An Empirical Study on the Handling of Crash Reports in a Large Software Company: An Experience Report

Abdou Maiga, Abdelwahab Hamou-Lhadj, Mathieu Nayrolles, Korosh Koochekian Sabor
SBA Research Lab
ECE, Concordia University, Montreal, QC, Canada
{amiga, abdelw, m_nayrol, k_kooche}@ece.concordia.ca

Alf Larsson
Senior Specialist Observability
Ericsson, Stockholm, Sweden
alf.larsson@ericsson.com

Abstract— **In this paper, we report on an empirical study we have conducted at Ericsson to understand the handling of crash reports (CRs). The study was performed on a dataset of CRs spanning over two years of activities on one of Ericsson's largest systems (+4 Million LOC). CRs at Ericsson are divided into two types: Internal and External. Internal CRs are reported within the organization after the integration and system testing phase. External CRs are submitted by customers and caused mainly by field failures. We examine the proportion and severity of internal CRs and that of external CRs. A large number of external (and severe) CRs could indicate flaws in the testing phase. Failing to react quickly to external CRs, on the other hand, may expose Ericsson to fines and penalties due to the Working Level Agreements (WLA) that Ericsson has with its customers. Moreover, we contrast the time it takes to handle each type of CRs with the dual aim to understand the similarities and differences as well as the factors that impact the handling of each type of CRs. Our results show that (a) it takes more time to fix external CRs compared to internal CRs, (b) the severity attribute is used inconsistently through organizational units, (c) assignment time of internal CRs is less than that of external CRs, (d) More than 50% of CRs are not answered within the organization's fixing time requirements defined in WLA.**

Index Terms— **Software Maintenance, Mining Crash Reports, Empirical Studies, Industrial Systems.**

I. Introduction

Maintenance activities are known to be costly and challenging [19]. Studies have shown that the cost of software maintenance can reach up to 70% of the overall cost of the software development process [1]. At Ericsson, one of the largest telecom and software companies in the world, the maintenance process is particularly complex due to the large client base the company serves. The company needs to react quickly when crashes (due to field failure) are reported. Failing to do so may have a significant impact on Ericsson's operational costs, and may also cause damages to the organization's reputation.

At Ericsson, Crash Reports (CRs)[1] are divided into two categories: internal and external reports. Internal reports are submitted within the organization to report on failures observed by software integration specialists. External reports, on the other hand, are submitted by customers who encounter system crashes (or unintended system behavior) in the field.

Both types of CRs are submitted using the same reporting system and mechanism. They are first triaged by a team that forms the first line of support. If the first line of support cannot provide a fix, it redirects the CR to the second line of support that can redirect it again to another line of support until the problem is resolved.

Both types of CRs are constrained by Working Level Agreements (WLA) that Ericsson has in place to improve team performance and quality of services. Essentially, a WLA describes the *required fixing time* of the CRs depending on their *severity level*. For internal CRs, comparing the actual time it takes to fix a CR to the required time in the WLA is used to assess the performance of the design and testing teams. For external CRs, a WLA contractually engages Ericsson to provide solutions to its customers within the required fixing time.

To further improve the CR handling process, Ericsson was interested in examining the factors that influence the CR process lead time[2] and the differences in the way internal and external CRs are handled, while taking the CR severity level into account. Determining the level of CR compliance with the WLA requirements is another important aspect. Knowing, for example, that a large number of CRs with high severity are originated from the field may trigger organizational changes or additional investment in tool support, due to the implications that such crashes may have on customer satisfaction. Similarly, examining CRs that are reported internally could provide insight to management teams into how effective the testing and integration steps of the development process are.

To draw an accurate picture of the current state, we have conducted an empirical study on Ericsson premises. We examined a large dataset[3] of CRs spanning over two years of activities on one of Ericsson's largest systems (+4 million LOC). We studied the current state of the CR handling process

[1]The term crash reports is the way Ericsson refers to bug reports.

[2] Lead time refers to the lifetime of the CR from the time it is submitted to the time it is resolved (or closed).

[3] We cannot reveal the number of CRs because of confidentiality reasons.

based on the CR type. We compared the proportion and severity of internal CRs to that of external CRs. We also studied the fixing time of the CRs using the CR type and severity as the main variables. Moreover, we examined the CRs that do not meet the WLA timing requirements to uncover the causes. We used statistical methods such as Mann-Whitney tests, Kruskal Wallis, and Pearson's chi-squared independence tests to analyze the data. This paper reports on the main findings of this study.

The remaining parts of this paper are as follows: In Section II, we provide definitions and background to understand the context of the study. In Section III, we describe the study definition and design. The results are presented in Section IV, followed by threats to validity in Section V. In Section VI, we present related work. Finally, we conclude the paper and outline future directions in Section 7.

II. Study Setup

Figure 1 illustrates our data collection and analysis process that we present here and discuss in more details in the subsequent sections. First, we extract data from the CR repository. We then mine the reports to extract the internal and external CRs and the CR handling information. We compute lead time for each CR type. Finally, using statistical methods, we analyze the data based on three research questions (RQ) that we discuss later, and report on the results.

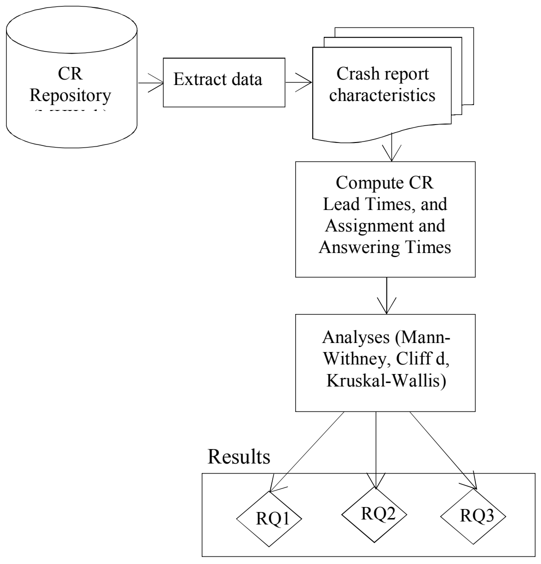

CR Repository
Extract data
Crash report characteristics
Compute CR Lead Times, and Assignment and Answering Times
Analyses (Mann-Withney, Cliff d, Kruskal-Wallis)
Results
RQ1
RQ2
RQ3

Fig. 1. Data collection and analysis process of the study

A. Target System

Ericsson is a Radio Access Network supplier to around 50% of all commercially launched WCDMA (Wideband Code Division Multiple Access) and HSPA (High Speed Packet Access) networks in the world. More than 410 HSPA networks are commercially launched in 162 countries, spread on all continents. The system we used in this study contains more than 50 different network products, all part of the Radio Access Network (RAN) which handles video/mobile telephony, mobile TV, PSTN (public switched telephone network), mobile broadband and fixed wireless broadband. It is built on a number of different embedded hardware types, running more than 200 different software products. The code base is over four millions of lines, implemented in several languages with C/C++ and Java dominating. The development team consists of application teams (Radio Network Controlled and Radio Based Systems) and the platform CPP (Connectivity Packet Platform).

