Behavior Metrics for Prioritizing Investigations of Exceptions

Zack Coker*, Kostadin Damevski†, Claire Le Goues*, Nicholas A. Kraft‡, David Shepherd‡, and Lori Pollock§
Carnegie Mellon University*, Virginia Commonwealth University†, ABB Corporate Research‡, University of Delaware§
Pittsburgh*, Richmond†, Raleigh‡, Newark§, USA
{zfc, clegoues}@cs.cmu.edu*, kdamevski@vcu.edu†, {nicholas.a.kraft, david.shepherd}@us.abb.com‡, pollock@udel.edu§

***Abstract*—Many software development teams collect product defect reports, which can either be manually submitted or automatically created from product logs. Periodically, the teams use the collected defect reports to prioritize which defect to address next. We present a set of behavior-based metrics that can be used in this process. These metrics are based on the insight that development teams can estimate user inconvenience from user and application behavior in interaction logs. To estimate user inconvenience, the behavior metrics capture important user and application behavior after exceptions (the defects of interest in our case). We validated these metrics through a survey of how developers would incorporate the behavior metrics into their prioritization decisions. We found that developers change their priority of investigating an exception about 31% of the time after including the behavior metrics in the priority decision. These findings provide evidence that behavior metrics provide a promising advance towards prioritizing application exceptions.**

***Index Terms*—Bug Triage, Behavior Metrics, IDE Usage Data, Exceptions, Stack Traces**

I. Introduction

While developers try to fix as many bugs (application defects) as possible before release, products commonly ship with unresolved bugs [1]. Once these projects are released, users and developers report product issues through bug reports [2]. Multiple prior investigations have found that large projects have large backlogs of bug reports [3] and receive more bugs than developers can address [4]. In these cases, the development team has to triage bug reports [4]. Prior work on bug triage has focused on assigning bugs to developers [5]. Other work has created techniques that automatically prioritize bugs reports [2], [6], which recommend a bug fix order to developers [6]. However, these automated prioritization techniques suffer from accuracy problems [6], [7], so many industrial teams still perform manual bug prioritization.

One step towards automating the bug prioritization process is to automate crash reports, which are core dumps from application crashes that are automatically sent to the application's developers [8]. Modern applications, such as the Windows operating system [8] and Firefox [9], use automatic crash reports to collect information about application failures. Automatic reports often contain more than just crashes. Many applications automatically report and collect a variety of runtime problems, such as exceptions thrown by the application or assertion violations. While automatic reports contain information that helps developers fix application problems [10], developers currently use metrics that may not accurately represent the problem's severity (the negative impact on the application's quality caused by the exception), such as the number of exception instances and number of users affected [8].

In this paper, we focus on the exception prioritiztion subset of the bug prioritization problem. As a target application, we use RobotStudio[1], an Integrated Development Environment (IDE) for robotic applications. RobotStudio has over 8,000 users who used the application for a combined 25,000+ hours each week of May 2016. The RobotStudio development team follows an Agile process composed of sprints. At the outset of each sprint, the RobotStudio product manager selects the product's exceptions for the team to investigate. That is, the product manager receives automatic exception reports from customers' product instances and selects a subset of them for the team to investigate further in each sprint. These C# exceptions occur during the users' interactions with the application. The team's developers then decide which of the exceptions in the exception subset should be fixed first.

At present, the product manager often selects the exception subset based on the number of exception instances and number of users affected. However, many exceptions with different severity produce similar results for the two metrics, and moreover those metrics do not completely measure the quality of RobotStudio as perceived by RobotStudio's user base. Instead, other factors, such as the severity of an exception for a user, may also be informative when selecting exceptions.

Our insight is that the development team should factor user inconvenience from exceptions into their triage decisions, to estimate the exception's severity. The development team collects user interaction logs, which document the application's exceptions along with the user and application interactions before and after exceptions. Such interaction logs could inform estimates of the user inconvenience caused by different exceptions, leading the team to prioritize more severe exceptions.

Using this scenario as a guide, we propose to improve the process of exception triage with behavior metrics, or measurements of user and application response to application problems as a proxy for the severity of the problems. While behavior metrics are not necessarily application specific, we focused our investigation on the exceptions thrown in Robot-

[1]new.abb.com/products/robotics/robotstudio

```
1  OnlineControllerRemove
2  AddVirtualController
3  Exception|Services.RobApi.RobApiException
4  ----EXCEPTION BEGIN----
5     at System.Runtime.CompilerServices.TaskAwaiter
          .ThrowForNonSuccess(Task task)
6  ----EXCEPTION END----
7  AddedController
8  VirtualFlexPendant
9  RapidEditorShow
```

Fig. 1. A simplified excerpt from a RobotStudio interaction log. This excerpt shows how exceptions are recorded in the interaction logs.

Studio. We calculated the behavior metrics from sequences in RobotStudio interaction logs that could provide insights into the degree in which exceptions imped user tasks (e.g., restarting RobotStudio soon after an exception).

We validated the behavior metrics with a survey of 12 RobotStudio developers. The survey investigated how participants changed their priority of investigating exceptions and their confidence in their priority evaluations after seeing behavior metrics calculated from past interaction logs. We found that developers considered the behavior metrics useful; 30.6% of respondents changed their exception investigation priority answers after seeing the behavior metrics.

The contributions of this paper are:

- new metrics for prioritizing automatically generated exception reports based on user and application interactions
- survey results that demonstrate developers change their ratings of an exception's investigation priority and their confidence in their priority assessments after the behavior metrics were presented

The remainder of this paper is organized as follows. We first present and motivate the behavior metrics (Section II). Then, we explain how we validated the behavior metrics through a survey (Section III) and present the survey results (Section IV). Next, we discuss the threats to validity (Section V), related work, (Section VI) and conclusions (Section VII).

II. Behavior Metrics

In this section, we first present an example interaction log that we used to calculate the metrics (Section II-A). Then, we describe the metric methodology and provide preliminary definitions (Section II-B). Next, we motivate and define the behavior metrics (Section II-C).

A. Interaction Logs

We calculate the behavior metrics from records of user-application interactions before and after exceptions. The behavior metrics can be extracted from sequences of users' interactions and application exceptions.

