style and personality in software engineering," *Inf Softw Technol*, vol. 136, Aug. 2021, doi: 10.1016/j.infsof.2021.106586.
[7] N. Kaushik, M. Amoui, L. Tahvildari, W. Liu, and S. Li, "Defect Prioritization in the Software Industry: Challenges and Opportunities," in *2013 IEEE Sixth International Conference on Software Testing, Verification, and Validation*, IEEE, Mar. 2013. doi: 10.1109/ICST.2013.40.
[8] S. Akbarinasaji, C. Kavaklioglu, A. Başar, and A. Neal, "Partially observable Markov decision process to generate policies in software defect management," *Journal of Systems and Software*, vol. 163, May 2020, doi: 10.1016/j.jss.2020.110518.
[9] H. Jahanshahi, M. Cevik, J. Navas-Sú, A. Başar, and A. González-Torres, "Wayback Machine: A tool to capture the evolutionary behavior of the bug reports and their triage process in open-source software systems," *Journal of Systems and Software*, vol. 189, Jul. 2022, doi: 10.1016/j.jss.2022.111308.
[10] J. Xie, M. Zhou, and A. Mockus, "Impact of Triage: A Study of Mozilla and Gnome," in *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, IEEE, Oct. 2013. doi: 10.1109/ESEM.2013.62.
[11] J. Zhang, X. Y. Wang, D. Hao, B. Xie, L. Zhang, and H. Mei, "A survey on bug-report analysis," *Science China Information Sciences*, vol. 58, no. 2, pp. 2–3, 2015, doi: 10.1007/s11432-014-5241-2.
[12] H. Bani-Salameh, M. Sallam, and B. al Shboul, "A Deep-Learning-Based Bug Priority Prediction Using RNN-LSTM Neural," *e-Informatica Software Engineering Journal*, vol. 15, no. 1, 2021, doi: 10.37190/e-Inf210102.
[13] (2023) The Atlassian website, Work with issue workflows, [Online]. Available: https://support.atlassian.com/jira-cloud-administration/docs/work-with-issue-workflows/. [Accessed: 21-Feb-2023].
[14] (2023) The Atlassian website, Atlassian Cloud Bug Fix Policy, Atlassian Documentation. [Online]. Available: https://confluence.atlassian.com/support/atlassian-cloud-bug-fixing-policy-206865884.html. [Accessed: 01-Feb-2023].
[15] J. Uddin, R. Ghazali, M. M. Deris, R. Naseem, and H. Shah, "A survey on bug prioritization," *Artif Intell Rev*, vol. 47, no. 2, pp. 145–180, 2017, doi: 10.1007/s10462-016-9478-6.
[16] M. Gökçeoğlu and H. Sözer, "Automated defect prioritization based on defects resolved at various project periods," *Journal of Systems and Software*, vol. 179, Sep. 2021, doi: 10.1016/j.jss.2021.110993.
[17] M. Kumari and V. B. Singh, "An Improved Classifier Based on Entropy and Deep Learning for Bug Priority Prediction," 2020. doi: 10.1007/978-3-030-16657-1_53.
[18] (2023) Apache Software Foundation, Bug Priority Guidelines and Release Schedule for Apache NetBeans - [Online]. Available: https://cwiki.apache.org/confluence/display/NETBEANS/Release+Schedule. [Accessed: 21-Feb-2023].
[19] Eclipse Foundation (2023), Mozilla Website for Describing Bug fields. [Online]. Available: https://bugs.eclipse.org/bugs/page.cgi?id=fields.html. [Accessed: 18-Feb-2023].
[20] J. Zhang, X. Y. Wang, D. Hao, B. Xie, L. Zhang, and H. Mei, "A survey on bug-report analysis," *Science China Information Sciences*, vol. 58, no. 2, 2015, doi: 10.1007/s11432-014-5241-2.
[21] R. Almhana, T. Ferreira, M. Kessentini, and T. Sharma, "Understanding and Characterizing Changes in Bugs Priority: The Practitioners' Perceptive," in *2020 IEEE 20th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, IEEE, Sep. 2020. doi: 10.1109/SCAM51674.2020.00015.
[22] K. Punitha and S. Chitra, "Software defect prediction using software metrics - A survey," in *2013 International Conference on Information Communication and Embedded Systems (ICICES)*, IEEE, Feb. 2013. doi: 10.1109/ICICES.2013.6508369.
[23] R. K. Saha, "An Empirical Study of Long-Lived Bugs," pp. 144–153, 2014.
[24] Z. Abou Khalil, E. Constantinou, T. Mens, and L. Duchien, "On the impact of release policies on bug handling activity: A case study of Eclipse," *Journal of Systems and Software*, vol. 173, Mar. 2021, doi: 10.1016/j.jss.2020.110882.
[25] Z. Abou Khalil, E. Constantinou, T. Mens, L. Duchien, and C. Quinton, "A Longitudinal Analysis of Bug Handling Across Eclipse Releases," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, IEEE, Sep. 2019. doi: 10.1109/ICSME.2019.00010.
[26] J. Keung, "An Empirical Analysis of Reopened Bugs Based on Open-Source Projects," no. October, 2017, doi: 10.1145/2915970.2915986.
[27] Y. Feng, J. A. Jones, Z. Chen, and C. Fang, "Multi-objective test report prioritization using image understanding," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, New York, NY, USA: ACM, Aug. 2016, pp. 202–213. doi: 10.1145/2970276.2970367.
[28] Y. Tian, D. Lo, X. Xia, and C. Sun, "Automated prediction of bug report priority using multi-factor analysis," *Empir Softw Eng*, vol. 20, no. 5, pp. 1354–1383, 2015, doi: 10.1007/s10664-014-9331-y.
[29] R. Almhana and M. Kessentini, "Considering dependencies between bug reports to improve bugs triage," *Automated Software Engineering*, vol. 28, no. 1, May 2021, doi: 10.1007/s10515-020-00279-2.
[30] (2023) The Standish Group, The Standish Group. [Online]. Available: https://www.standishgroup.com/. [Accessed: 02-Feb-2023].
[31] K.-J. Stol and B. Fitzgerald, "The ABC of Software Engineering Research," *ACM Transactions on Software Engineering and Methodology*, vol. 27, no. 3, pp. 1–51, Jul. 2018, doi: 10.1145/3241743.
[32] Y. Noyori *et al.*, "What are Good Discussions Within Bug Report Comments for Shortening Bug Fixing Time?" in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*, IEEE, Jul. 2019, pp. 280–287. doi: 10.1109/QRS.2019.00044.
[33] A. J. Ko and P. K. Chilana, "Design, discussion, and dissent in open bug reports," in *Proceedings of the 2011 iConference*, New York, NY, USA: ACM, Feb. 2011, pp. 106–113. doi: 10.1145/1940761.1940776.
[34] (2023) The Atlassian website for Browse projects - Jira Bug Tracking System for Closed-Source Projects of Atlassian. [Online]. Available: https://jira.atlassian.com/projects. [Accessed: 01-Feb-2023].
[35] Eclipse Foundation Bug Reports (2023), Bugzilla Bug Tracking System to Browse Eclipse Bug Reports, [Online]. Available: https://bugs.eclipse.org/bugs/query.cgi. [Accessed: 21-Feb-2023].
[36] Red Hat Bug Reports (2023), Jira Bug Tracking System to Browse Red Hat Bug Reports. [Online]. Available: https://issues.redhat.com/projects/. [Accessed: 01-Feb-2023].
[37] Mozilla Bug Reports (2023), Bugzilla Bug Tracking System to Access Mozilla Bug Reports. [Online]. Available: https://bugzilla.mozilla.org/. [Accessed: 21-Feb-2023].
[38] Bugzilla Guide (2023), The Firefox Source Docs documentation for Bug Handling and Triaging in Bugzilla. [Online]. Available: https://firefox-source-docs.mozilla.org/bug-mgmt/policies/triage-bugzilla.html. [Accessed: 01-Feb-2023].

