Understanding and Characterizing Changes in Bugs Priority: The Practitioners' Perceptive

Rafi Almhana
Dept. of Computer Science
University of Michigan
ralmhana@umich.edu

Thiago Ferreira
Dept. of Computer Science
University of Michigan
thiagod@umich.edu

Marouane Kessentini
Dept. Computer Science
University of Michigan
marouane@umich.edu

Tushar Sharma
Research Scientist
Siemens Corporate Technology
tusharsharma@ieee.org

Abstract*—Assigning appropriate priority to bugs is critical for timely addressing important software maintenance issues. An underlying aspect is the effectiveness of assigning priorities: if the priorities of a fair number of bugs are changed, it indicates delays in fixing critical bugs. There has been little prior work on understanding the dynamics of changing bug priorities. In this paper, we performed an empirical study to observe and understand the changes in bugs' priority to build a 3-W model on *Why* and *When* bug priorities change, and *Who* performs the change. We conducted interviews and a survey with practitioners as well as performed a quantitative analysis containing 225,000 bug reports, developers' comments, and source code changes from 24 open-source systems. The interviews with 11 developers from industry aim to establish an initial model to characterize the changes in bugs priority. The survey with an additional 38 developers was to understand their experience in why and when bug priorities change, and who performs the change. Then, we conducted a manual inspection of the collected data on open-source projects to compare our final bugs priority change model with changes identified in practice. Our quantitative results confirmed the outcomes of our interviews and surveys. For instance, we observed frequent changes in bug priorities and their impact on delaying critical bug fixes especially just before shipping a new release. Our findings can enable 1) researchers to build automated tools for checking and validating requests for bug priority changes, 2) practitioners to use a standard format in documenting and approving bug priority changes, and 3) educators to teach the better management of bug priorities.

***Index Terms*—Bug Report, Software Bugs, Bugs Priority, Empirical Study**

I. Introduction

A bug is a software defect that causes abnormal or erroneous behavior according to functional or non-functional requirements (such as security and performance) [1]–[5]. Different bugs impact the software system differently based on the degree of severity associated with each bug [6]–[8]. Therefore, it is critical to efficiently manage bugs priority [9]–[13]. In this context, the effectiveness of assigning priorities becomes important—if priorities of a fair number of bugs are changed, it indicates delays in fixing critical bugs [9]–[13].

Several studies explored methods to predict bugs priority in software systems [9], [10], [13]–[17]. To the best of our knowledge, there has been little prior work on understanding the dynamics of changing bug priorities. Understanding changes in bugs priority can help us to quickly fix severe bugs and avoid delays, identify areas that need tool support for automated validation of bug priority change requests, and better documentation of these changes. The goal of this paper is to characterize the overall change process of bugs' priority.

We advocate that a critical and fundamental step in providing efficient support for managers and developers to enable them to validate bugs priority change is to understand the bugs priority dynamics; it involves discover and characterize *Why* and *When* bug priorities change, and *Who* performs the change. Thus, the primary goal of this paper is to observe and understand the changes in bugs priority to build a *3-W (Why, When, and Who) model*. In this pursuit, we used two complementary methods in our study. As a first step, we discovered insights about the rationale of bug priority changes, their frequency, and when/why these changes were observed by interviewing 11 software developers, managers, and executives from industry partners as part of a funded project. We established an initial model for characterizing changes in bug priority as an outcome of this first step. In a second step, we performed a survey with an additional 38 developers to enquire about their experiences with finding, validating, and documenting the changes in bugs priority. During these two steps, we answered the following three research questions (RQ).

- **RQ1** *Why does priority of a bug change?*
- **RQ2** *Who changes the priority of a bug?*
- **RQ3** *When does the priority of a bug change?*

We propose a 3-W bugs priority change model obtained from the interviews and the survey. We have also conducted a manual inspection of more than 225,000 bugs reports, developers' comments, and source code changes from 24 open-source systems to compare the final 3-W bugs priority changes model with actual bugs priority changes extracted from open-source projects to answer the research questions.

We have also compared the experience of developers in finding, validating, and documenting bugs priority changes with samples of actual bugs to reveal areas for improvement. We found that developers indeed change the priority of bugs multiple times. Table I shows an example of a bug report with its log of activities that exhibit the number of times developers change the bug priority both before and after the release. Note that the priority field ranges between P1 (the highest) to P5 (the lowest).

Who	When	Before	After	Attributes Changed
User1	2008-05-11	P3	P5	-
User2	2008-05-20	P5	P3	Target Milestone
User3	2008-08-03	P3	P5	-
User2	2008-08-21	P5	P3	Status & Resolution
User1	2009-05-18	P3	P5	Target Milestone
User2	2009-05-26	P5	P3	Target Milestone

TABLE I
BUG REPORT# 221310 FROM BIRT PROJECT TO SHOW THE ACTIVITIES THAT HAPPENED ON THE PRIORITY OF THIS BUG REPORT

Our 3-W model suggests the following rationale of changes in bugs priority: 1) lack of time to complete the task, 2) the category of the bug such as security-related bugs, functionality related bugs, or a user interface related issues, 3) the type or the domain of project such as desktop application or web-based application, a plugin tool or standalone program, 3) dependencies with other bugs, 4) lack of understanding or misunderstanding the bug report, and 5) accidental changes by mistake. Also, we found that some developers do not follow "ethical" practices while changing bugs' priority. For instance, they may reduce bugs priority just to bypass quality gates and to release code quickly.

Our findings can enable 1) researchers to automatically validate bugs priority changes and understanding their rationale, 2) educators to teach and emphasize the management of bugs and prioritize software maintenance activities, and 3) practitioners to use a standard format for documenting and discussing changes in bugs priority. Though we identified a set of essential components of changes in bugs priority, adoption of the components remains context-dependent in practice. Using our model, software development teams can design their organization-specific guidelines to include or exclude the proposed components for validating changes in bugs priority.

The primary contributions of this paper are as follows:

- A detailed model to understand changes in bugs priority based on practitioners' perspective.
- A study of the experiences of software developers requiring, finding, and documenting changes in bug reports.
- Investigation of areas for improvement in current practices of changing bugs report.