In our context, we use RobotStudio interaction logs to calculate the behavior metrics. The RobotStudio development team collects interaction logs to improve the product's user interface design with customer usage information. The interaction logs contain the anonymized sequence of interactions between the user and the product, which consist of actions taken by the user and the application's responses to these actions. From the interaction logs, we can deduce the sequence of actions that the user took before and after an exception, but not complete information about the user's or the system's actions (e.g., the interaction log will show a file was saved but not which file was saved). While this imperfect information protects a user's privacy, the lack of detail prevents the reproduction of all exceptions from only the information in the interaction logs.

Figure 1 shows a segment from a RobotStudio interaction log. Interaction logs contain the list of actions performed by the user in sequential order, shown in lines 1–2 and 7–9. The interaction logs also contain inline descriptions of exceptions that occurred during execution. A simplified exception description is shown in lines 3–6. This description includes a message that describes the exception and the stack trace of the exception. The *exception type* is composed of the exception's name, the second item on line 3 (`Services.RobApi.RobApiException`); and the exception's stack trace, shown on line 5.

B. Metric Methodology and Preliminary Definitions

Based on a manual analysis of a random 300 interaction log subset of the 296,581 logs with exceptions that span over a 5-month period, we identified 5 behavior patterns. Three of the behavior patterns consist of user responses to an exception (`Single Restart`, `Multiple Restarts`, and `Repeat Action`) and two patterns indicate how the application responds to an exception (`Exception Repeats` and `Similar Exceptions`).

Before specifying the behavior patterns, we present three definitions. (1) A *user action* is an execution of an application command, or state change, caused by the application's user. In Figure 1, user commands are lines 1–3 and 7–9. (2) A *session* is all logged interactions from the time the user open the application to the time the application's process ends (i.e., user exits or application crashes). The interactions in a session include user actions and application exceptions. Each log file is a single session. (3) *Exceptions* are application errors that are logged with a name and stack trace.

C. Metric Definitions

We created metrics to aid developers in the exception triage process of a sprint. During the sprint process, development teams can calculate these metrics from past interactions logs to estimate how often an exception causes user inconvenience. The metrics show how often user are impeded when working with RobotStudio, which can be used to denote which exceptions are causing the greatest user inconvenience.

We define each *behavior metric* with its corresponding behavior type. Each behavior metric shares the same name as the behavior pattern used to calculate the metric. All behavior patterns can extend across multiple user sessions.

`Single Restart`. This pattern occurs when the exception in the interaction log is followed by at least one end of session in the next *10* user actions. This pattern is a superset of the `Multiple Restarts` pattern.

`Multiple Restarts`. This pattern occurs when the exception in the interaction log is followed by >1 end of sessions in the interaction logs for that user in the next *10* user actions.

`Repeat Action`. This pattern occurs when the last user action before an exception occurred is repeated in the next *10* user actions after the exception.

`Repeat Exception`. This pattern occurs when the same exception occurs in the next *10* user actions after the exception.

`Similar Exception`. This pattern occurs when a similar exception occurs within the next *10* user actions after the first exception. Similar exceptions are exceptions that have the same exception name (e.g., `Services.RobApi.RobApiException`) but different stack traces. This pattern is limited to similar exceptions because, in this product, exceptions are more likely to be caused by a similar exception.

We determined the *10* user actions in the behavior metrics from manual inspection of the log files. This value may need to be adjusted for different applications.

The behavior metric for an exception type corresponds to two values: (1) the percentage and (2) the rank (examples in Figure 2). We calculate the behavior metric percentage for an exception type e and behavior pattern b using the equation:

$$\frac{number\ of\ \boldsymbol{e}\ instances\ associated\ with\ \boldsymbol{b}}{number\ of\ \boldsymbol{e}\ instances} \tag{1}$$

This percentage is calculated over some time period, which can be the full product history. In our case, we had data for five months, so we limit the calculation to instances in that five month period. We calculate the metric rank by ranking the exception types in the data set by the behavior metric percentage in descending order. Ties are allowed in the ranking (2 exceptions with a metric percentage of 100% would both be ranked 1st, the next highest would be ranked 3rd).

D. Goal of the Metrics

The goal of the metrics were to provide insight into user inconvenience with exceptions to aid developers in the exception prioritization process, not to make the decision for developers. Each behavior metric was designed with a different scenario in mind. `Multiple restarts` was designed to approximate situations were users cannot do what they want to do without encountering a session ending exception. `Single restart` was designed to measure a less severe situation than `multiple restarts`, where the exception causes a user to enter a state that the user decides to fix by restarting the application. The other metrics were designed to measure less severe situations. `Repeat action` measures the situation where the user has to repeat the failing action due to an exception. `Repeat exception` measures the situation where the application enters a bad state and continues to throw the same exception. `Similar exception` measures when the application goes into a bad state, where one exception causes another exception.

III. Validation Study Methodology

In this section, we present the behavior metrics research questions (Section III-A), the exceptions in the survey (Section III-B), the survey methodology (Section III-C), and the statistical tests in the evaluation (Section III-D).

A. Research Questions

We designed a survey to study the following set of research questions to better understand the benefits of behavior metrics when prioritizing exceptions.

RQ1: Do behavior metrics help developers prioritize which exceptions to investigate first?

We created the metrics to assist in the exception triage process in Section I. Thus, the actionability (the ability to make an empirically informed decision based on the metrics' values) of the metrics was our most important validation criterion [11]. We evaluate the actionability of the metric by participants' changes in investigation priority and confidence.

RQ2: Does developers' behavior metric interpretation differ based on experience with RobotStudio exceptions?

Past work has shown experienced developers respond differently than novice developers to different development situations [12]. These results in similar circumstances suggest differences in a developer's RobotStudio exception experience may cause different behavior metrics interpretations.

RQ3: Do developers consider certain metrics more influential than others when determining investigation priority?

Prior work has found that certain metrics are more useful than other metrics in different situations (e.g. defect prediction [13]). Thus, we investigated whether certain metrics were more influential using participants' selected change reasons.

RQ4: What causes developers not to change priority after incorporating the behavior metrics in their priority decisions?

While the behavior metrics should assist developers when making priority decisions, the metrics will probably not be the only factor when developers make their final priority decisions. For example, developers may consider the information in the stack trace to be more influential than the behavior metrics for certain exceptions. Past work has found that developers consider stack traces to be one of the most important aspects of a bug report [14]. Behavior metrics could also add support for a developer's initial priority determination.