B. Crash Report Management System at Ericsson

Developers report faults and crashes in the system to warn on pending issues with the system functionalities. They use various issue-tracking systems to post the descriptions of faults and fixes. The issue-tracking system, used by the studied unit within Ericsson is called MHWeb. MHWeb is an integrated Web-based environment for maintenance and customer support. It has many components to allow design, maintenance, and support teams to share information that describe faults in the system, fixes, etc. The tool offers an easy-to-use interface for CR handling, analysis and measurement. MHWeb typical functionalities include searching and browsing of CRs.

In MHWeb, a CR is described using the following information:

- CR reference: This is a unique reference for the CR.
- CR type: The type of the CR (internal or external).
- CR severity: The severity of a CR is measured based on the impact of the fault on network traffic. The CR is of severity "A" when the crash affects more than 30% of the traffic or the crash leads to a complete failure. The CR is of severity "B" when the crash affects less than 30% of the traffic or more than 30% in a single occasion. Finally, the CR is considered of severity "C" when it is a minor fault not affecting the traffic at all.
- CR Status: The current status of the CR (registered, assigned, cancelled, fixed, finished, etc.).
- Product information: The faulty product and version, affected by the fault.
- History data: This describes the stages of the CRs and the people who are responsible of handling each stage.

A simplified view of the CR states and the state transition from reporting to providing a solution is described below:

1. The CR is registered.
2. The CR is assigned to a design team to solve it.
3. The CR is solved.
4. A technical answer is provided.

In this study, we only consider CRs with the status "Finished". That is, we exclude ongoing CRs. It should also be mentioned that we exclude duplicate CRs from the dataset. Duplicate reports are marked as duplicate by developers.

C. Data Parsing and Computation

For each CR, we extract the reference number from MHWeb and download its corresponding html file. We extract information from the html files including the description of the CR (CR type, submitter, severity, etc.) and the CR historical data and handling process. Based on the historical data which contains the date and time of each step of the CR, we compute the duration between each step and we extract:

- AT_i: The CR assignment time is the time interval between the time the CR is registered to the time it is assigned to a design team.
- FTi: The CR fixing time is the time interval between the time the CR is registered to when the technical answer to the CR is provided.

III. Study design

We describe the design of our study by first stating the research questions, and then explaining the variables, and analysis methods we used to answer these questions. We formulate three research questions (RQs) with the overall objective to understand the similarities and differences between the way internal CRs and external CRs are handled at Ericsson. The objective of the first research question is to analyze the importance of the testing process (i.e., before the product is released to the customers) by contrasting the number of internal CRs vs. external CRs. The remaining two questions address the duration of CR handling including the duration of CR assignment and fixing according to CR type (i.e., internal, external) and severity.

RQ1: What is the proportion of internal vs. external CRs?

One direct benefit in answering this question it to help Ericsson's development teams assess (at least in the beginning) the effectiveness of the testing process. Knowing, for example, that there is a high number of external CRs may trigger project managers to enhance the testing process so as to find more internal CRs (especially the critical ones) and fix them before the product is released to customers.

To answer this question, we compare the proportion of internal CRs to the total number of CRs. Since not all faults have the same severity level, we also examine the severity of the CRs with respect to their type. For this part, we state the following null hypothesis:

- $H01$: The proportion of internal CRs is not significantly different from the proportion of external CRs with respect to the CR severity.

Variables: When comparing the proportion of internal CRs to that of external CRs based on severity, we use as variables the type of CRs (internal, external) and the severity levels (A, B, C). We build a contingency table and perform the test of independence between the observed frequencies.

Analysis method: We answer RQ1 in two steps. The first step is to use descriptive statistics; we compute the ratio of internal CRs and external CRs to the total number of CRs in the dataset. This shows the importance of CR (in proportion) found during the testing process compared to the proportion of CRs sent by customers. In the second step, we compare the severity of CRs detected internally to the severity of CRs detected by customers by computing the number of CRs for each type and each severity level (A, B, C). We build the contingency table with these two qualitative variables (the type and severity) and we test the null hypothesis H01 to assess if the number of CRs of a particular severity is related to the CR type. We use the Pearson's chi-squared test to reject the null hypothesis $H01$. Pearson's chi-squared independence test is used to analyze the relationship between two qualitative data, in our study the CR type and CR severity. The results of Pearson's chi-squared independence test are considered statistically significant at $\alpha = 0.05$. If p-value < 0.05, we reject the null hypothesis $H01$ and conclude that the proportion of internal CRs for each severity level are significantly different from the proportion of external CRs in severity. is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

RQ2: How fast are CRs fixed?

This question has three aspects. First, we analyze the time it takes to resolve each type of CRs. This will help Ericsson designers and project managers evaluate the resources dedicated to each CR type. Second, we study the impact of CR severity on the CR fixing time. The objective is to assess whether Ericsson's designers take into account the CR severity when prioritizing the CRs. Answering this question can help Ericsson evaluate whether the severity attribute, as it is currently defined, should be maintained for CR prioritization or there is need to investigate additional criteria.

To asses these two aspects of question RQ2 we state the following null hypotheses.

- $H02A$: There is no statistically-significant difference between the fixing time of external CRs and that of internal CRs.
- $H02BInt$: There is no relation between the fixing time of internal CRs and the severity of the CRs.
- $H02BExt$: There is no relation between the fixing time of external CRs and the severity of the CRs.