Replication Package. All material and data of the bugs report used in our study as well as the developers' anonymized answers are available in our replication package [18].

The remainder of this paper is as follows: Section II describes the design of our empirical study including the research questions. The results are described in Section III and Section IV discusses the implications of our study. Section VI is dedicated to related work. Finally, concluding remarks and future work are provided in Section VII.

II. STUDY DESIGN

Our study aims to understand the rationale of changes in bugs priority and therefore will guide the automation of validating change requests by developers. As described in Figure 1, we first used unstructured interviews with 11 developers, managers, and executives from industry partners, as part of a funded project, to discover the rationale of changes in bugs priority and thereby, design an initial model. These interviews allowed for rich conversations and insights. These brainstorming sessions helped us to establish the initial thoughts in characterizing the changes in bugs priority to discover the reasons behind changes the priority, the time of changing the priority, and the individuals who perform the changes. Then, we extended the obtained initial documentation model with a larger number of 38 practitioners using a survey. The practitioners answered our questions about their experiences in performing, finding, and documenting these bug priority changes. Finally, we conducted a quantitative validation to compare the outcomes of the interviews and survey with actual changes in bugs priority extracted from 24 open-source systems. The use of these mixed methods has been widely employed by several other studies of software developers [19]–[22].

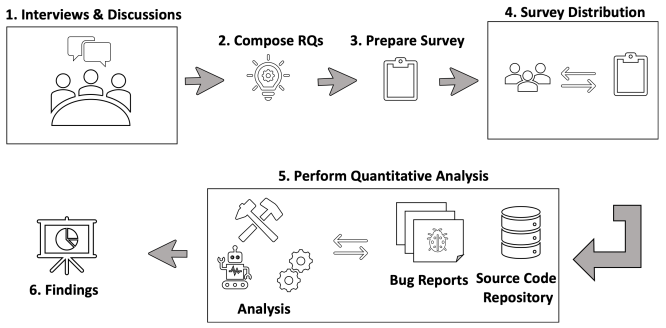

Fig. 1. Study design

A. Research Questions

We defined the following main three research questions.

RQ1 *Why does priority of a bug change?*
This first research question aims to gather exhaustive possible reasons behind changing the priority of bugs. An understanding of the rationale will help a) practitioners to better document priority changes of bugs, b) researchers to build tools for automatically checking the requests of changing bug's priority.

RQ2 *Who changes the priority of a bug?*
This research question aims to identify the main stakeholders who change the priority of bugs and their role in the team or in the project (e.g. tester, manager, and the owner of a bug). The outcome of this research question can help us to understand the needs for changing the priority based on the role of the people who make the change.

RQ3 *When does the priority of a bug change?*
In general there are no restrictions on bug priority changes. This research question aims to study the possible correlations between the dates of priority changes and bug's creation, release date, or the date of assigning a developer on the bug. The outcomes of this research question may inform us about temporal

patterns to change bugs priority in suspicious time such as a new release deadline.

To summarize the outcome of this contribution, we aim to compare the outcomes of both the interviews and the survey with actual changes in bugs priority extracted from open-source systems. Observations from the comparison between the practitioners' needs and actual priority changes found through a quantitative analysis could lead us to the areas of improvement for researchers and practitioners to address them.

B. Phase 1: Interviews with developers to design an initial model for bug priority changes

1) Interview setup: The goal of this first phase of our research was to build a model to characterize the dynamics related to bug's priority, to understand why and when they happen, and to identify the individuals' role who make the change or get impacted by the change.

Prior to starting the interview sessions, we performed an in-depth analysis of previous studies that are related to the bug's priority changes or bug's priority in general. We list them in Table VI and discuss them in detail in the related work section (Section VI). The aim of this in-depth analysis is to understand the current state-of-the-art and to gather insights on *why* priorities change, *when* do they change, and *who* makes the change. The majority of the existing literature focuses on bug's priority prediction and, in some cases, recommending developers to fix bug reports. To the best of our knowledge, none of them analyzed the historical changes in priority levels to build a model that can validate the accuracy of changes in priority made by developers. Typically, textual and temporal components of a bug report, the author's information with historical bug reports are the four most-used components or metrics for predicting bug's priority. We considered these observations from the current literature about bugs priority in our unstructured interviews with developers, managers, and executives.

2) Participants Selection: We advertised our study in mailing lists that covered developers from many industrial partners, including those who collaborated with us in the past, to validate our approach in characterizing the overall change process of bugs priority. We interviewed 11 participants, after eliminating three other practitioners because they were not able to attend the whole interview session and provided very limited and quick feedback in the discussion.

3) Interview with selected volunteers: We started the face-to-face interviews by providing examples of several bug-tracking systems such as Jira[1] and BugZilla[2]. We presented the process of logging a bug report and setting the bug's priority. Then, we showed the mechanism of changing the priority in those bug-tracking systems by using several examples. We also demonstrated tracking a change or restricting certain users from performing a change at a certain period of time. We showed the interviewees some examples of bug reports in which the priority has changed along with related information such as the time, who did the change, and relevant comments about the priority changes. The exhibited examples set the context and scope of our discussion with the participants.

Then, we asked them to tell us some real-world situations when they needed to carefully check a change in bugs priority. These steps helped stimulate the practitioners' memories as well. As a next step, we asked them to think about an exhaustive set of reasons to change bugs priority after describing their experiences in changing, finding, and documenting these bugs priority so that it does not slow down addressing critical bugs while respecting deadlines and dealing with available resources. Based on these unstructured interviews, we built an initial 3-W model for bugs priority changes. We validated the initial model later with more participants via surveys and a quantitative validation using data collected from open-source systems.

C. Phase 2: Survey about the bug priority change model

After deriving a bug priority change model from the interviews, we designed a survey to understand their experiences and opinions about changes in bug's priority. The survey also aimed to elicit the individual roles who are responsible for changing the bug's priority and when they make these changes along with the reasons that lead to increasing or decreasing the priority.

We distributed our 12-question survey to multiple software engineering groups, software testing groups, and software maintenance groups on several social media channels and a list of private emails of researchers with a related background in software engineering and testing. We used the snowball sampling of the interviews for our survey by reaching out to our industry partners and asking them to advertise it to their contacts.