B. Selected Exceptions For Survey

In this subsection, we first present the exceptions in the data set and then explain the selection process for the exceptions in the survey. We present the exception information for two reasons: (1) to provide intuition about how the metrics are distributed across exceptions from a sample data set, which may provide insight when applying the metrics to a new data set, and (2) to show the metrics for the selected exceptions, which influenced participants' survey answers. In this section, we also explain the steps taken to reduce participants' bias due to the selected exceptions.

Exception Types and Behavior in Whole Dataset. To aid in understanding how the metrics apply to a real-world system, we present the distributions of the metrics across the exceptions in the RobotStudio data set. We filtered the 6,982 exception types in the data set to the exceptions with

	Exception	Instances #	Instances rank	Users Affected #	Users Affected rank	Single Restart %	Single Restart rank	Multiple Restarts %	Multiple Restarts rank	Repeat Action %	Repeat Action rank	Exception Repeats %	Exception Repeats rank	Similar Exception %	Similar Exception rank
1	RobApi	508	163	122	110	16	388	5	362	100	1	1	375	0	94
2	RobApi	333	221	224	47	16	385	5	366	6	444	2	357	0	53
3	ProductInstallationFailed	1,205	85	220	51	73	88	53	26	27	380	61	10	1	27
4	TypeInitialization	942	97	121	113	89	45	62	14	99	148	44	41	1	35
5	RobApi	196	316	151	90	10	424	2	414	100	1	0	413	0	94
6	NullReference	326	226	165	75	100	1	24	194	1	436	10	236	0	52
7	NullReference	280	251	109	128	100	1	33	108	19	405	3	325	0	94
8	ProductInstallationFailed	482	175	105	134	66	120	37	81	38	351	64	7	1	32
9	TypeInitialization	1,981	53	239	43	87	50	67	9	100	131	49	27	1	32
10	Argument	182	330	105	134	5	440	1	430	3	450	0	413	0	94
11	RobApi	507	165	330	28	98	27	39	68	24	384	5	296	3	10
12	InvalidOperation	58	177	116	1756	99	24	36	87	79	249	2	340	4	6

Fig. 2. The exceptions presented in the survey. We numbered the exceptions to differentiate exceptions with the same name but a different stack trace. # are the number of users affected (Users Affected column), and total exception instances (Instances column). % means the percentage of total exception instances associated with the behavior patterns (all columns with %). *rank* in the Instances column denotes the position of the exception when all exceptions are ordered by most to least instances. For the other columns, *rank* denotes the position of the exception when all exception types are ordered from the highest to lowest percentage (e.g., *rank* in Users Affected means the position when exceptions are ordered by number of users affected). Ties are permitted in the rank (further explained in Section II-C).

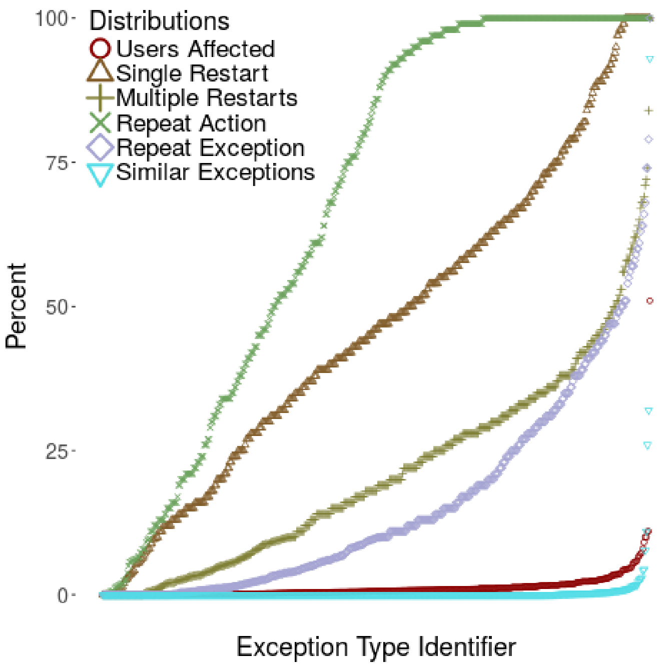

Fig. 3. The distributions of behavior metric percentages in the data set shown through a scatter plot. Each distribution is independently sorted, so the first exception type identifier is the lowest percentage for each distribution, and not the same exception type in each distribution.

		Spearman Rho	p-value
single restart	multiple restarts	0.848	<0.001

TABLE I
THE STATISTICALLY SIGNIFICANT (<0.05), POSITIVELY CORRELATED (>0.7) SPEARMAN RHO CORRELATIONS BETWEEN BEHAVIOR METRIC PERCENTAGES IN THE DATA SET.

PCA Component: single restart	multiple restarts	repeat action	repeat exceptions	similar exceptions	% of Variance Explained by Component
-0.332	**-0.216**	**0.885**	**-0.246**	**0.004**	**51.912**
0.805	**0.442**	**0.394**	**-0.055**	**0.013**	**35.082**
0.143	0.159	0.247	0.945	-0.010	9.198
-0.472	0.856	-0.025	-0.209	0.007	2.953
0.007	0.009	0.006	-0.012	-1.000	0.856

TABLE II
THE EXPLAINED VARIANCE OF THE COMPONENTS GENERATED BY PRINCIPLE COMPONENT ANALYSIS ON THE BEHAVIOR METRIC PERCENTAGES IN THE DATA SET. THE FIRST TWO COMPONENTS, WHICH CONSIST OF THE FIRST FOUR METRICS, ACCOUNT FOR ABOUT 87% OF THE DATA'S VARIANCE.

100 or more instances to prevent exceptions with a low occurrence rate from skewing the distributions. After filtering the exceptions, we were left with 470 exception types. We first show the distribution of the percent of users affected by the filtered exceptions to provide contrast with a metric used in prior work [8], [9] and RobotStudio. Figure 3 shows the scatter plot for the percent of users affected and the behavior metric percentages for different exception types in the data set. Notice that the user response patterns occur more frequently than the application response patterns.

Table I shows the correlation between the different behavior metric percentages in the data set. These correlation results show only `single restart` and `multiple restarts` are highly correlated, which is expected because `single restart` is a superset of `multiple restarts`. Table II shows the explained variance of the different components produced by a Principle Component Analysis (PCA) [15] of the data set. The PCA results show the fist two components explain about 87% of the variance in the data set. The results also show the `similar exceptions` metric explains very little variance (<1%) and may be redundant.