The third aspect of RQ2 is to evaluate if CRs are handled according to the timing requirements defined in the WLA of Ericsson (not shown in this paper for confidentiality reasons). This is critical, especially, for external CRs. Failing to meet these requirements can expose Ericsson to additional costs. We compare the duration of CR fixing time to the deadlines defined in the WLA for each CR type and severity level. We compute the percentage of CRs that meet the WLA requirements. Knowing this will help Ericsson evaluate if the actual WLA deadlines for CR fixing should be maintained, or if there is a need to do a re-evaluation.

Variables: To compare the CR fixing time with respect to their type we use as independent variable the type T_i of a CR CR_i, to distinguish between internal and external CRs. We consider as dependent variable the fixing time FT_i of a CR. We compute the fixing time FT_i of a CR CR_i . The fixing time FT_i is the time between when the CR is registered and when a technical answer is provided. To analyze the impact of the severity of the CR on the fixing time, we consider as independent variable the CR severity and as dependent variable the CR fixing time FT_i. In addition, we compare the time spent in average to answer a CR to the time defined in the WLA and we analyze the proportion of CRs which satisfy the WLA requirements.

Analysis method: We analyze RQ2 in three steps. First, we compute the (non-parametric) Mann-Whitney test to compare the CR fixing time with respect to the CR type and analyze whether the difference in the average fixing time is statistically significant. We use the Mann-Whitney test because, as a non-parametric test, it does not make any assumption on the underlying distributions. We analyze the results of the test to assess the null hypothesis H_{02A}. The result is considered as statistically significant at $\alpha = 0.05$. Therefore, if p-value < 0.05, we reject the null hypothesis H_{02A} and conclude that the average fixing time of internal CRs is significantly different from the average fixing time of external CRs. Other than testing the null hypothesis, we also estimate the magnitude of the difference in average between the fixing time of the two types of CRs. We use the non-parametric effect size measure Cliff's d, which indicates the magnitude of the effect size of the treatment on the dependent variable. Cliff's d is defined as Eq. 1 [8]:

$$d = \frac{\#(y_i > x_j) - \#(y_i < x_j)}{n_y . n_x} = \frac{\sum_i \sum_j d_{ij}}{n_y . n_x} \qquad (1)$$

where $d_{ij} = sign(y_i - x_j)$ and $\#(y_i > x_j)$ the number of comparisons between observations in the two groups for which the Group i observation is larger than the Group j observation. The effect size is small for $0.147 \leq d < 0.33$, medium for $0.33 \leq d < 0.474$, and large for $d \geq 0.474$ [8].

Second, we analyze for each type of CRs the impact of severity on the fixing time. We use Kruskal-Wallis one-way analysis of variance by ranks to analyze the impact of the type and severity of CRs on the fixing time. The Kruskal-Wallis test aims at testing the hypothesis that several populations have the same continuous distribution versus the alternative that measurements tend to be higher in one or more of the populations. We want to see if there is a statistically significant difference between the fixing time according to the CR severity and type. In other words, does the fixing time of the CR increase or decrease according to its severity? Kruskal-Wallis one-way analysis of variance by ranks is a non-parametric test that does not require making assumption about the data set distribution. Using Kruskal-Wallis test, we test the null hypotheses H_{02BInt} (respectively H_{02BExt}) to see if there is a statistically-significant difference between the fixing time of internal CRs (respectively external) according to their severity. The results of the Kruskal-Wallis test are considered statistically significant at $\alpha = 0.05$. If p-value < 0.05, we reject the null hypothesis H_{02BInt} (respectively H_{02BExt}) and conclude that the severity of internal CRs (respectively external) impact significantly its fixing time. Third, we compare the fixing time of the CRs to the deadlines specified in the WLA. We compute the percentage of CRs (according to their type and severity) that meet the WLA requirements.

RQ3: How fast are CRs Assigned?

After working with Ericsson software engineers and examining a large number of CRs, we have observed that the CR assignment at Ericsson tends to be considerably time consuming. Bhattacharya et al. [4] also found that CR assignment is problematic despite the quality of the CR data. At Ericsson, there are many factors that impact the assignment time including the size of the organization and the large number of its products and customers [10]. The assignment activity takes place before starting solving the problem. It could be a bottleneck task if it is performed inefficiently (i.e., without automation). Assigning a CR to the proper organizational unit is a challenging task. We have therefore decided to study the relation between the CR type and the duration of CR assignment activity. Similar to the previous questions, we took into account the severity levels. This will help assess the importance of the severity attribute during CR assignment.

We analyze whether the assignment of external CRs takes more time than the assignment of internal CRs by testing the null hypothesis:

- H_{03A}: There is no statistically-significant difference between the duration of the assignment of external CRs and that of internal CRs.

In this research question, we analyze the duration of the assignment activity with respect to the severity. We test the following null hypotheses:

- H_{03BInt}: There is no relation between the duration of internal CRs assignment and the CR severity.
- H_{03BExt}: There is no relation between the duration of external CRs assignment and the CR severity.

Variables: We use as independent variable the type T_i of a CR (CR_i). For the dependent variables, we measure the assignment time, AT_i , of the CRs. We compute the time spent between the time the CR is registered until it is assigned to the right designer to be fixed. When analyzing the impact of the severity of the CR on the assignment time, we consider as independent variable the severity of the CR and as dependent variable the CR assignment time, AT_i.

Analysis method: First, we compute the (non-parametric) Mann-Whitney test to compare the CR assignment time with respect to the CR type (as in RQ2). This will show if a particular type of CRs is assigned faster than the other one. We test the null hypothesis H_{03A} to know whether the difference in average between the assignment time of internal CRs and that of external CRs is statistically significant. Second, we analyze if the severity of a CR impacts its assignment time. We use Kruskal-Wallis one-way analysis of variance by ranks to analyze for each type of CR (internal, external), the impact of the severity of CR on the assignment time. We test the null hypotheses $H03BInt$ and $H03BExt$ according to the results of Kruskal-Wallis one-way analysis of variance by ranks. The results are considered as statistically significant at $\alpha = 0.05$. Therefore, if p-value < 0.05, we reject the null hypothesis $H03BInt$ (respectively $H03BExt$) and conclude that the severity of internal (respectively external) CRs impact significantly the assignment time.

IV. Study Results and Discussion

In this section, we report on the results of the analyses performed to answer the research questions. As we showed in Section 2, we used a CR dataset covering two years of maintenance activities on one Ericsson largest systems. For confidentiality reasons, throughout this paper, we do not reveal the exact number of CRs, the CR fixing and assignment time, and the WLA required fixing time. Instead, we use relative data (i.e., percentages).