Table II shows a list of questions of our survey which was carried out using Qualtrics[3]. The table shows the connections between our research questions and the questions of the survey. In the first section, we prepared three questions to capture demographic information about the background of the participants (number of years in the field, the level of education, and their current role/occupation).

Our intention in the second section of the survey is to gather information about the rationale of changing the priority of the bug reports. So we asked our participants to choose from a list of possible reasons collected from the interviews from Phase 1 or propose a new reason to be added to our findings. Furthermore, we asked the participants whether they leave a comment about the rationale when they change the bug's priority and whether they change other attributes/fields in bug reports while they are changing the priority. The goal of this question is to identify a possible correlation between bug's priority and other attributes, as well as to discover other reasons for changing the bug's priority. Finally, we asked the participants whether they consider the priority to rank their list of tasks and organize their schedules around it.

[1]https://www.atlassian.com/software/jira

[2]https://www.bugzilla.org

[3]https://www.qualtrics.com

#	Survey question	Research Question / Intention
1	What is your highest level of education related to Computer Science?	Background Information
2	What is your current occupation?	
3	How many years have you worked with software?	
4	Why do you change bug's priority?	Why does the priority of the bug change and when?
5	Do you leave a comment about the rationale when you change bug's priority?	
6	What are the attributes that you change when you change bug's priority?	
7	Do you consider bug's priority to manage your workload?	
8	When do you change, in general, the priority of bugs?	How does the priority of the bug change?
9	Is there any approval process or poll/vote mechanism to check your change of bug's priority?	
10	Who does change bug's priority and when?	Who does change the priority of the bug?
11	Do you change the priority of bugs?	
12	How many times do you change, in general, the priority per bug?	

TABLE II
THE TRACEABILITY BETWEEN OUR RESEARCH QUESTIONS AND SURVEY QUESTIONS

The third section of the survey focused on when bugs priority are changed compared to other temporal events such as the date of the bug's creation, the date of bug's resolution, the date on which developers are assigned to the bug, and the date of the deployment or releasing a new version of the software. Also, we asked our participants if they follow an approval or voting process or any other mechanisms before changing the priority of a bug.

In the last section, we asked the participants whether they change the priority themselves and how often they change it. We were also interested to know the group of people who changes the priority including their profiles, current occupations, and their role in the project. Furthermore, we asked the frequency of changing a bug's priority in general.

We discarded three responses based on the short time that they spent to take the survey (less than 5 minutes). We considered 38 survey responses (after discarding three responses); the participants took 8-12 minutes to finish the survey.

D. Phase 3: Quantitative analysis

Table III shows the list of open-source systems that we analyzed in this study. The table shows the number of bugs and comments in bug reports for each project. We used Bugzilla API to fetch all bug reports and the list of related comments along with changes on any attributes of the bug reports. Furthermore, we have collected all the commits belonging to all the considered projects on their GitHub repositories. Overall, we collected a total of $225,534$ bug reports belonging to 24 projects. Those bug reports included more than 1.79 million changes in bug reports and 1.35 million developers' comments related to these changes. In addition, these projects have $302,760$ GitHub commits with more than $15,000$ releases.

A bug report from Bugzilla is composed of several attributes listed below.

- *Creator:* The login name of the person who filed the bug (the reporter)
- *Creation time:* When the bug was created
- *Keywords:* Each keyword mentioned on the bug
- *Severity:* The current severity of the bug

Project Name	#Bugs	#Comments	#Commits	#Versions
Eclipse Platform	118,309	761,180	8,190	5,201
BIRT	23,270	111,178	41,290	437
AspectJ	3,049	16,666	46,910	112
JDT	59,780	367,172	24,222	5,723
Buildship	482	2,304	3,314	37
JGit	1,351	7,254	7,655	174
openj9	9	36	41,948	42
PDT	6,062	31,580	9,490	487
TCF	1,238	5,896	4,830	28
SW360	2	5	512	12
Antenna	2	4	773	28
Hawkbit	2	2	2,267	21
Californium	61	212	1,927	32
Kapua	4	13	4,078	24
GEF	3,167	14,612	5,113	7
Ditto	2	2	4,343	19
Vorto	160	467	1,987	31
Titan	563	1,857	8,237	15
Jetty	3,813	16,661	66,427	340
BPEL	386	1,341	1,047	9
e4	3,788	21,057	993	1,122
Milo	1	1	759	24
Che	31	96	8,372	163
OMR	2	12	8,076	1

TABLE III
LIST OF STUDIED PROJECTS ALONG WITH THE NUMBER OF BUGS, COMMENTS, GITHUB COMMITS, AND RELEASES

- *Resolution:* The current resolution of the bug, or an empty string if the bug is open
- *Summary:* A summary of the bug
- *Status:* The current status of the bug
- *Priority:* The current priority of the bug
- *Assignee:* The login name of the user to whom the bug is currently assigned

We collected every change that happened on any attribute on the bug reports and extracted the following fields corresponding to each change.

- *Field Name:* The changed field name
- *Time:* The date-time stamp on when the change happened
- *Old Value:* The value of the field before it was changed
- *New Value:* The value of the field after it was changed
- *Person:* The login name of the user who changed the value of the field

The collected data is used to answer our research questions and find potential gaps between the developers' perception to changes of bugs priority and actual ones observed in the practice.

III. Results

In this section, we answer the research questions by combining the results of the survey and the outcomes of the quantitative analysis on several open-source projects. Figure 2 represents the demographic information of the 38 participants of the survey. Their current occupation ranges between researcher, senior and junior software engineers, and QA/testers.