Exception Types used in Survey. We determined the survey length through a multi-step process. We were interested in documenting participants' priority and confidence changes for exceptions that were high in individual behavior metrics. For comparison, we also needed exceptions that were not high in any of the metrics. Thus, we had six exception categories of

		Spearman Rho	p-value
single restart	multiple restarts	0.958	<0.001
	repeat exception	0.884	<0.001
mulitple restarts	repeat exception	0.933	<0.001

TABLE III
THE STATISITICALLY SIGNIFICIANT (<0.05), POSITIVELY CORRELATED (>0.7) SPEARMAN RHO CORRELATIONS BETWEEN BEHAVIOR METRIC PERCENTAGES IN THE PRESENTED EXCEPTIONS.

PCA Component					% of Variance
single restart	multiple restarts	repeat action	repeat exceptions	similar exceptions	Explained by Component
-0.614	**-0.402**	**-0.631**	**-0.253**	**-0.010**	**89.103**
0.222	-0.362	0.349	-0.834	0.054	6.748
0.723	-0.028	0.681	-0.102	0.052	4.027
0.210	-0.829	0.129	0.478	0.142	0.119
0.089	-0.142	0.009	0.026	-0.986	0.000

TABLE IV
THE EXPLAINED VARIANCE OF THE COMPONENTS GENERATED BY PRINCIPLE COMPONENT ANALYSIS ON THE PRESENTED BEHAVIOR METRIC PERCENTAGES. A MAJORITY (89%) OF THE PRESENTED EXCEPTION VARIANCE IS EXPLAINED BY THE FIRST COMPONENT, WHICH MAINLY CONSISTS OF THE FIRST FOUR METRICS.

interest: one that was high in each of the behavior metrics and one control group (i.e., not high in any of the metrics). After performing preliminary tests of the survey, we found participants could answer the questions for an exception in about two minutes. To be respectful of participants' time, we created a survey that could be finished in under 30 minutes. This time limit restricted the survey to 12 exceptions (out of 470 in the data set), 2 for each of the 6 exception categories. These 12 exceptions are shown in Figure 2.

We selected the 12 exceptions in the survey based on the exceptions' behavior metrics. To reduce the effect of the number of users affected by the exception type on participants, we first filtered the exceptions in the data set to exceptions which affected between 1–2% of the total RobotStudio users in the data set. We filtered the exceptions to only include exceptions that were in the top 5% percent for an individual behavior metric. Then, we narrowed the list to two exceptions for each behavior metric by selecting the two exceptions that had the lowest sum for the other behavior metric percents (i.e., **`4TypeInitializationException`** and **`9TypeInitializationException`** for **`Muiltple Restarts`**). The two exceptions in the category without a behavior metric were selected by taking the two exceptions with the lowest sum of all the behavior metric percents (i.e., **`2RobApiException`** and **`10ArgumentException`**). When we presented the exceptions in the survey, we presented all of the behavior metrics for each exception, not just the metrics used to select the exceptions.

One drawback of this approach was the high correlation between the metrics in the chosen exceptions. Table III shows the correlations between the presented exceptions and Table IV shows the PCA components of the behavior metric percentages in the presented exceptions. These tables demonstrate the high correlation between **`Multiple Restarts`**, **`Single Restart`**, and **`Repeat Actions`** in the presented exceptions.

Exception Question Set #1
Exception Type, Exception Instances, and Users Affected

1. Please select the priority you would put on further investigating this exception.

Investigate as soon as possible, Investigate in the next sprint, Investigate soon but after next sprint, Investigate when you have time, Ignore

2. Please rate your confidence in your survey answers.

High confidence, Moderate confidence, Low confidence, No confidence

Exception Question Set #2
Exception Type, Exception Instances, Users Affected, and Behavior Metrics

3. Does the new information change your opinion about the exception's investigation priority?

Yes, No

If the participant selected Yes:
4. Please select the recalculated investigation priority.

Investigate as soon as possible, Investigate in the next sprint, Investigate soon but after next sprint, Investigate when you have time, Ignore (without the ability to select their previous investigation priority)

5. Why did you choose that investigation priority?

The number of multiple restarts, The number of single restarts, The number of repeat actions, The number of repeat exceptions, The number of similar exceptions
Other: *(free-form response)*

6. Which option best describes what influenced your investigation priority selection?

Only the new information was used, The new information influenced the decision more than the old information, The new and old information were equally important, The old information influenced the decision more than the new information

7. Please rate your confidence in your answers above.

High confidence, Moderate confidence, Low confidence,

If the participant selected No:
8. Why did your opinion not change?

The new information reinforces the old information, Users affected are more important, Harmful exception instances are too rare, The exception type seems harmless
Other: *(free-form response)*

9. Please rate your confidence in your answers above.

High confidence, Moderate confidence, Low confidence

Fig. 4. The exception survey questions.

C. Survey Structure

In this subsection, we first explain the survey setting. We then explain the layout of the survey and the survey questions.

Survey Setting. We created an online survey for the 17 ABB Inc. developers of RobotStudio. The survey asked RobotStudio developers how their assessment of an exceptions' investigation priority would change due to behavior metrics, with the goal of addressing the research questions presented in Section III-A. We received 12 survey responses. Participants' RobotStudio development experience ranged from 0 to 17 years, with a mean of 6.4 years.

Survey Layout. The survey started with an explanation of the survey's organization and how to take the survey, followed by questions which asked participants about their experience with

RobotStudio and RobotStudio exceptions. Then the survey presented the 12 exceptions in a random order. Finally, the survey concluded by asking participants whether anything else could be done to improve their evaluation of exceptions' investigation priorities and thanked them for completing the survey. In total, the survey contained a maximum of 85 closed-ended questions and 14 open-ended questions.

For each exception, the survey initially presented information that the developers currently used to determine the exception's investigation priority: the type of the exception, the stack trace, the number of users affected by the exception, and the number of exception occurrences. After we presented this exception information to participants, we asked participants to select the priority of investigating the exception. The possible options were on a Likert scale [16] with five categories (survey question 1). Participants were also asked how confident they were in their selection using a Likert scale with four categories (survey question 2). The first set of exception questions are shown in Figure 4. Participants were only able to select one answer for each question in this set.

