Characterizing the Differences Between Pre- and Post- Release Versions of Software

Paul Luo Li[+#], Ryan Kivett[+], Zhiyuan Zhan[+], Sung-eok Jeon[+], Nachiappan Nagappan[*], Brendan Murphy[*], Andrew J. Ko[#]

[+]Microsoft Windows Reliability Team
One Microsoft Way
Redmond, WA USA
pal,ryankiv,zhizhan,sujeon@microsoft.com

[*]Microsoft Research
Redmond, WA USA
nachin,bmurphy@Microsoft.com

[#]The Information School | DUB Group
University of Washington
Seattle, WA USA
ajko@UW.edu

ABSTRACT

Many software producers utilize beta programs to predict post-release quality and to ensure that their products meet quality expectations of users. Prior work indicates that software producers need to adjust predictions to account for usage environments and usage scenarios differences between beta populations and post-release populations. However, little is known about how usage characteristics relate to field quality and how usage characteristics differ between beta and post-release. In this study, we examine application crash, application hang, system crash, and usage information from millions of Windows® users to 1) examine the effects of usage characteristics differences on field quality (e.g. which usage characteristics impact quality), 2) examine usage characteristics differences between beta and post-release (e.g. do impactful usage characteristics differ), and 3) report experiences adjusting field quality predictions for Windows. Among the 18 usage characteristics that we examined, the five most important were: the number of application executed, whether the machines was pre-installed by the original equipment manufacturer, two sub-populations (two language/geographic locales), and whether Windows was 64-bit (not 32-bit). We found each of these usage characteristics to differ between beta and post-release, and by adjusting for the differences, accuracy of field quality predictions for Windows improved by ~59%.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification - *Reliability, Statistical Methods, Validation*

D.2.8 [**Software Engineering**]: Metrics – *Product Metrics*

General Terms

Human Factors, Measurement, Reliability, Verification

Keywords

Windows, Customer experience improvement program, Windows Error Reporting (WER), Reliability Analysis Component (RAC), Usage, Beta

1. INTRODUCTION

Many commercial (e.g. Apple®, Adobe®, and Microsoft®) and open-source (e.g. Firefox®, MySql®) software producers utilize beta programs (i.e. distribution of free pre-release software to volunteer or selected users) to predict post-release quality and to ensure that their products meet quality expectations of users. Software producers commonly use quality measured on beta versions as the predicted post-release quality (i.e. field quality) and address issues found in beta versions to improve field quality. Misleading field quality predictions can lead software producers to take sub-optimal actions, such as basing the decision to release on the misleading predictions and releasing low-quality software that do not meet quality expectations of users.

Prior work indicates that software producers need to adjust field quality predictions to account for usage environments and usage scenario differences between beta (i.e. pre-release) and post-release populations [11]. Approaches for adjusting measurements, such as screening beta users or adjusting measurements statistically [22], need information on which usage characteristics to target. However, little is known about how usage characteristics relate to field quality and how usage characteristics differ between pre-release and post-release, which makes adjusting for pre-release and post-release differences difficult to do with confidence.

In this study, we focus on three aspects of field quality: application crashes, application hangs, and system crashes. We:

- Examine the effects of usage characteristic differences on field quality: Which usage characteristics are the most impactful to field quality? How do they affect field quality? Why do they affect field quality?
- Examine usage characteristics differences/similarities between pre-release and post-release machines: Which usage characteristics differ or remain similar? If they differ, how and why?
- Report experiences using the *Usage Profile-based Reliability Measurement Calibration* (UPRMC), introduced by Xue et al. in [22], to adjust field defect predictions for Windows: How do adjustments improve field quality predictions?