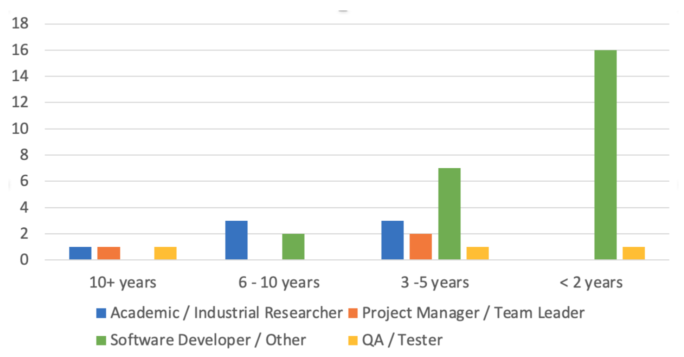

Fig. 2. Current occupation and years of experience

A. *RQ1: Why does the priority of a bug change?*

Figure 3 shows that the majority of our participants agree that they consider the bug's priority to manage their workload. Eight participants mentioned that they use the priority sometimes, and only 2 participants highlighted that they rarely look at the priority to manage their tasks. None of the participants expressed that they never consider the bug's priority. Our quantitative analysis shows that 100% of collected bugs have a priority assigned to them which proves the importance of bug's priority in managing bug reports.

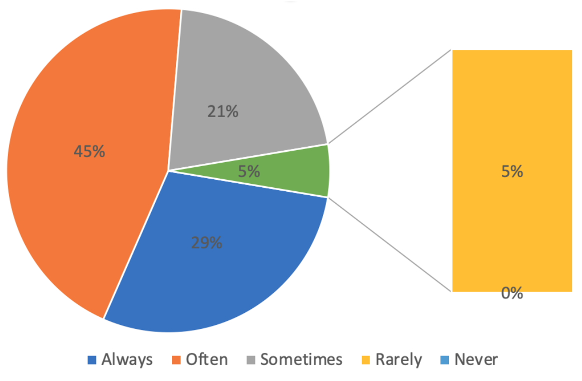

Fig. 3. Responses to the survey question: Do you consider bug's priority to manage your workload?

Figure 4 shows that only 2 participants said that they never changed a bug's priority where the majority of them agree to the fact that they do need to change the bug's priority frequently. Furthermore, Figure 5 shows that most of the developers agree to change the priority once, twice, or more in general. Our quantitative analysis shows that more than 10% of collected bug reports have their priority changed at least one time, and over 86% of those bugs had their priority changed at least two times.

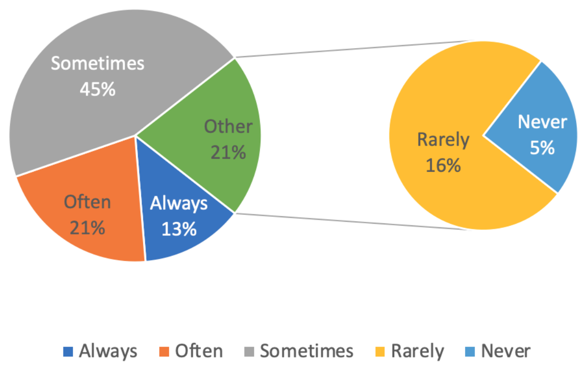

Fig. 4. Responses to the survey question: Do you change the priority of bugs?

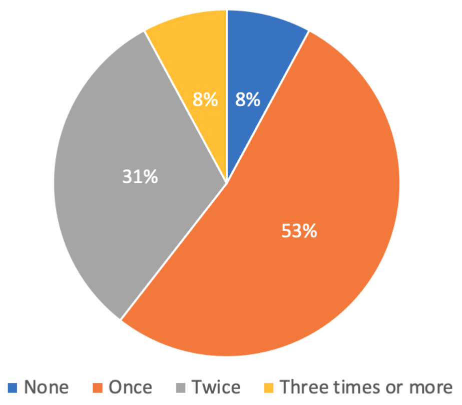

Fig. 5. Responses to the survey question: How many times do you change, in general, the priority per bug?

Figure 6 shows the distribution of the votes among several possible reasons for changing the bug's priority. From the response, we observe that the dependencies among the bugs are the biggest reason for changing a bug's priority. Lack of time and high workload are the next biggest reasons of priority change with 19 and 17 votes, respectively. 13 participants said that the initial priority value is always not accurate and hence they have to correct it to match the reality of the issue. We also found that 10 participants agree that the domain of the project or the category of the bug report may affect the clarity of the bug report which makes a big difference whether changing the priority is needed or not. Other participants confirmed that changing priority by mistake does happen but it is not frequent.

Our quantitative analysis shows that 14% of the bug reports with priority changes have their priority changed twice from high to low and then from low to high. This outcome shows

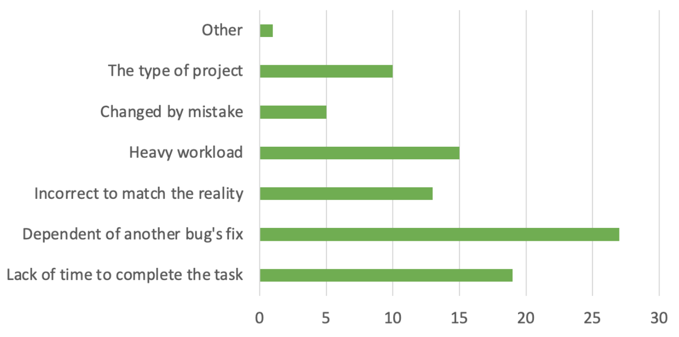

Fig. 6. Responses to the survey question: Why do you change bug's priority?

that bug's assignee may change the priority once it gets assigned to them to reduce the perceived urgency of the bug and to delay the delivery of the solution to the end-users. While it may not be among the best practices to change priorities to reduce the workload, the quantitative analysis of the bug reports shows that this aspect is common when bugs priority is lowered.

We found that developers may change the priority based on their present workload to avoid any interruptions in the current sprint or simply due to lack of time. In fact, we looked into developers' comments in the collected bug reports and we found some comments complaining about workload or lack of time. We also found comments mentioning dependency with other bugs/issues from other teams/projects. Examples #1, #3, and #7 in Table IV show the details of those comments.

Another finding is that developers might not see the necessity of assigned high priority or they might see the urgency in the assigned high priority of a bug. Examples #4 and #8 of Table IV show that developers change the priority to better match the reality.

We also noticed that when the initial bug's description is not clear or not detailed enough, developers tend to introduce priority changes after some investigation based on the analysis of the comments in bug reports. We also found that some projects do not have any priority changes, the reasons could be the lack of using the bug's priority to prioritize their workload or the bug's description is clear enough to make an accurate estimation of the priority of the bug.