A. What is the proportion of internal vs. external CRs?

Figure 2 shows the percentage of internal and external CRs. As shown in the figure, we found that 64% of the total reported CRs are internal, i.e., detected during integration system testing, whereas only 36% of the total CRs are external, i.e., sent by customers. In other words, the percentage of internal CRs is almost twice the percentage of external CRs. On one hand, this finding is considered positive and reflects good performance of the testing teams, especially given such a large organization and that the system under study is one of the largest systems at Ericsson. On the other hand, 36% of external crashes is still considered high and should require acute attention from the management team.

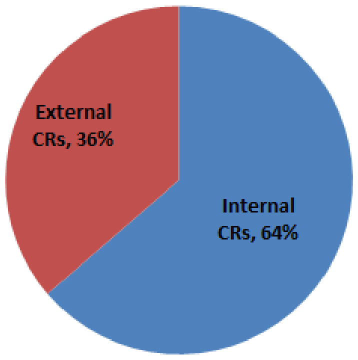


Fig. 2. Percentages of internal and external CRs

Figure 3 shows the proportion of internal and external CRs by severity. To assess the null hypothesis H01, we performed the Pearson's chi-squared independence test.

According to the results of the test (p-value < 0.001), we reject the null hypothesis H01 and conclude that there is a significant different between the severity of internal CRs and external CRs.

An interesting observation is that the percentage of internal CRs of severity "A" (16%) is higher than the percentage of external CRs of severity "A" (12%). This means that the designers find internally the most critical crashes before the product is released to the customer. Also when analyzing the percentage of CRs of severity "A" for both internal and external CRs, we can observe that the percentage of CRs of severity "A" is lower which could indicate overall a good system quality process. However, because of the critical nature of faults of severity "A" and their cost implications, it is still recommended to conduct additional studies to further reduce the percentage of CRs of this severity level.

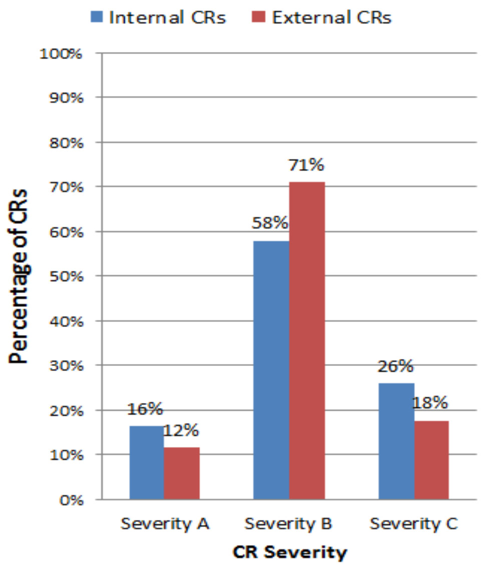


Fig. 3. Proportions of internal and external CRs by severity

B. How fast are CRs fixed?

We analyze the difference in the fixing time between internal and external CRs. We conduct Mann-Whitney test to assess $H02A$. Table 1 reports on the results of the Mann-Whitney test and Cliff's d effect size.

TABLE 1: MANN-WHITNEY TESTS AND CLIFF'S D RESULTS FOR ANSWERING TIME OF INTERNAL CRS COMPARED TO EXTERNAL CRS

	Median	M-W p	Cliff's d
Internal CRs	3	**< 0.01**	0.56
External CRs	4		

The results show that the difference between the fixing time of internal CRs and the fixing time of external CRs is statistically significant with large effect size.

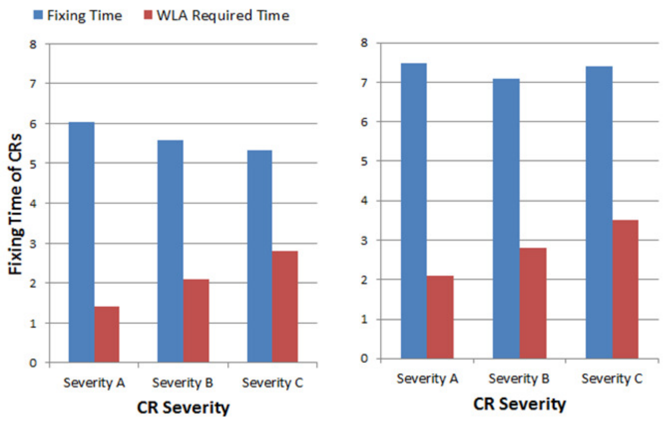
Therefore, we can reject the null hypothesis $H02A$ and conclude that the fixing of external CRs takes more time with large effect size than the fixing of internal CRs.

The fixing of internal CRs takes less time than the external CRs. This observation can be explained by the fact that, as expected, design teams have more knowledge about the system than external users and therefore can provide relevant information to understand the crash, locate the fault, and fix it. In addition, at Ericsson, external CRs cannot normally be answered until a team of reviewers have acknowledged the proposed answer. This team does not meet daily. This may explain the delay in closing external CRs.

To understand the relation between the CR severity and CR fixing time, we compute the test of Kruskal-Wallis one-way analysis of variance by ranks. The result of the test for internal CRs (p – value = 0.4734) and the result for external CRs (p–value = 0.6987) show that there is no relation between the severity of a CR and the fixing time.

Therefore, we cannot reject the null hypotheses $H02BInt$ and $H02BExt$ and we conclude that there is no effect of the severity of CRs on the fixing time.

We analyze the impact of the CR severity on the fixing time. Figure 4 shows the difference in fixing time for internal and external CRs according to the severity and the WLA required fixing time.



(a) Internal CRs (b) External CRs

Fig. 4. Fixing time and WLA required time for internal and external CRs in days—the time unit is days mapped to an interval between 1 to 10 to protect the confidentiality ofthe original data

This result is surprising. We expected the opposite. Intuitively, the more severe a CR, the less time it should take to fix it. It is more logical to answer faster a CR of severity "A" before the other CRs because of the impact of the crash on the system. As mentioned earlier, a crash of severity "A" affects more than 30% of the traffic or may cause a network failure.