Once participants answered these questions, the survey presented the behavior metrics for that exception and asked the second set of exception questions. These questions asked participants whether the new information changed their priority of investigating the exception (survey question 3). If participants changed their priority of investigation based on the behavior information, then participants were asked to select the new priority of investigation (survey question 4), provide a reason for the change (survey questions 5 and 6), and select their confidence in the new investigation priority (survey question 7). If participants did not have a priority of investigation change, then participants were asked to provide a reason for not changing their priority (survey question 8) and their confidence in their no priority change decision (survey question 9). We present the second set of exception questions in Figure 4. Participants were only able to select one answer for survey questions 3, 4, 7, and 9. Participants were able to select multiple options for survey questions 5, 6, and 8[2].

D. Statistical Tests in Survey Evaluation

For RQ1, we wanted to evaluate if the behavior metrics caused participants to change their investigation priority and confidence. Thus, use the Wilcoxon signed rank test, which evaluates if two samples from the same population have statistically significant different distributions (between-subject) [17]. For RQ2, we investigated if experience influences investigation priority and confidence rating. To evaluate this question, we use the Wilcoxon rank sum test, which evaluates if two independent populations came from statistically significant distributions (within-subject) [18]. We also investigated the correlation between experience and survey changes. The Spearman's rank correlation coefficient [19] measures the statistical significance of the correlation between to variables. For all the statistical tests, we use a p-value of <0.05 to determine significance.

[2]survey: https://whispering-meadow-68629.herokuapp.com/

IV. Results

In this section, we discuss the survey results. We show how behavior metrics cause priority and confidence changes (Section IV-A) and how experience affects how developers use the metrics (Section IV-B). We then discuss the reasons developers gave for making changes (Section IV-C), and the reasons for not making changes (Section IV-D).

A. Behavior Metrics Cause Priority And Confidence Changes

For RQ1 (*Do behavior metrics help developers prioritize which exceptions to investigate first?*), we investigate whether participants change their priority of investigation evaluations after accounting for the behavior metrics. Figure 5 shows how participants changed their priority of investigation (from survey question 1 to survey question 4) after incorporating the behavior metrics into the priority decision. Figure 5 also shows how participants changed their confidence (from survey question 2 to survey question 7 or 9) after incorporating the behavior metrics. While two participants reported having no priority or confidence changes due to the behavior metrics, those participants also reported inexperience with RobotStudio and RobotStudio exceptions. The effects of experience on participants' answers are further discussed in Section IV-B.

We found that participants change their original priority evaluations (survey question 1) 30.6% (44/144) of the time. 75% (33/44) of those changes were increases in priority while 25% (11/44) were decreases in priority. The behavior metrics caused participants to change their original confidence (survey question 2) 33.3% (48/144) of the time. 75% (36/48) of those changes were increases in confidence while 25% (12/48) were decreases in confidence. These results suggest that participants found behavior metrics useful.

Developers commonly adjusted their priority and confidence based on the behavior metrics. If values are assigned to the different priorities (0 for ignore, 1 for free-time, etc.) in survey questions 1 and 4, then there was a significant priority difference before and after developers were presented behavior metrics. The mean original priority (survey question 1) and mean final priority (survey question 4 or when developers kept the same priority by answering *No* to survey question 3) were similar, 2.0 and 2.2 respectively. However, there was a significant difference between the two samples. A Wilcoxon signed rank test showed the original and final priority selections are drawn from different distributions with a high probability, $W = 247.5$, $p < 0.05$, $r = 0.62$. These results provide evidence that participants found behavior metrics useful when determining the priority of investigating exceptions.

Confidence changes also occurred after the behavior metrics were presented. Different confidence answers were assigned different scores (0 for none, 1 for low, etc.). The mean of the original confidence (survey question 2) was 1.9 and the mean of the final confidence (survey question 7 or 9) was 2.1. A Wilcoxon signed rank test showed the original confidence

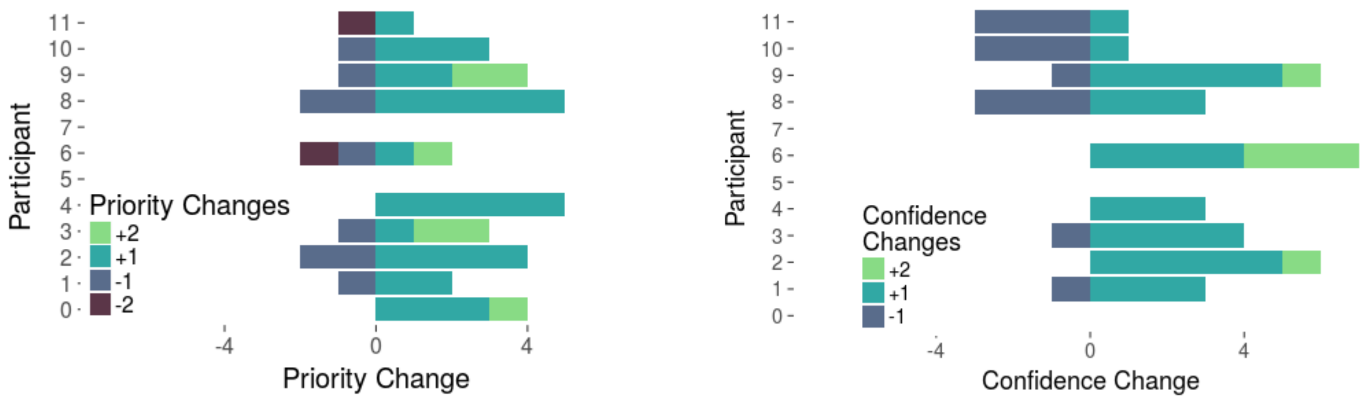

Fig. 5. Left: The priority changes of all survey participants after factoring in the behavior metrics. Right: The confidence changes of all survey participants after factoring in the behavior metrics. For both graphs, the number of changes is shown as the length of the bar while the magnitude of the changes is shown as the color of the bar. The change values are calculated using the conversion discussed in Section IV-A.

and final confidence distributions are different with statistical significance $W = 246$, $p < 0.05$, $r = 0.75$. The confidence changes provide evidence that participants considered the behavior metrics useful when they made priority judgments.

Developers provided multiple comments that indicate they took the behavior metrics into account.

"If the users restarts in high frequency, this error must be annoying. [sic]"

"When I saw the low number of repeats etc. I changed my mind a bit. If very few users are affected and the problem just happens 'once' then it's not so important after all. [sic]"

"The rating of restarts is very high. This could mean we should look into it before [the next] sprint."