Regarding the documentation of bug priority changes, most of the participants in our survey confirmed that they tend to leave a comment explaining the change; only nine responses said that they rarely or never document these changes. However, we found that developers are not following specific guidelines in documenting these changes and they are documented in an informal way.

We also noticed that the description, status, and severity attributes are the most common attributes that get changed upon changing the priority. Based on the participants' responses, only five of them mentioned that they have to pass through an approval process before changing the priority. The remaining participants agree that there is no process that restricts developers from changing the priority at any time and for any bug report. The lack of this process to check and validate priority changes can be a reason for some suspicious changes due to a high workload, release deadlines, showing less impact of a bug created by a developer, etc.

We found in our quantitative analysis that the status, resolution, assignee, and target milestone are the fields that get changed when the priority attribute is changed. In 27% of priority change activities, we noticed that the assignee field changed. It indicates that assigning the right developer to fix the bug is important to get the right estimation of the priority. We also discovered that 37% of priority change activities where priority changed at the same time with status and target milestone fields. We consider those activities as suspicious because they tend to delay the deliverable of the project by lowering the priority due to an upcoming release deadline. To confirm this observation, we investigated and found that 44% of these bug reports get back to the same level of priority after a short period of time that can be associated with a release deadline.

By exploring some comments among developers in several of the bug reports in open-source projects, we found that many discussions are about raising or lowering priority or simply asking for clarity about the bug's description itself. This observation may confirm that there are some discussions happening before changing the priority between developers which can contribute to better explaining the reasons behind the changes. On the other hand, we found, by looking at the comments of developers, that bug's priority gets changed by accident as described in example #5 of Table IV. These accidental changes confirm again the importance of adding a mechanism in the pipeline of localizing and fixing bugs to validate the priority changes.

Key findings: The bug priority changes for the following reasons:

- The dependency of another bug's fix
- Incorrect priority
- Type/Domain of project
- Category of the bug report
- Lack of time / Heavy workload / Tight schedule
- Accident
- Hot-fix request
- Business requirements

Key findings: Bug tracking systems track the changes of the bug's priority but they lack the ability to document the reasons of the change.

B. *RQ2: Who does change priority of a bug?*

Figure 7 shows that most of the priority changes are carried out by developers, team leaders, or project managers. Out of 38, 31 responses show that developers change the bug's priority making them playing the biggest role in the change. It is not surprising because they are the ones who work on localizing and fixing the bugs. Other responses, 22 and 26, suggest that team leaders or project managers/owners also

Example#	Bug ID	Project	Comment
1	150807, 151061	JDT	Downgrading priority since we will probably not have time for this.
2	33897, 34076, 35075	Eclipse Platform	There are no plans for the UI team to work on this defect until higher priority items are addressed.
3	217891, 233481	JDT	Ownership has changed for the javadoc comments bugs, but I surely will not have enough time to fix your bug during the 3.5 development process,
4	151612, 170140	Eclipse Platform	Lowering priority to better match reality.
5	75829	Eclipse Platform	My apology, I inadvertently changed the priority when I changed the severity, I'm changing it back now.
6	50888, 52115	JDT	Resetting priority to P3. Will be reassessed for the next release.
7	191927	BIRT	Firefox hasn't fix this bug yet. Set the priority to p5.
8	21652	Eclipse Platform	Lowering priority to P2 (P1 means that this is a "stop-ship" bug report)

TABLE IV
DEVELOPERS' COMMENTS FROM SEVERAL OPEN SOURCE SOFTWARE REGARDING THE CHANGES ON THE PRIORITY OF THE BUG.

change the priority to rush certain software features or meet future expectations or milestones.

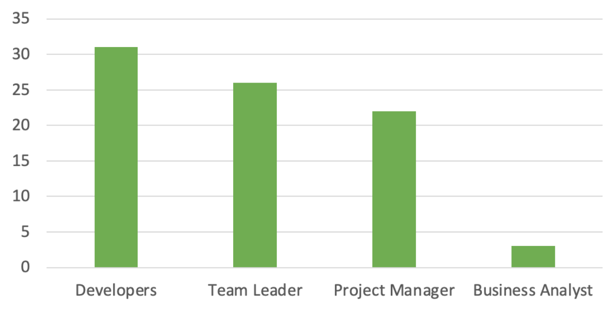

Fig. 7. Responses to the survey question: Who does change priority of a bug?

In the quantitative analysis, we found that 28% of bug reports with priority changes have been changed by their assignee. Also, there is 19% of bug reports with priority changes where the priority is changed by their reporter or creator of the bug. We assume that the rest of the priority changes were performed by the project leader, business analyst, or another developer. The lack of information to describe the role or the profile of each of the team members is also an issue in the open-source software and bug tracking systems. We note that there is no mechanism or approval process by which the project's stakeholders can request the change and apply the change to the bug report.

Key findings: Most priority changes are made by project's stakeholders including developers, team leaders, and project managers.
Key findings: Bug tracking systems track the individuals who make the changes on the priority but they lack the ability to

- Restrict certain individuals to change the priority since they may not have the required knowledge and expertise of the addressed bug.
- Capture profile or role information about project's stakeholders.

C. *RQ3: When does priority of a bug change?*

Figure 8 shows that the most of the priority changes happen between the date in which the bug gets assigned to the developer and the date before releasing a new version of the software. Some answers claim that it is necessary to change the priority by the project manager who is assigning them because each developer has their own tasks and therefore the priority should be tuned based on the type and the number of tasks assigned to the developer. Another explanation comes from best practices of agile methodology where they are advised to change the priority after scrum planning sessions with the development and the business teams.

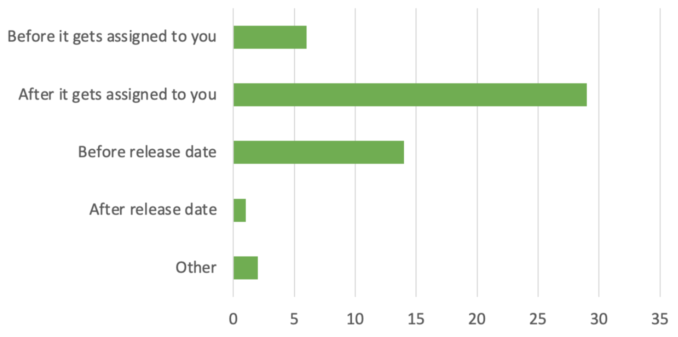

Fig. 8. When do you change, in general, the priority of bugs?