In addition, as we can see from Figure 4(a), the fixing time of crashes of severity "A" for both types of CRs is higher than the fixing time of crashes of severity "B" and "C". This finding seems to indicate that the designers do not consider the severity attribute as submitted by the CR submitters. According to the discussion we had with software engineers at Ericsson, we found out that the reason for which the severity level does not impact the fixing time is due to three main factors. The submitter may assign a high severity level to the CR just to speed up the answering process to receive a fix within reasonable time. This seems to happen for external CRs more than the internal ones.

The second factor lies in the fact that the submitter (internal or external) has little knowledge of the severity classification scheme. At the present time, there are no tools that could help identify the severity level by observing the effect of the crash on the network traffic.

The third factor is related to how the design teams in different organizational units function at Ericsson. There seems to be a lack of consistency on how the severity levels are defined (and interpreted). This has led some designers to use their own CR prioritization criteria that do not always map to the severity classification defined at the corporate level. These results can help direct managers about the necessity to conduct further studies to understand how internally each team prioritizes the CRs. The ultimate objective is to set standards and guidelines that could be used by all the design teams to prioritize consistently the CRs. Another recommendation that emerged from these findings is the necessity to invest in tools for fault diagnosis that could automatically suggest adequate severity levels by observing the impact of faults on network traffic. Such recommendation systems can reduce the time and efforts spent on analyzing fault severity. They can also help standardize the way severity levels are reported.

Finally, we compare the CR fixing time by severity to the WLA required fixing time. Figure 4 shows for each severity level the average fixing time and the WLA required fixing time for internal and external CRs. Then, we analyze the percentage of CRs with respect to the WLA required fixing time (see Figure 5). Figure 4 shows that, in average, the time it takes to fix a CR exceeds the required time according to WLA for each CR type and severity level. For internal CRs, the fixing time of CRs of severity "A" takes four times longer than what is required in the WLA. Similarly, the fixing of CRs of severity "B" and "C" are in average two times longer. This finding is almost the same for external CRs.

We also found that the severity is not a discriminating factor that impacts the percentage of CRs that respect the WLA. We note that the rate of the CRs with severity "A" that meet the WLA requirements is the lowest compared to other severity levels (29% internal and 31% external CRs) (see Figures 6 and 7). In fact, for all CR types and severity levels, the ratio of the CRs that satisfy WLA does not exceed 50%. This is problematic giving that not being able to meet the WLA may be costly to Ericsson.

These results are consistent with the results of previous research questions that suggest that prioritization of CRs is not performed on the basis of severity. As discussed earlier, this could be explained by the consistency and the relevance of the severity attribute as well as the way severity levels are used (and interpreted) by different design teams.

Another important factor that contributes to these findings is concerned with the WLA requirements themselves. Crashes that are truly of severity "A" are usually caused by faults that are difficult to detect and fix. Paradoxically, the WLA requires for such crashes to be addressed the fastest way possible. That said, WLA requirements seem to have been designed with a focus on the business value of handling CRs with less attention to the technical implications of fixing severe crashes. Combined with lack of diagnosis tools, it becomes very challenging for design teams to properly evaluate the importance of a crash, and even more challenging to estimate the time it would take to bring the fixes, which defeats the purpose of having predefined fixing times such as the ones in WLA in the first place.

Based on these findings, it becomes clear that there is a need for consistent CR prioritization criteria based on work team habits and the complexity of the crashes. Another recommendation is to review the WLA requirements and adjust them to take into account the complexity of the failures, a problem that is easier said than done, given the inherent challenges in assessing failure complexity.

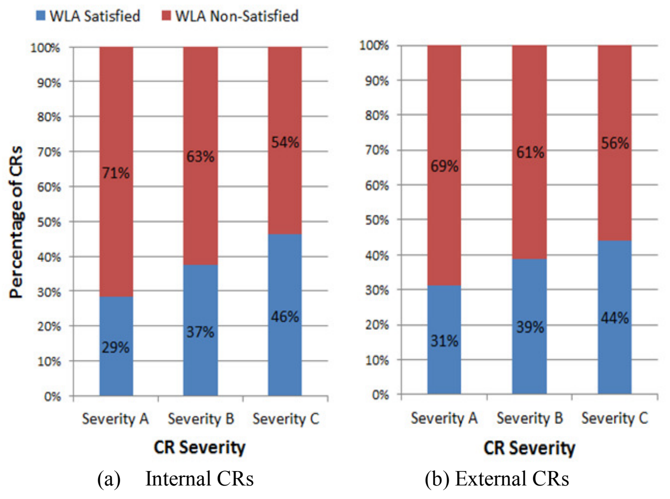


Fig. 5. Percentage of internal CRs satisfying the WLA required time.

C. How fast are CRs Assigned?

We conduct the Mann-Whitney test to assess the null hypothesis H_{03A} on the difference between assignment time according to the type of CRs. Table 2 reports the results of the Mann-Whitney test and Cliff's d effect size to compare the assignment time of internal CRs and the assignment time of external CRs.

TABLE 2. MANN-WHITNEY TEST AND CLIFF'S D RESULTS FOR ASSIGNMENT TIME OF INTERNAL CRS COMPARED TO EXTERNAL CRS

	Median	M-W *p*	Cliff's *d*
Internal CRs	0.5	< 0.01	0.36
External CRs	0.9		

The results show that the difference between the assignment time of internal CRs and the assignment time of external CRs is statistically significant with medium effect size.

Therefore, we can reject the null hypothesis H03A and conclude that the assignment of external CRs takes more time with medium effect size than the assignment of internal CRs.

The assignment of CRs is a triaging task. A triaging team uses the information provided when the crash is reported to redirect the CRs to the right development teams. The assignment is performed quickly if the CR is described in sufficient details to help triagers identify the right developer. It is expected that internal CRs are assigned faster than external CRs. This is because, unlike customers, developers have better knowledge of the system and the type of faults.

Furthermore, we analyze the impact of the CR severity on the assignment time. Figure 6 shows the difference in assignment time for internal and external CRs according to the CR severity. We compute the test of Kruskal-Wallis one-way analysis of variance by ranks. For external CRs, the result of Kruskal-Wallis (p-value=0.2441) shows that the severity does not impact the assignment time while for the internal CRs the result of the test (p- value < 0.05) shows that the severity has a significant impact on assignment time.