We analyzed anonymous failure and usage data from millions of pre-release and post-release Windows7® machines in the Customer Experience Improvement Program [12]. We examined field quality data captured through the Windows Error Reporting (WER) system [6] and the Reliability Analysis Component (RAC) [9], explained in more detail in Sections 3.1 and 3.2. Among the various usage characteristics that we examined, the five most important were:

1. **Number of applications executed**: machines executing more applications had higher rates of failures
2. **Install Type: Pre-installed by the Original Equipment Manufacturer (OEM)**: machines that were pre-installed by the

OEM had lower rates of failures compared to other methods of installations (likely due to users that utilize this Install Type)

3. **Sub-population: Locale 2:** machines belonging to one the key language/geographic locales had lower rates of failures compared with the world-wide average
4. **64-bit (not 32-bit) Windows:** machines running 64-bit Windows had lower rates of failure compared to machines running 32-bit Windows
5. **Sub-population: Locale 6** machines belonging to another of the key language/geographic locales had higher rates of application crashes and application hangs, but lower rate of system crashes compared with the world-wide average

We found each of these usage characteristics to differ between pre-release and post-release machines. Using UPRMC to adjust for the differences, accuracy of Windows' field quality prediction improved by ~59%.

Our work makes three contributions to the state of knowledge. First, using data from millions of machines world-wide, we identified five usage characteristics that are impactful to field quality, quantified their effects, and identified possible causes for their impact. Second, using empirical data, we verified and described usage characteristics differences between pre-release and post-release, as well as identified possible causes for the differences. Third, using experiences from a large commercial software producing organization, we provided evidence of the feasibility and effectiveness of field quality prediction adjustments. Our findings help software producers determine which usage characteristics to consider for prediction adjustments and whether usage characteristics actually differ; experiences with adjustments increase confidence that field quality prediction accuracy improvements are possible.

Our findings can help to improve user experiences in three additional ways. First, software producers may use the findings to target testing effort on failure-prone usage characteristics. Second, the findings may help field service organizations. Field service organizations may use the findings to suggest less failure-prone usage environments and usage scenarios to users, and they may staff call centers with experts on the failure-prone usage characteristics. Finally, the findings may help guide future research to investigate and remove underlying causes of field quality differences.

Rest of this paper is organized as follows. Section 2 discusses prior work. Section 3 provides background information on Windows data sources and processes. Section 4 discusses our analysis methodology. Section 5 discusses the usage characteristics we examined. Section 6 presents findings on effects of usage characteristics differences on field quality. Section 7 presents findings on usage characteristics differences between pre-release and post-release. Section 8 presents Windows Reliability Team's experiences adjusting field quality predictions. Section 9 discusses conclusions.

2. PRIOR WORK

This section discusses prior work examining effects of usage characteristics differences on field quality and usage characteristics differences between pre-release and post-release.

There is general consensus in the software engineering community that different usage environments and usage scenarios can lead to different quality experiences because the software is executed in different ways. Many researchers have explored characterizing and accounting for these differences. Cheung explored using probabilistic distribution of component utilization to determine reliability of software services in [2]. Musa et al. described an approach for identifying and testing intended usage environments and usage scenarios to produce accurate reliability predictions in [17]. Hassan et al. characterized usage scenarios based on repeated events for a telecom system in [7]. Weyn and Host showed that changes in usage scenarios need to be accounted for in order to accurately predict the reliability of websites in [21]. Similarly, Wang and Tang showed that user scenarios need to be considered in modeling reliability of websites in [20]. LeGuen et al. showed that software reliability predictions need to consider the amount of usage in [8]. Li et al. examined effects of several usage characteristics on the reliability for two industrial control systems in [9]. Mockus et al. examined reliability differences due to usage environment differences for a telecom system in [14]. Munson et al. predicted reliability of a software system based on module usage in [15]. Elbaum et al. discussed a method to profiling usage to aid testing and reliability predictions in [3]. Diep et al. proposed a technique to probe execution of software systems to assess reliability in [3].

This study extends prior work through a combination of breath and depth. Many previous studies examined software used in limited usage environments and usage scenarios, e.g. telecom and control systems. In this study, we examine the most widely-used software system in the world with a large software/hardware ecosystem. Furthermore, we examine field quality at the system level. This study goes beyond a single piece of software in a single environment, the method in many prior studies, and examines interactions of multiple software applications and multiple environmental factors (e.g. hardware configurations).