In the quantitative analysis on bug reports from open-source projects, we found that a bug's priority changes 2 times on an average with a minimum of one time and a maximum of 18 times. Interestingly, we found that a bug's priority changes in a relatively short period of time with an average 15 days from releasing a new version of the software. More surprisingly, we identified that there are about 44% of the bugs where the priority has been changed twice—the first change is to lower the priority and the second one is to reset it to the original priority of the bug. Thus, the bug's priority changes twice, once after the developer or assignee reviews it, another time comes in a short period after the release date. This bad practice should be avoided since developers may tend to ship their code quickly with bugs in that some of them could be critical.

🔑 **Key findings:** Most priority changes happen between the date when the bug gets assigned to the developer and the date just before releasing new version of the software.

🔑 **Key findings:** Bug tracking systems track the timestamp when changes have happened but they lack the ability to

- 💎 lock the priority so that it cannot be changed after certain time or restrict certain individuals to change the priority after it passes a period of time or when it reaches certain status.
- 💎 capture the project's milestones along with their due date.

To summarize, we collected all our findings in the survey and the results of the quantitative analysis. The outcome is presented in Table V as a 3-W model to show the consolidated findings from both the survey and the quantitative analysis. We also present a list of recommendations for any future improvements by industrial and research communities. Therefore, we classified our findings and recommendations into three different groups to answer each of our research questions.

The first group is designed to answer the first research question as to *why does the priority change*. We found that the dependency of another bug could be one of the reasons besides some other obstacles that programmers encounter such as lack of time, heavy workload, or a tight schedule. Also, we identified some cases where developers set incorrect priority for bug reports in which they are not knowledgeable enough to do so. Another reason could be related to business requirements where they have to prioritize some bugs over other bugs considering the severity of the bugs. Likewise, the priority may need to be changed if the bug report is considered to be a hot-fix request and thus it needs to be tackled by the development team right away. According to our quantitative analysis, we noticed that some projects do not have any changes in priority due to the clarity of the description of the bug report, the size of the project, or the size of the development team. Similarly, we found that the changes in the priority of some bug reports are different than the priority of other bug reports in the same project due to the category of the bug report such as security, functionality, or user interface related bugs. Lastly, we noticed some cases where the developers changed the priority by accident. As a recommendation, due to a lack of documentation upon changing the priority, we recommend that all bug tracking systems should have the ability to track the changes in priority with appropriate documentation noted by the user who makes the change. Capturing such information will help in the priority prediction process to accurately predicate the priority of bug reports and therefore improve the bug triage process. Also, we recommend adopting a standard in bug report documentation to avoid ambiguity in the process.

The next group is focused to answer the second research question as to *who changes the priority*. According to our findings from both the survey and the quantitative analysis, we discovered that there is no rule on who is allowed or not allowed to change a priority. We encounter several cases where developers, team leaders, or project owners have the permission to change the priority without prior approval process or team decision making. As a recommendation, we suggest that bug tracking software should have the ability to prevent some users from changing the priority and doing so by knowing more about the team structure or hierarchy.

The last group is to answer the third research question as to *when does a priority change*. We discovered that a priority gets changed anytime as long as the bug has not been resolved. We have seen examples where it gets changed after the creation, before assigning to the developer, after assigning to the developer, before or upon closing the bug report. More importantly, we noticed that the priorities get changed in the last phase of bug resolution—a short period of time just before closing the bug report and set a resolution for it. We recommend that bug tracking systems should be aware of the project's future releases, milestones, and the current bugs and features pipeline targeted to the release. Subsequently, the bug tracking software should be able to restrict certain users from changing the priority in a critical time to prevent any delays in resolving the bug reports.

IV. Discussions and Implications

In this section, we discuss the results observed from our experiments and articulate the potential implications for researchers as well as practitioners.

A. *Better understanding of bug priority changes*

The observations gathered from interviews with 11 developers and conducting a survey with 38 participants show the possible reasons of why the priority of bug reports get changed. Moreover, the results show many temporal patterns related to when the priority gets changed especially close to the release deadlines. Our survey shows that, in general, any team member can change the priority including the developers, testers, and project owners. We aimed to better understand these patterns so that researchers, practitioners, and educators can manage their projects, tasks, and bug reports efficiently.

B. *Implications for researchers*

The current state-of-art about the generation of documentation for bug priority changes is still in its infancy. Therefore this study will facilitate the software engineering community to automatically generate documentation for bug priority changes. Furthermore, this study suggests that new tools may need to be developed to validate the changes in bug priority since not all priority changes are made with a good reason or intent.

C. *Implications for tool builders*

This study shows a list of recommendations that could improve bugs tracking systems in two different dimensions: 1) document the changes of the bugs appropriately, and 2) implement checks/validation routines to secure bug priority changes.

Research Question	Findings	Recommendations
Why does the priority of a bug change?	💎 The dependency of another bug's fix 💎 Incorrect priority 💎 Heavy workload /Tight schedule 💎 Category of bug report 💎 Hot-fix request 💎 Business requirements 💎 Type of project 💎 On accident	🚀 Bug tracking systems should have the ability to document the reasons for changing priority 🚀 Priority prediction systems should rely on priority changes to improve their prediction model 🚀 Standardize documentation methods based on different projects' domains and bugs' categories
Who does change the priority of a bug?	💎 Stakeholders including developers, team leaders, project owners	🚀 Bug tracking systems should have the ability to restrict certain users from changing the priority if they don't have permission to do so. 🚀 Bugs tracking systems should be aware of team structure and the role of each stakeholder.
When does the priority of a bug change?	💎 Priority changes happen between the date the bug gets assigned to the developer and date before releasing a new version of the software.	🚀 Bugs tracking systems should prevent stakeholders from changing the priority unnecessarily if the bug milestone is close to the release date or if the bug is active or pending status. 🚀 Bugs tracking systems should be aware of the project's milestones and timelines.