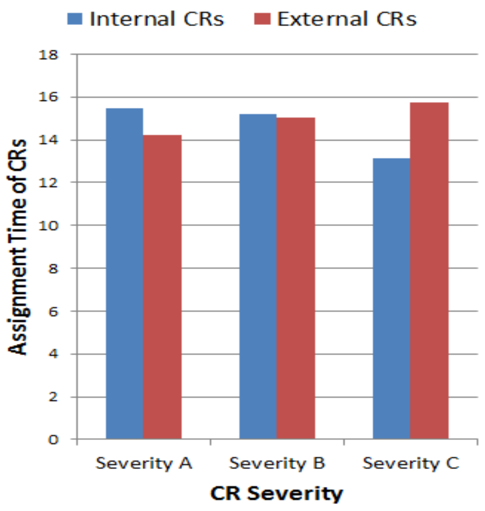


Fig. 6. Assignment time by CR severity for internal and external CRs (the y-axis shows the number of days, mapped to an interval of 1 to 20 for confidentiality reasons).

Therefore, we can reject the null hypothesis $H03BInt$ and conclude that when assigning internal CRs, designers are taking into account the severity of the CR. On the contrary, we cannot reject the null hypothesis $H03BExt$ and we conclude that there is no evidence of the impact of CR severity on the assignment of external CRs.

We observe that the assignment time of internal is impacted significantly by the CR severity. One might expect that the more severe the CR, the faster it is assigned. We found, however, that CRs of severity "A" take more time than CRs of severity "B" and "C". This may be due to the complexity of the severe CRs, but does not justify, at least from the business standpoint, the slow reaction. On the contrary, we found that there is no significant impact of CR severity on the assignment time of external CRs. Even if the difference is not statistically significant for external CRs, the trend between the assignment time according to the CR severity is the opposite for internal CRs. This is again due to the relevance and the consistency of the use of the severity attribute as a CR prioritization criterion as discussed in RQ2.

D. Discussion

Severity of crash reports: As we showed in this study, the severity of crash reports is not a good indicator for prioritizing crashes. Similar findings are noted by other researchers when working on open source systems (see the work by Khomh et al. [10]). While severity may be well defined, it is used inaccurately and interpreted differently by various stakeholders (customers, developers, and managers). In addition, we believe that customers should not be asked to specify the severity of crashes. It is at their advantage to indicate that a crash has a high severity to receive fixes within reasonable time. One solution to overcome this issue is to have automated solutions that can determine the severity of crashes by monitoring, for example, network traffic. An alternative solution would be to assign this task to CR triagers. Crash reporting systems should generate data that can facilitate this task.

Need for fault diagnosis tools: Not all faults are the same. The discussions we had with a team of five Ericsson software engineers suggest that a better way to handle crashes is to group the faults into categories, and use the resulting classification to improve the assignment and fixing time of CRs. Possible classification criteria include the impact of the fault on the system's functional and non-functional requirements. For example, we can group faults that impact system availability into one category, since availability is usually managed through dedicated middleware. Knowing these categories in advance can not only help with CR triaging tasks, but also help build automated fault diagnosis tools. These tools can collect additional data that can later be automatically attached to the reports when a certain fault category is detected. The idea is to work towards a crash reporting system that is 'intelligent' enough to include data that is most relevant to the fault at hand.

Quality of the crash reports: The quality of crash report descriptions is another cause of delaying the fixing of crash reports. Despite the fact that Ericsson has in place well defined data quality guidelines[4] for reporting crashes, many of the crash reports we examined in this study do not follow these guidelines and sometimes even omit essential information. We believe that this is due to many factors. The first one is lack of training of the users of these guidelines (crash reporters and submitters). The second reason is that these guidelines are not enforced in crash reporting tools (e.g., MHWeb). For example, simple completeness and consistency checks could have improved the quality of many of the reports we have examined. Another important factor that can improve the quality of CR descriptions is to have consistent terminology. This is can be achieved by building a common data dictionary.

Moreover, we believe that CR descriptions should be combined with more formal crash report data such as crash traces for a better categorization of CRs. This is aligned with the survey conducted by Bettenberg et al. in [3], in which the authors found that crash traces and more formal data should be used for triaging and CR handling in general. An example of a technique that detects duplicate reports based on traces is the one proposed by Kim et al. in [26]. Traces, however, can be challenging to analyze because of their large size ([9]). Techniques for trace reduction through abstraction such as the ones presented in [16, 17] may be needed.

[4]For confidentiality reasons, we cannot reveal the content of the guidelines.

Working level agreements: The fact that more than 50% of the CRs do not comply with the WLA requirements is an alarming issue. On one hand, this can be addressed by having automated fault diagnosis tools and improving the various aspects of handling CRs as discussed earlier. The process of handling external CRs, in particular, can reduce the impact of this issue, since external CRs require a team of reviewers to approve the resolutions. This team does not meet on a daily basis, which may cause significant delays. On the other hand, there is an organizational dimension to this issue. At the managerial level, a WLA is used for performance evaluation. Many of the performance indicators in a WLA were designed with a business objective in mind. There seems to be a gap between the business objectives and the implementation of these objectives at the operational level. Therefore, we conjecture that the solution must involve both management and technical staff to find appropriate performance indicators from both the business and operational (technical) sides.

V. Threats to validity

We discuss the threats to the validity of our study following the guidelines for case study research [22].

Construct validity: Construct validity threats concern the relation between theory and observation. In our study, the construct validity threats are mainly due to measurement errors and how we build our dataset. We extract CR information and history by parsing the html files of the CR coming from MHWeb. We rely on information provided by users on MHWeb. Some designers may fix the CR long time before changing the CR status in MHWeb and this may bias our data.

Internal Validity: Threats to internal validity do not affect our study, being an exploratory study [22]. We do not claim causation; we only report the observations from the results of our study and discuss some explanation.

Conclusion Validity: Conclusion validity threats concern the relation between the treatment and the outcome. We paid attention not to violate assumptions of the performed statistical tests. We used non-parametric tests that do not make any assumption about the data set distribution (i.e., Mann-Whitney tests and Cliff's d effect size, Kruskal-Wallis tests etc.). We used two years activities crash reports from one large system from Ericsson involving thousands of designers. Again we do not claim causality, we report the results of our observation and discuss and explain these results.