Few previous studies have examined usage characteristics differences between pre-release and post-release populations. Augustine and Podgurski showed usage differences between ordinary users and internal users (e.g. experienced testers) in [1] ; however, their work did not seek to explain the differences. Our work [11] noted differences between pre-release and post-release machines and corresponding failure rates, but did not examine the differences in detail. To our knowledge, this study is the first to examine and explain usage characteristic differences between pre-release and post-release machines in detail.

3. BACKGROUND

This section provides background information on data and processes used by the Windows Reliability Team to measure, predict, and improve field quality. For this paper, we used anonymous data from two Windows reliability telemetry components (i.e. programs that communicate reliability data from machines to Windows): WER and RAC. These data are representative of general consumers, not of users in large enterprises, government agencies, financial institutions, etc., that do not send telemetry data by policy. We describe the two components, the data they collect, and how the Windows Reliability Team uses the data to assess and improve field quality.

3.1 Windows Error Reporting (WER)

WER [13] is a per incidence failure reporting system that collects detailed debugging information, with the user's consent, to help developers address failures. WER automatically activates when the operating system detects a failure, e.g. application crashes, application hangs, and system crashes. WER collects debugging data, e.g. memory state, to help developers diagnose and fix issues. WER collects data with the user's consent and only when necessary (i.e. if sufficient information has already been collected for an issue, no additional data is collected). WER combines and replaces two earlier systems: Online Crash Analysis (OCA) and Dr. Watson [6]. WER was first included in Windows XP and has been in all subsequent versions of Windows. Ganapathi et al. [1] and Glerum et al. [6] discuss WER in detail.

3.2 Reliability Analysis Component (RAC)

RAC is an opt-in Windows component that continuously collects, analyzes, and reports field quality and usage data. It randomly samples a sub-set of machines from the Customer Experience Improvement Program (CEIP) [12] to send data to Windows. The CEIP population is representative of general consumers. These data include a variety of anonymous information, such as machine configuration (e.g. hard drive size and laptop/desktop), system events (e.g. application installations), user initiated events (e.g. duration of application execution, number of OS boots), and state transitions (e.g. system hibernations). An updateable set of data collection rules is shipped with Windows and defines the sub-set of data collected (i.e. not all data are collected from all machines). RAC sends data with the consent of the user. RAC is included with Windows Vista and is in all subsequent versions of Windows. Li et al. [9] and Xue et al. [22] discuss RAC in detail.

3.3 Processes and Definitions

In this study, we relied heavily on processes and definitions used by the Windows Reliability Team; we describe the key ones here.

The Windows Reliability Team uses an iterative feedback process to assess and improve field quality pre-release. The process consists of four steps: 1) product deployment, 2) telemetry data collection, 3) measurement and issue identification and prioritization, and then 4) issue resolution. After beta deployment, field quality is predicted from the telemetry data and compared with goals (goals are typically established based on previous releases). The gap between field quality and goals is quantified, along with a list of issues contributing to the gap. These issues are prioritized based on their overall impact on field quality and then assigned to developers or partner engagement teams to resolve.

The Windows Reliability Team treats *user* and *machine* as synonymous. The Windows reliability telemetry components capture data on a per-machine basis. Multiple users can use the same machine; however, they are generally not distinguished because there is no accurate way to distinguish between users (e.g. an entire family using the machine with the same account).

The Windows Reliability Team examines the average application crash, application hang, and system crash rates during the first 7 hours of active usage after installation (among many other measures of field quality). In this study, we focused on these three measures, which we collectively refer to as *failure rates*. A system crash is a catastrophic failure that prevents the system from performing its function and requires a complete system reboot [1]. System crashes are often caused by 3rd party device drivers or hardware failures. An application crash is when an application terminates with an error exit code. An application hang is defined as when an application failed to process user inputs for at least five seconds and is closed by the user.

Active usage is defined as the number of minutes with input from input devices like mouse, keyboard, stylus, touch screen, etc. Active usage is different from calendar time and system runtime. Calendar time is defined as the elapsed calendar time since install and does not consider whether the machine is running or being actively used. System runtime is defined as the accumulated runtime logged by the system and does not consider how much calendar time it took to accumulate the runtime or how much of the runtime includes interactions with the user.