TABLE V
3-W MODEL FINDINGS AND RECOMMENDATIONS

D. *Implications for educators*

Our findings as presented in 3-W model will be the initial guide to help educators to emphasize the importance of bug priority to their students to manage the bug reports. Therefore, educators will be able to teach and transfer this knowledge to future practitioners and tool builders to follow the best practices in managing their software maintenance activities.

E. *Implications for practitioners*

Practitioners will be able to follow the guidelines illustrated in Table V to manage bug priority. Though we do not expect practitioners to document all the bug priority change components independently from the context, we expect them to judge which components are more relevant and adequate for their specific contextual needs. Managers and team-leads can work with developers to establish customized guidelines from this study for documenting bug priority changes and validate them as well. Such guidelines can also be enforced by customizing the continuous integration pipeline. These guidelines could trigger developers to capture the impact and the rationale of their bug priority changes appropriately for each situation, developing beneficial habits, long-lasting documentation, and better check of bug priority changes.

V. THREATS TO VALIDITY

A. *Construct validity*

Some threats can be related to the way we construct our 3-W model. To mitigate this threat, our interviews with developers were limited to a set of predefined questions and give opportunities to the participants to share their experiences without bias. Also, we gave our participants enough time to comprehend our questions so that they can provide opinions and ideas without any implicit inputs and bias. We have included "Other" option with a free text as an answer in some of the interviews and survey questions to open the door for any suggestion or feedback.

B. *Internal validity*

The first possible internal threat is that some participants of the survey may provide a biased perceptive about bug priority changes. To mitigate this threat, we ensured that our participants have diverse backgrounds, different education/experience levels, and hired from different industrial companies to bring their in-house practices to the survey. Another typical co-factor in survey studies is the respondent's fatigue bias, so to mitigate this threat, we ran a pilot study with five PhD students to make sure that the survey can be answered within 10 minutes, and the context of the survey is clear enough to the participants without any ambiguity.

C. *External validity*

The number of participants may not represent very large population of developers. To mitigate this threat, our participants were chosen from a diverse population, with diverse expertise and years of experience as presented in Figure 2.

VI. RELATED WORK

Most of the existing defect management studies focused on the prediction of bug severity/priority from bug reports [9], [13]–[15], [24]–[29]. Machine learning algorithms were extensively used for that purpose such as Support Vector

Study	**Description / Technique Used**	**Have they addressed priority changes?**
Yang et al. [14]	Extract and identify multi-feature (e.g., Component, product, priority and severity) from bug report	No
Tian et al. [9], [13]	Use several factors such as temporal, textual, author, related-report, severity, and product, to predict the priority level of a bug report	No
Sharma et al. [15]	Use Support Vector Machine, Naive Bayes, K-Nearest Neighbors and Neural Network in predicting the priority of bugs	No
Kanwal et al. [11], [23]	Propose a priority recommendation module based on Naïve Bayes and Support Vector Machine.	No
Yu et al. [12]	Utilize neural network techniques to predict the priorities of bugs	No
Alenezi et al. [24]	Present an approach to use different machine learning algorithms namely Naive Bayes, Decision Trees, and Random Forest	No
Kumari et al. [10]	Build classifiers using machine learning and Naïve Bayes and Deep Learning techniques	No

TABLE VI
SUMMARY OF PREVIOUS STUDIES ABOUT BUG PRIORITY PREDICTIONS.

Machine, Naive Bayes, K-Nearest Neighbors, and Neural Networks. To the best of our knowledge, there is no existing study about understanding the changes in bug's priority and their rationale. We present, in the following, the closest studies to this paper but a more comprehensive summary can be found in Table VI about the prediction of bugs' priority.

Yang et al. [14] proposed an approach to managing the bug triage by predicting the workload. They were also able to extract and identify multi-feature (e.g., Component, product, priority, and severity) from bug reports in order to assign developers to bugs and predict the severity of those bugs [14].

Tian et al. [9], [13] proposed an automated approach using machine learning to recommend a priority level based on information available in bug reports. Their method used several factors such as temporal, textual, author, related-report, severity, and product, to predict the priority level of a bug report [9], [13].

In the work of Sharma et al. [15], they use different machine learning techniques such as Support Vector Machine, Naive Bayes, K-Nearest Neighbors, and Neural Network in predicting the priority of bugs. Also, they evaluated the performance by performing cross-project validation [15]. Similarly, Kumari et al. [10] built classifiers using machine learning and Naïve Bayes and Deep Learning techniques. These classifiers considered the severity, summary weight, and entropy attribute to recommend the priority of bugs [10].

Yu et al. [12] used neural network techniques to predict the priorities of bugs, adopted an evolutionary training process to solve problems associated with reducing features, and reused data sets from similar software systems to speed up the convergence of training [12].

Kanwal et al. [23], proposed a priority recommendation module based on Naïve Bayes and Support Vector Machine. Also, they provided another comparative study to evaluate which classifier performs better in terms of accuracy [23].

Alenezi et al. [24] presented an approach to predict the priority of a reported bug using different machine learning algorithms namely Naive Bayes, Decision Trees, and Random Forest. They also evaluated the performance of each one of these algorithms in predicting the priority of bug reports [24].

As a summary, all existing research papers focus on predicting the priority of bugs and therefore it helps in the bugs triage process and assigning developers to given bugs. More details can be found in Table VI.

VII. CONCLUSIONS

In this paper, we used a combination of qualitative and quantitative techniques (interviews, a survey, and bug reports analysis) to understand the changes in bug priority and their rationale. We started first with a set of interviews with practitioners to define a bugs priority change model. Then, we performed a large online survey to gather the experiences of practitioners with the rationale, frequency, and experiences of changing the priority of bugs. We have also collected a large data-set of bugs priority change on open-source projects. We looked into actions that happen in bug reports such as the date when priority changes happen, the reasons beyond changing the priority documented in the comments, and the profile of the users who changed the priority. The quantitative validation on this created data-set revealed several areas of improvement as discussed in the implications section.

The outcomes of this empirical study can be used to build tools for automatically validating the changes' request for bugs priority. We are planning as part of our future work to leverage machine learning to check and validate the bug priority changes submitted by developers based on the data-set collected in this study [18].