External Validity: Threats to external validity concern the possibility to generalize our results. Ericsson is a very large software organization and the system under study is one of the largest systems and we take CRs that cover two years of activities, which we believe is a very representative dataset. However, this is related to an industrial context. In future work, we will conduct the same study on open source projects to compare the results.

VI. Related Work

In what follows, we summarize related studies on crash report handling and analysis with a focus on studies that deal with reducing the bug fixing lead time. We discuss what distinguishes our work from the existing literature at the end of the section.

Zaman et al. [23] conducted a case study on the Firefox project and analyzed the bugs lead time by categorizing the bugs based on whether they are caused by security attacks or due to performance degradation. They found that security bugs are fixed and triaged much faster than performance bugs.

Marks et al. [14] studied the bug fixing time in open source projects based on three categories: the bug location, the reporter of the bug, and the description of the bug. Their main finding is that the factors that affect the bug fixing time vary according to the project and over time. Weiss et al. [21] proposed a technique based on mining past bug reports to find similar bugs in new versions. They have also worked on predicting the fixing time by using K-Nearest Neighbour (kNN) clustering algorithm. Panjer [15] conducted as case on study on Eclipse project and built a prediction model of the bug lifetime. They found that the most important factors that influence the bug lifetime are the comments, the severity of the bug determined by the development team and the product. Tian et al. [27] proposed to improve the detection of bug reports by having better ways to compute the similarity between two reports.

Anbalagan and Vouk [2] conducted an empirical study on the time taken to correct bug in an open source project (i.e., Ubuntu distribution). They build a prediction model of the bug fixing time. They found that most of the bugs are corrected by people in small groups. They also found that there is a relation between the number of people involved in a bug report and the time taken to fix the bug. Similarly Giger et al [7] investigated using open source projects (i.e., Eclipse, Mozilla, and Gnome) the relation between bug reports characteristics and bug fixing time. They build a prediction model for the bug fixing time. They reach a precision of 0.65 and a recall 0.69 when predicting eclipse bug fixing time.

Canfora et al. [5] used a survival model to analyze the time it takes to fix a bug. They conducted a case study on Eclipse, Mozilla, OpenLDAP and Vuze projects and found that long-lived bugs can be characterized by changes to specific code constructs. Bhattacharya et al. [4] conducted a study on the bugs in the Google android platform and android open-source applications. They found that the bug triage is problematic despite the high quality of bug report. They also found that fixing security bugs takes more time than others.

Dhaliwal et al. [6] conducted a study of crash-reports from Mozilla Firefox and found that grouping crash reports triggered by multiple bugs takes longer time to be fixed than the bugs where crash reports triggered by each bug are grouped separately. They proposed a grouping approach based on groups that contain the crash reports triggered by only one bug. Kim et al. [12] proposed crash reports grouping approach based on crash graphs consisting on an aggregated view of multiple crashes. They showed that the crash graphs could reduce misclassification and help predict fixable crashes. Soh et al. [20] conducted a study using open source projects on understanding how developers spend their effort during maintenance activities. They showed that there is no relation between the complexity of the fixing tasks and the effort spent by maintainers. However, the maintainers spent most of the time exploring files which are not relevant to fixing of the bug. Zhang et al. [24] studied using open source projects the delays between the bug assignment and the bug fixing. They found that the delays in bug fixing are due mainly to the bug type, the severity of the bugs, the operating system, the description of the bugs, and the comments of the bugs. Podgurski et al. [18] presented a clustering approach for automatic classification and prioritization of CRs. They applied their approach to open source compilers. Although their work is not aimed at investigating the CR process lead time (which is the objective of this paper), the authors argued that prioritization of CRs should be automated to allow incoming CRs to be properly prioritized.

Aggarwal et al. [25] showed that domain knowledge, if available, can improve the detection of duplicate reports. They developed a method to extract contextual word lists from software-engineering literature and use the list to deduplicate crash reports.

The problem of CR prioritization has also been tackled by Kim et al. in [13]. Through an empirical investigation of Firefox crash reports, the authors showed that only 10 to 20 crashes account for the large majority of crash reports. Therefore, predicting these crashes should improve the CR process handling time.

In summary, most of the studies conducted so far are limited to open source systems. Many of these studies have also a different scope than ours. In this paper, the emphasis is on examining the differences between the way internal and external CRs are handled at Ericsson using the CR severity levels as well as the WLA required fixing time. This is because the handling of each CR type has different implications on Ericsson's operational costs. Providing insights on the way CRs are handled and pinpointing the key challenges faced by software engineers when dealing with each type of CR can help Ericsson put in place the proper strategies and devote the right resources to overcome these challenges.

VII. CONCLUSION AND FUTURE WORKS

We performed an empirical study at Ericsson to analyze: (1) the importance of internal CRs vs. external CRs; (2) the impact of the type and severity of CRs the fixing time and the level of satisfaction in delivering the answer to CR according to the WLA; (3) the impact of the type and severity of CRs on the assignment time. We performed our study on a dataset of CRs resulting from two years of activities on one of the large industrial systems at Ericsson. The key findings are: (a) it takes more time to fix external CRs compared to internal CRs; (b) the severity attribute does not have an impact on the fixing time; (c) the assignment time of internal CRs is less than that of external CRs; and (d) less than 50% of CRs are answered within the organization's fixing time requirements defined in WLA. We attribute these results to many factors. First, the severity attribute does not seem to be used consistently across the organization including the customers, which often leads to poor prioritization of CRs. The second factor is related to the fact that it is often challenging to provide an adequate description of the fault. This is mainly due to lack of automated diagnosis tools. It is also caused by the poor quality of the data provided when submitting a report, despite the existence of corporate data collection guidelines. Finally, we have noticed that design team across organizational units tend to adopt their own work practices which may result in inconsistencies in the way CRs are prioritized and handled.

To build on this work, we need to gain more comprehensive knowledge on the faults and the type of faults that are reported. This would allow us to develop criteria to be used as guidance for software engineers and customers when prioritizing the CRs. This knowledge can also be used to design automatic fault diagnosis tools. Moreover, we need to understand how the quality of the data in the CRs can be improved to accelerate the assignment of CRs to the design teams. The ultimate goal of these studies altogether is to improve the CR fixing time. Finally, we need to work with software engineers at Ericsson to study how design teams work internally and propose standardized practices to enforce any existing (or new) guidelines for CR management.