[39] Bug Handling Guideline (2023), Triage and Priority Guide for Firefox Bugs on Mozilla Central. [Online]. Available: https://searchfox.org/mozilla-central/source/docs/bug-mgmt/. [Accessed: 01-Feb-2023].
[40] Priority definitions (2023), Firefox Source Docs documentation for Bug Handling Policy. [Online]. Available: https://firefox-source-docs.mozilla.org/bug-mgmt/guides/priority.html. [Accessed: 01-Feb-2023].
[41] Bug Handling Guideline (2023), What are issue statuses, priorities, and resolutions? Published by Atlassian Support. [Online]. Available: https://support.atlassian.com/jira-cloud-administration/docs/what-are-issue-statuses-priorities-and-resolutions/. [Accessed: 01-Feb-2023].
[42] Priority definitions (2023), What are the Priority Levels in Jira Service Management? For Jira Service Management Cloud Published by Atlassian Support. [Online]. Available: https://support.atlassian.com/jira-service-management-cloud/docs/what-are-priority-levels-in-jira-service-management/. [Accessed: 03-Feb-2023].
[43] (2023) Apache Software Foundation, Bug Priority Guidelines, and Release Schedule for Apache NetBeans - [Online]. Available: https://cwiki.apache.org/confluence/display/NETBEANS/Release+Schedule. [Accessed: 21-Feb-2023].
[44] (2023) The Atlassian website, Application Lifecycle Development and Collaboration Tools for Software, IT, and Business Teams. [Online]. Available: https://www.atlassian.com/. [Accessed: 01-Feb-2023].