In total, 27% (10/37) of participants' comments mentioned using behavior metrics in their decisions. When participants selected how the new and old information was factored into their change decisions (survey question 6), 70% (31/44) of the selected answers stated that the new metrics were more important than the old metrics. Based on the result in this section, we conclude that developers find the behavior metrics useful when investigating the priority of exceptions.

B. Effects of Developers' RobotStudio Exception Experience

RQ2 states — *Does behavior metrics interpretation differ based on RobotStudio exception experience?* We investigated this question by separating the participant responses based on their experience with RobotStudio exceptions.

Figure 6 shows the priority of investigation and confidence changes from the behavior metrics grouped by experience with RobotStudio exceptions. Six developers who reported none, very little, or some experience with RobotStudio exceptions were grouped into the less-experience category. The other six developers (who reported quite a bit or very much) were grouped into the more-experienced category. This graph shows developers with more experience had more investigation priority and confidence changes. Less experienced participants were more likely to change their investigation priority than their confidence. More experienced participants were more likely to change their confidence than their investigation priority.

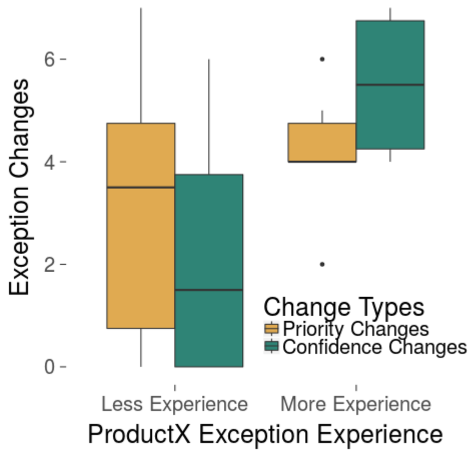

Fig. 6. A box-and-whisker plot of participant's investigation priority and confidence changes grouped by reported experience with RobotStudio exceptions. The change values are calculated using the conversion in Section IV-A.

To determine the independence of the metric interpretations based on experience, we calculated the Wilcoxon rank sum test [18] between the priority changes of participants with more experience and participants with less-experience. A Wilcoxon rank sum test showed there was not a statistically significant difference between the priority changes of more-experienced and less-experienced participants. We next calculated the Wilcoxon rank sum test between the confidence changes of participants with more experience and participants with less-experience. A Wilcoxon rank sum test showed the effect between the confidence changes of more-experienced participants and less-experienced participants to be $W = 4.5$, $p < 0.05$, $r = 0.64$. These results show there is a statistically significant difference in the number of a participant's confidence changes between more and less-experienced participants. These results provide further evidence that more-experienced developers reconsidered their confidence changes more frequently after accounting for the behavior metrics.

To better understand the relationship between experience and metric interpretation, we investigated the correlation between participants' experience and their survey changes. We performed a Spearman's correlation [19] between participants' exception experience and their number of investigation priority changes. To calculate the Spearman's correlation, we converted the participants' exception experience to a 0–4 scale, which corresponded to the participants' selected experience. There was a weak positive and not statistically significant correlation between participants' exception experience and the number of their investigation priority changes, $r_s = 0.245$, $p = 0.442$. We also performed a Spearman's correlation between participants' exception experience and the number of their confidence changes, and we found a statistically significant, positive correlation ($r_s = 0.771$, $p = 0.003$). Since experience changes were correlated with exception experience, we were curious whether participant's confidence increased as their exception experience increased. Thus, we performed a Spearman's correlation between participants' exception experience and their increases in total confidence (a single ordinal increase in confidence for an exception cancels out another single ordinal decrease); and we found a not statistically significant, weak correlation ($r_s = 0.211$, $p = 0.511$). These results show that experience was strongly associated with the number of confidence changes, but we cannot make conclusive claims about the correlation between the number of priority changes or increases in total confidence and exception experience.

These results suggest more experience correlates with changes in behavior metric interpretation. As participants become more experienced with RobotStudio exceptions, participants are more likely to change their confidence than their investigation priority. Participants with more exception experience also had more confidence change, but did not have a statistically significant increase in total confidence.

C. Reasons for Changes

For RQ3, *Do developers consider certain metrics more influential than other metrics when determining investigation priority?*, we investigated the reasons that participants gave for investigation priority changes and created a ranking that averaged the participants' final investigation priority. We found that only two participants consistently changed their investigation priority based on any of the behavior metrics or pattern counts. Instead, participants commonly gave multiple reasons for investigation priority changes. If each presented option and the free response (survey question 5) all count for a single reason each, then developers had 64 reasons for increasing the investigation priority of 33 exception instances, an average of about 2 reasons per increased exception. When prioritizing exceptions after factoring in the behavior metrics, participants consistently considered the exceptions that were not high for any of the metrics, or only high in the `repeat action` metric, as low priority exceptions. Based on these results, it seems that participants consider the `repeat action` metric to be relatively unimportant by itself. Further investigations will be required to determine the influence of the other metrics.

To investigate participant consistency, we also investigated how participants changed investigation priority based on the behavior metrics. We evaluated whether participants created a threshold for the metrics, (e.g., the participant changed the investigation priority for all exceptions with a `single restart` percentage higher than 80%). For each metric, we determined three possible thresholds: (1) the behavior pattern counts (2) the metric percentage (3) the metric rank (the metric percentage and metric rank are shown in Figure 2). We calculated the three thresholds as the largest (1 and 2) or best (3) value from exceptions that the participant did not change the exception's investigation priority. We then compared the thresholds to exceptions where the participant specified that metric as a reason for an investigation priority change. If a participant changed all exceptions that were better than any of the three thersholds, then a participant was consistent.

We found that only two participants were consistent according to this approach: one participant consistently changed the investigation priority when the exception's `single restart` instances were above a threshold, and another participant consistently changed the investigation priority when the exception's `repeat action` instances were above a threshold. Participants did not consistently change exceptions' investigation priority in all other cases. We speculate that these results show participants assessed these metrics as a group when making investigation priority changes and that participants were influenced by other exception information.

When participants had a priority change, 70% (31/44) of the participants' responses contained 2 or more reasons for the change. Certain metrics were often selected together. `Single restart` was selected in 12 of the total 15 (80%) instances where `repeat action` was selected. When `multiple restarts` was selected (17 instances), it was paired with `single restart` 71% of the time (12/17).