ACKNOWLEDGMENT

We would like to thank the software engineers at Ericsson, Stockholm, Sweden, for their valuable feedback and their active participation in this study. This research is partly supported by MITCAS Canada, and Ericsson.

REFERENCES

[1] N. I. of Standards & Technology, “The economic impacts of inadequate infrastructure for software testing,” 2002, US Department of Commerce

[2] P. Anbalagan and M. Vouk, “On predicting the time taken to correct bug reports in open source projects,” in *Proc. of the International Conference on Software Maintenance (ICSM’09)*, pp. 523–526, 2009.

[3] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, T. Zimmermann, "What makes a good bug report?," in *Proc. of the 16th ACM SIGSOFT International Symposium on Foundations of Software engineering*, pp. 308-318, 2008.

[4] P. Bhattacharya, L. Ulanova, I. Neamtiu, and S. Koduru, “An empirical analysis of bug reports and bug fixing in open source android apps,” in *Proc. of the 17th European Conference on Software Maintenance and Reengineering (CSMR’13)*, pp. 133–143, 2013.

[5] G. Canfora, M. Ceccarelli, L. Cerulo, and M. Di Penta, “How long does a bug survive? an empirical study,” in *Proc. of the 18th Working Conference on Reverse Engineering (WCRE’11)*, pp. 191–200, 2011.

[6] T. Dhaliwal, F. Khomh, and Y. Zou, “Classifying field crash reports for fixing bugs: A case study of mozilla firefox,” in *Proc. of the 27th International Conference on Software Maintenance (ICSM’11)*, pp. 333–342, 2011.

[7] E. Giger, M. Pinzger, and H. Gall, “Predicting the fix time of bugs,” *In Proc. of the 2nd International Workshop on Recommendation Systems for Software Engineering, (RSSE’10)*, pp. 52–56, 2010.

[8] R. Grissom and J. Kim. *Effect Sizes for Research: A Broad Practical Approach.* Lawrence Erlbaum Associates, 2005.

[9] A. Hamou-Lhadj, "Techniques to Simplify the Analysis of Execution Traces for Program Comprehension," *Ph.D. Dissertation, School of IT and Engineering (SITE),* University of Ottawa, 2005.

[10] F. Khomh, B. Chan, Y. Zou, A. Hassan, "An Entropy Evaluation Approach for Triaging Field Crashes: A Case Study of Mozilla Firefox," in *Proc. of 18th Working Conference on Reverse Engineering (WCRE)*, pp. 261 - 270, 2011.

[11] L. Jonsson, D. Broman, K. Sandahl, and S. Eldh, “Towards automated anomaly report assignment in large complex systems using stacked generalization,” in *Proc. of the 5th Conference on Software Testing, Verification and Validation (ICST’12),* pp. 437–446, 2012.

[12] S. Kim, T. Zimmermann, and N. Nagappan, “Crash graphs: An aggregated view of multiple crashes to improve crash triage,” in *Proc. of the 41st International Conference on In Dependable Systems Networks (DSN’11),* pp. 486–493, 2011.

[13] D. Kim, X. Wang, S. Kim, A. Zeller, “Which Crashes Should I Fix First?: Predicting Top Crashes at an Early Stage to Prioritize Debugging Efforts,” *IEEE Transactions on Software Engineering, 37(3),* pp. 430 - 447, 2011.

[14] L. Marks, Y. Zou, and A. E. Hassan, “Studying the fix-time for bugs in large open source projects,” in *Proc. of the 7th International Conference on Predictive Models in Software Engineering (PROMISE’11),* pp. 1–1,2011.

[15] L. Panjer, “Predicting eclipse bug lifetimes,” in *ICSE Workshop on Mining Software Repositories (MSR’07),* pp. 29–29, 2007.

[16] H. Pirzadeh, A. Hamou-Lhadj, "A Software Behaviour Analysis Framework Based on the Human Perception System," in *Proc. of the 33rd International Conference on Software Engineering (ICSE'12),* pp. 948-951, 2011.

[17] H. Pirzadeh, A. Hamou-Lhadj, "A Novel Approach Based on Gestalt Psychology for Abstracting the Content of Large Execution Traces for Program Comprehension," in *Proc. of the 16th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS '11),* pp. 221-230, 2011.

[18] A. Podgurski, D. Leon, P. Francis, W. Masfi, M. Minch, “Automated support for classifying software failure reports,” in *Proc. of the International Conference on Software Engineering (ICSE’03),* pp. 465-475, 2003.

[19] R. S. Pressman. Software Engineering – A Practitioner’s Approach. McGraw-Hill Higher Education, 5th edition, November 2001.

[20] Z. Soh, F. Khomh, Y.-G. Guéhéneuc, and G. Antoniol, “Towards understanding how developers spend their effort during maintenance activities,” in *Proc. of the Working Conference on Reverse Engineering,* pp. 152–161, 2013.

[21] C. Weiss, R. Premraj, T. Zimmermann, and A. Zeller, “How long will it take to fix this bug?” in *Proc. of the 4th International Workshop on Mining Software Repositories (MSR’07),* 2007.

[22] R. K. Yin. *Case Study Research: Design and Methods (Applied Social Research Methods).* Sage Publications, Inc, 2008.

[23] S. Zaman, B. Adams, and A. E. Hassan, “Security versus performance bugs: A case study on Firefox,” in *Proc. of the 8th Working Conference on Mining Software Repositories (MSR’11),* pp. 93–102, 2011.

[24] F. Zhang, F. Khomh, Y. Zou, and A. E. Hassan, “An empirical study on factors impacting bug fixing time,” in *Proc. of the Working Conference on Reverse Engineering (WCRE’12),* pp. 225-234, 2012.

[25] K. Aggarwal, T. Rutgers, F. Timbers, A. Hindle, "Detecting duplicate bug reports with software engineering domain knowledge," in *Proc. of the IEEE 22nd International Conference on Software Analysis, Evolution and Reengineering (SANER’*15), pp. 211-220, 2015.

[26] S. Kim, T. Zimmermann, N. Nagappan, "Crash Graphs: An Aggregated View of Multiple Crashes to Improve Crash Triage," in *Proc. of the 2011 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN’11),* 2011.

[27] Y. Tian, C. Sun, and D. Lo, “Improved Duplicate Bug Report Identification,” in *Proc. of 16th European Conference on Software Maintenance and Reengineering,* pp. 385–390, 2012.