References

[1] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, "Non-functional requirements," *Software Engineering*, 2000.

[2] S. Zaman, B. Adams, and A. E. Hassan, "Security versus performance bugs: a case study on firefox," in *Proceedings of the 8th working conference on mining software repositories*, 2011, pp. 93–102.

[3] P. J. Guo, T. Zimmermann, N. Nagappan, and B. Murphy, "Characterizing and predicting which bugs get fixed: an empirical study of microsoft windows," in *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering-Volume 1*, 2010, pp. 495–504.

[4] T. Zimmermann, N. Nagappan, P. J. Guo, and B. Murphy, "Characterizing and predicting which bugs get reopened," in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 1074–1083.

[5] E. Shihab, A. Ihara, Y. Kamei, W. M. Ibrahim, M. Ohira, B. Adams, A. E. Hassan, and K.-i. Matsumoto, "Studying re-opened bugs in open source software," *Empirical Software Engineering*, vol. 18, no. 5, pp. 1005–1042, 2013.

[6] K. Chaturvedi and V. Singh, "Determining bug severity using machine learning techniques," in *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*. IEEE, 2012, pp. 1–6.

[7] A. Lamkanfi, S. Demeyer, Q. D. Soetens, and T. Verdonck, "Comparing mining algorithms for predicting the severity of a reported bug," in *2011 15th European Conference on Software Maintenance and Reengineering*. IEEE, 2011, pp. 249–258.

[8] X. Xia, D. Lo, M. Wen, E. Shihab, and B. Zhou, "An empirical study of bug report field reassignment," in *2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*. IEEE, 2014, pp. 174–183.

[9] Y. Tian, D. Lo, and C. Sun, "Drone: Predicting priority of reported bugs by multi-factor analysis," in *2013 IEEE International Conference on Software Maintenance*. IEEE, 2013, pp. 200–209.

[10] M. Kumari and V. Singh, "An improved classifier based on entropy and deep learning for bug priority prediction," in *International Conference on Intelligent Systems Design and Applications*. Springer, 2018, pp. 571–580.

[11] J. Kanwal and O. Maqbool, "Managing open bug repositories through bug report prioritization using svms," in *Proceedings of the International Conference on Open-Source Systems and Technologies, Lahore, Pakistan*, 2010, pp. 22–24.

[12] L. Yu, W.-T. Tsai, W. Zhao, and F. Wu, "Predicting defect priority based on neural networks," in *International Conference on Advanced Data Mining and Applications*. Springer, 2010, pp. 356–367.

[13] Y. Tian, D. Lo, X. Xia, and C. Sun, "Automated prediction of bug report priority using multi-factor analysis," *Empirical Software Engineering*, vol. 20, no. 5, pp. 1354–1383, 2015.

[14] G. Yang, T. Zhang, and B. Lee, "Towards semi-automatic bug triage and severity prediction based on topic model and multi-feature of bug reports," in *2014 IEEE 38th Annual Computer Software and Applications Conference*. IEEE, 2014, pp. 97–106.

[15] M. Sharma, P. Bedi, K. Chaturvedi, and V. Singh, "Predicting the priority of a reported bug using machine learning techniques and cross project validation," in *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)*. IEEE, 2012, pp. 539–545.

[16] S. Zaman, B. Adams, and A. E. Hassan, "A qualitative study on performance bugs," in *2012 9th IEEE working conference on mining software repositories (MSR)*. IEEE, 2012, pp. 199–208.

[17] J. Uddin, R. Ghazali, M. M. Deris, R. Naseem, and H. Shah, "A survey on bug prioritization," *Artificial Intelligence Review*, vol. 47, no. 2, pp. 145–180, 2017.

[18] A. authors. (2020) Replication package. URL: https://sites.google.com/view/scam2020-bugs-priority.

[19] M. Codoban, S. S. Ragavan, D. Dig, and B. Bailey, "Software history under the lens: A study on why and how developers examine it," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2015, pp. 1–10.

[20] M. Hilton, N. Nelson, T. Tunnell, D. Marinov, and D. Dig, "Trade-offs in continuous integration: assurance, security, and flexibility," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 2017, pp. 197–207.

[21] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," in *Guide to advanced empirical software engineering*. Springer, 2008, pp. 285–311.

[22] Y. Tao, Y. Dang, T. Xie, D. Zhang, and S. Kim, "How do software engineers understand code changes? an exploratory study in industry," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, 2012, pp. 1–11.

[23] J. Kanwal and O. Maqbool, "Bug prioritization to facilitate bug report triage," *Journal of Computer Science and Technology*, vol. 27, no. 2, pp. 397–412, 2012.

[24] M. Alenezi and S. Banitaan, "Bug reports prioritization: Which features and classifier to use?" in *2013 12th International Conference on Machine Learning and Applications*, vol. 2. IEEE, 2013, pp. 112–116.

[25] A. Ouni, M. Kessentini, M. Ó Cinnéide, H. Sahraoui, K. Deb, and K. Inoue, "More: A multi-objective refactoring recommendation approach to introducing design patterns and fixing code smells," *Journal of Software: Evolution and Process*, vol. 29, no. 5, p. e1843, 2017.

[26] A. Ghannem, G. El Boussaidi, and M. Kessentini, "On the use of design defect examples to detect model refactoring opportunities," *Software Quality Journal*, vol. 24, no. 4, pp. 947–965, 2016.

[27] B. Amal, M. Kessentini, S. Bechikh, J. Dea, and L. B. Said, "On the use of machine learning and search-based software engineering for ill-defined fitness function: a case study on software refactoring," in *International Symposium on Search Based Software Engineering*. Springer, Cham, 2014, pp. 31–45.

[28] M. Kessentini, A. Ouni, P. Langer, M. Wimmer, and S. Bechikh, "Search-based metamodel matching with structural and syntactic measures," *Journal of Systems and Software*, vol. 97, pp. 1–14, 2014.

[29] A. Ghannem, M. Kessentini, and G. El Boussaidi, "Detecting model refactoring opportunities using heuristic search," in *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, 2011, pp. 175–187.