EndNotes

[i] (a) Bug Report Resolved with Workaround Status (b) Bug Report Reprioritized Many Times: *Bitbucket Cloud: BCLOUD-19548, JIRA Bug Tracking System*. [Online]. Available: https://jira.atlassian.com/browse/BCLOUD-19548 [Accessed: 02-Feb-2023].

[ii] Bug Report with Empty Priority Value: *Jira Work Management Cloud: JWMCLOUD-105, JIRA Bug Tracking System*. [Online]. Available: https://jira.atlassian.com/browse/JWMCLOUD-105. [Accessed: 03-Feb-2023].

[iii] Priority of Bug Report is Left Empty But Later on Priority is Assigned: *Server Deployments and Scale*: SCALE-20, *JIRA Bug Tracking System*. [Online]. Available: https://jira.atlassian.com/browse/SCALE-20 [Accessed: 03-Feb-2023].

[iv] Priority is Assigned to Bug Report Once: *Sourcetree for Windows: SRCTREEWIN-13863, JIRA Bug Tracking System*. [Online]. Available: https://jira.atlassian.com/browse/SRCTREEWIN-13863 [Accessed: 03-Feb-2023].

[v] Priority of Bug Report is Changed: *Sourcetree for Windows: SRCTREEWIN-66547, JIRA Bug Tracking System*. [Online]. Available: https://jira.atlassian.com/browse/CONFSERVER-66547 [Accessed: 03-Feb-2023].

[vi] Bug Report is Deprioritized: *Jira Service Management Cloud: JSDCLOUD-8330, JIRA Bug Tracking System.* [Online]. Available: https://jira.atlassian.com/browse/JSDCLOUD-8330 [Accessed: 03-Feb-2023].

[vii] Following Bug Reports are accessible through JIRA Bug Tracking System. *Browser using* https://jira.atlassian.com/browse/ [*BCLOUD-19548; CONFCLOUD-73781; CONFSERVER-66547: JRACLOUD-77460: BAM-21778; JRASERVER-73435; CONFSERVER-79118; SRCTREEWIN-13863; OPSGENIE-396*] - [Accessed: 03-Feb-2023].

[viii] Following Bug Reports are accessible through Bugzilla Bug Tracking System. *Browser using* https://bugs.eclipse.org/bugs/ [*Bug 121995; Bug 178923, Bug 532097, Bug 121995, Bug 575890, Bug 551483*] - [Accessed: 03-Feb-2023].