When we filtered the responses to the responses that increased priority, `Single restart` was the most common reason for increasing priority (20 of the 64 selected increase reasons), while `multiple restarts` and `repeats action` were tied for the second (both 14 of the 64 increase reasons). These three reasons captured 75% (48/64) of all the reasons selected. Participants selected these reasons when the corresponding metric was high for a particular exception.

Of the 30 reasons provided for decreasing the investigation priority of 11 exception instances, over a third of the reasons selected were `exception repeats` (11/30). Participants selected this behavior metric when it was low for an exception.

When we averaged the selected investigation priority, we found that participants consistently ranked two groups of exceptions at the bottom of their priority range: the exceptions that were not high in any metrics (`2RobApiException` and `10ArgumentException`), and exceptions that only had a high repeat action metric score (`1RobApiException` and `5RobApiException`). This suggests that a high `repeat action` metric was not important enough by itself to produce a high investigation priority. While the exceptions with high similar exception scores (`11RobApiException` and

`l2InvalidOprationException`) were in the group of higher priority exceptions, the results do not seem to indicate that participants responded to the **`similar exception`** metric, and instead responded to other correlated metrics. The **`similar exceptions`** metric was selected as the reason for improving the investigation priority three times (the lowest of the metrics) and these exceptions also had high **`single restart`** metric scores, so it seems that participants considered other metrics when prioritizing those exceptions.

D. No Change Reasons

We looked into the reasons participants provided for not changing their exception priority to answer RQ4: *What causes developers not to change priority after incorporating the behavior metrics in their priority decisions?* To answer this question, we asked participants why their priority decision did not change (survey question 8). We found the most common reason (67 out of the 100 exceptions for which participants priority did not change) was that **`the new information reinforces the old information`**. The next most common reason (22/100) was participants considered the information in the exception type to be more important. For example, participants comments about the exception type included:

"TypeInitializationExceptions are almost always indications that something has gone very wrong."

"NullReference exceptions are always programming errors and should be fixed immediately"

"Seems that the user is closing a station, can be when closing [RobotStudio]. If so get a exception when closing a application is not a severe as having a exception during work. [sic]"

These results indicate the behavior metrics reinforced participants initial priority decisions for most exceptions and the exception type was important for priority decisions.

V. Threats to Validity

External Validity. The general concept of using behavior patterns after exceptions to estimate user impediment should generalize to other applications. However, the metrics collected in this paper may not generalize. For example, other applications may have new exception behaviors which require new behavior patterns. Some participants' survey answers are due to RobotStudio specific information, so the conclusions in the survey may not apply to another application.

Internal Validity. Through survey trials, we determined that the presented exception order may affect participants' exception handling priority responses; survey respondents may change their definition of the different investigation priority categories as they see more exceptions. To mitigate this risk, we presented the exceptions in the survey in random order. Also, the presented exceptions may influence our results. We reduced this risk through our exception selection process and present correlation statistics to make any correlations clear to the reader. Another threat to internal validity is that one participant reportedly took the survey twice, but we were unable to remove the second response due to response anonymization. Another possible threat is averaging ordinal data, which may lead to false conclusions if the distance between the ordinal categories are too skewed. We tried to reduce this threat by choosing categories which we postulate have similar distance, although we did not verify this assumption. Finally, another possible threat is the way we determined the exception types. We used the definition of an exception type that the RobotStudio development team is using in practice, but other definitions of exceptions types may lead to different results.

Construct Validity. The metrics that we present may not completely measure the severity of the exception. Repeating some actions may indicate a more severe exception than other actions, although the metrics cannot discern this. The metrics may also produce incorrect severity measurement in certain situations. For example, if the exception always silently occurs in the shutdown process, the **`single restart`** numbers may indicate a more severe problem than what users experience. These metrics also assume a product with a large user base. Further investigation will have to determine whether the metrics work well for products with a small user base.

Conclusion Validity. Twelve participants may not be a large enough sample to draw definitive conclusions about how these metrics will be interpreted by developers. If the sample was too small, there is greater risk of type II errors (the collected data misses real relationships in the studied phenomenon). There is also the possibility that investigating multiple questions for the same data set led to type I errors (incorrect relationship conclusions caused by chance). Future work can investigate the impact of these threats and reduce any negative consequences.

VI. Related Work

Related work can be divided into three categories: (1) approaches for organizing and prioritizing bug reports using stack traces (2) bug triage, and (3) uses of large-scale behavioral data for software maintenance.

The closest work to ours are those on prioritizing bug reports using stack traces. These works have investigated stack traces on an individual basis and prioritized exceptions by the number of occurrences [8], but they have not considered any of the metrics proposed in this paper. Another stack prioritization work created metrics to identify the failing methods in stack traces [20], such as function frequency and the function's lines of code. Other techniques have focused on accurately grouping reports triggered by the same bug [21], [22].

While we focus on triaging application exceptions, related work has focused on triaging bug reports. Past work in bug report triage has found that filtering bug reports by tag makes prioritizing bug reports faster [3]. Other work has found that crash reports can be effectively prioritized by the number of users affected and the distribution of crash types across users [9], or path similarity metrics and path alignment metrics [23]. Microsoft has also investigated how to prioritize specific bug types, such as performance bugs [24]. A recent survey by Uddin [6] provides a comprehensive overview of bug prioritization techniques, but none of these works use behavior metrics to prioritize investigating exceptions.

Several popular IDEs, such as Eclipse, have released similar data without exceptions, enabling research in developers behavior, such as preferred IDE views [25], code search commands [26], refactoring commands [27], and high-level debugging behavior [28]. While IDE data has been used in many contexts, our work explores the under-investigated area of user behavior and application exceptions in logs.

VII. Conclusion

We have presented behavior metrics, which calculate user and application responses to exceptions, designed to aid in the triage process. These metrics are `single restart`, `multiple restarts`, `repeat action`, `repeat exception`, and `similar exceptions`. We have created these metrics from the logs of an industrial application with thousands of users: RobotStudio.

Through a survey of RobotStudio developers, we determined that developers find the behavior metrics useful when estimating an exception's investigation priority. After considering behavior metrics in their evaluation, developers changed their investigation priority 30.6% of the time. We found that developers with more experience had a higher confidence change rate after incorporating behavior metrics in their evaluation and a slightly increased total confidence. We also found that developers often consider these metrics to reinforce their initial investigation priority decisions. Based on these results, we conclude that developers found the behavior metrics useful.

In the future, more work could be conducted to explore how well the behavior metrics generalize to different contexts. Future work could also investigate how to present behavior metrics in the most insightful way, and determine a way to use these metrics in an automated triage process. While these metrics are an important step in using behavior to triage application problems, these metrics will need to be studied further before these metrics are widely used.

VIII. Acknowledgements

This material is based upon work supported by ABB Inc. and the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE1252522. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or ABB.

References

[1] B. Liblit, A. Aiken, A. X. Zheng, and M. I. Jordan, "Bug isolation via remote program sampling," in *Conference on Programming Language Design and Implementation*, ser. PLDI '03, 2003, pp. 141–154.

[2] 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 *Computer Software and Applications Conference*, ser. COMPSAC '14, 2014, pp. 97–106.

[3] G. Bortis and A. van der Hoek, "Porchlight: A tag-based approach to bug triaging," in *International Conference on Software Engineering*, ser. ICSE '13, 2013, pp. 342–351.

[4] Z. Jie, W. XiaoYin, H. Dan, X. Bing, Z. Lu, and M. Hong, "A survey on bug-report analysis," *SCIENCE CHINA Information Sciences*, vol. 58, no. 2, pp. 1–24, 2015.

[5] G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triage with bug tossing graphs," in *Foundations of Software Engineering*, ser. FSE '09, 2009, pp. 111–120.

[6] 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, Feb. 2017.

[7] A. J. Ko, B. A. Myers, and D. H. Chau, "A linguistic analysis of how people describe software problems," in *Proceedings of the Visual Languages and Human-Centric Computing*, ser. VLHCC '06, 2006, pp. 127–134.

[8] K. Glerum, K. Kinshumann, S. Greenberg, G. Aul, V. Orgovan, G. Nichols, D. Grant, G. Loihle, and G. Hunt, "Debugging in the (very) large: ten years of implementation and experience," in *Operating Systems Principles*, ser. SIGOPS '09, 2009, pp. 103–116.

[9] F. Khomh, B. Chan, Y. Zou, and A. E. Hassan, "An entropy evaluation approach for triaging field crashes: A case study of mozilla firefox," in *Working Conference on Reverse Engineering*, ser. WCRE '11, 2011, pp. 261–270.

[10] P. J. Guo, T. Zimmermann, N. Nagappan, and B. Murphy, "Characterizing and predicting which bugs get fixed: An empirical study of Microsoft Windows," in *International Conference on Software Engineering*, ser. ICSE '10, 2010, pp. 495–504.

[11] A. Meneely, B. Smith, and L. Williams, "Validating software metrics: A spectrum of philosophies," *ACM Transactions on Software Engineering and Methodology*, vol. 21, no. 4, pp. 24:1–24:28, Feb. 2013.

[12] R. Latorre, "Effects of developer experience on learning and applying unit test-driven development," *IEEE Transactions on Software Engineering*, vol. 40, no. 4, pp. 381–395, Apr. 2014.

[13] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 2–13, Jan. 2007.

[14] N. Bettenburg, S. Just, A. Schrter, C. Weiss, R. Premraj, and T. Zimmermann, "What makes a good bug report?" in *Foundations of Software Engineering*, ser. FSE '08, 2008, pp. 308–318.

[15] T. M. F. Taha, E. Shomo, N. E. Oweis, and V. Snasel, "Feature selection by principle component analysis for mining frequent association rules," in *Intelligent Information Technologies for Industry*, ser. IITI '16, 2016, pp. 99–109.

[16] "Likert scale," April 2017. [Online]. Available: http://academic.eb.com/levels/collegiate/article/Likert-scale/605393

[17] D. Rey and M. Neuhäuser, "Wilcoxon–signed–rank test," in *International Encyclopedia of Statistical Science*, M. Lovric, Ed., 2011, pp. 1658–1659.

[18] M. Neuhäuser, "Wilcoxon–mann–whitney test," in *International Encyclopedia of Statistical Science*, M. Lovric, Ed., 2011, pp. 1656–1658.

[19] T. W. MacFarland and J. M. Yates, *Spearman's Rank-Difference Coefficient of Correlation*. Introduction to Nonparametric Statistics for the Biological Sciences Using R, 2016, pp. 249–297.

[20] R. Wu, H. Zhang, S.-C. Cheung, and S. Kim, "CrashLocator: locating crashing faults based on crash stacks," in *International Symposium on Software Testing and Analysis*, ser. ISSTA '14, 2014, pp. 204–214.

[21] T. Dhaliwal, F. Khomh, and Y. Zou, "Classifying field crash reports for fixing bugs: A case study of Mozilla Firefox," in *International Conference on Software Maintenance*, ser. ICSM '11, 2011, pp. 333–342.

[22] Y. Dang, R. Wu, H. Zhang, D. Zhang, and P. Nobel, "ReBucket: A method for clustering duplicate crash reports based on call stack similarity," in *International Conference on Software Engineering*, ser. ICSE '12, 2012, pp. 1084–1093.

[23] V. Akila and G. Zayarz, "Novel metrics for bug triage," *Journal of Software*, vol. 9, no. 12, pp. 3035–3040, 2104.

[24] S. Han, Y. Dang, S. Ge, D. Zhang, and T. Xie, "Performance debugging in the large via mining millions of stack traces," in *International Conference on Software Engineering*, ser. ICSE '12, 2012, pp. 145–155.

[25] G. Murphy, M. Kersten, and L. Findlater, "How are Java software developers using the Elipse IDE?" *IEEE Software*, vol. 23, no. 4, pp. 76–83, 2006.

[26] K. Damevski, D. Shepherd, and L. Pollock, "A field study of how developers locate features in source code," *Empirical Software Engineering*, vol. 21, no. 2, pp. 724–747, Apr. 2016.

[27] M. Vakilian and R. E. Johnson, "Alternate refactoring paths reveal usability problems," in *International Conference on Software Engineering*, ser. ICSE '14, 2014, pp. 1106–1116.

[28] K. Damevski, H. Chen, D. Shepherd, and L. Pollock, "Interactive exploration of developer interaction traces using a hidden markov model," in *International Conference on Mining Software Repositories*, ser. MSR '16, 2016, pp. 126–136.