The Windows Reliability Team measures failure rates during the first 7 active hours of usage after install for business reasons. The measurements assess user experience during the important initial usage period, are sufficiently stable for comparisons against goals, and can be obtained relatively quickly after the deployment of a beta version.

The Windows Reliability Team measures average failure rate over 99% of the machines. The top 1% of machines is trimmed so that the average is more resilient against outliers, which is especially important for pre-release versions. Pre-release populations often contain machines running automated tests, which can generate large numbers of failures in a short amount of time. Untrimmed, data from these machines pollute the average. The failure rate (for each type of failure) for each machine is defined as:

$$failure\ rate_{Machine} = \frac{1}{7}\sum_{i=1}^{7} failures\ in\ hour_i$$

Machines are ordered according to their failure frequency in ascending order, and the average is computed over the top 99%.

4. METHODOLOGY

This section describes our analysis methodology, i.e. the steps we took to address our research questions. This section focuses on 'why' we chose our approach, and subsequent sections (5, 6, and 7) discusses 'how' and 'what'.

- Data selection: To understand field quality variations and usage characteristics differences between pre-release and post-release, we selected failure and usage data from pre-release and post-release machines. To be indicative of the current state of practice, our data comprised of millions of machines world-wide running Windows 7, the most recent release of the Windows operating system.
- Usage characteristics selection: In order to avoid spurious correlations, we surveyed Windows reliability experts to determine a set of usage characteristics to examine.
- Identification of important usage characteristics: Due to space constraints and to prevent over-fitting of the field quality prediction adjustment model, we focused our analysis on a subset of statistically important usage characteristics, based on correlations to failure rates.
- Examination of field quality variations: We plotted and quantified usage characteristics changes and corresponding failure rates changes for the important usage characteristics. To understand possible causes for the changes, we examined the underlying failures and machines.
- Examination of usage characteristics differences between pre-release and post-release: For the important usage characteristics, we plotted and tested for differences between pre-release and post-release. We hypothesized possible causes for the differences based on Windows' experiences with beta programs.

- Windows experiences with field quality prediction adjustments: We quantified discrepancies between field quality predictions and actual post-release quality for Windows 7, likely due to usage characteristics differences. We reported Windows' experiences and results adjusting the predictions using *Usage Profile-based Reliability Measurement Calibration*.

4.1 Select Data

To be reflective of the current state of practice, we used CEIP data collected in 2009 and 2010 from machines running pre-release and post-release versions of the Windows 7 operating system. The Windows operating system is the most widely-used everyday operating system today (not specialized operating systems like ones operating aircrafts). The CEIP data were randomly sampled from machines world-wide. We used data from 1.4 million machines for the post-release version, 24 thousand machines for one pre-release version, and 34 thousand machines from a later pre-release version (during peak adoption for the two pre-release versions). All machines met the measurement requirements of the Windows Reliability Team described in Section 3.3.

4.2 Select Usage Characteristics

We chose to examine usage characteristics that were likely to be related to field quality; we informed our decision by interviewing Windows reliability experts. This approach avoided spurious correlations (i.e. examining many usage characteristics and obtaining statistically significant results by chance). We interviewed five reliability experts in the Windows Reliability Team. Collectively, these experts have over 100 years of experience assessing quality and debugging failures at Microsoft, IBM®, Digital®, and other software development organizations.

4.3 Identify Important Usage Characteristics

We identified a subset of important usage characteristics based on statistically relationships to failure rates and focused the rest of our analyses on this subset of usage characteristics. We did this for two reasons. First, due to space constraints, we could only present in-depth analysis for a limited set of usage characteristics. Second, we wanted to prevent over-fitting of the field defect prediction adjustment model (i.e. adjusting for more usage characteristics resulted in less accurate predictions), discussed further in Section 8. To make this identification, we used general linear regression (GLM) to analyze the relationship between failure rates and usage characteristics.

4.4 Examine Field Quality Variations

We examined changes in field quality associated with changes in the important usage characteristics in two ways. First, we performed descriptive statistical analysis (i.e. we plotted and quantified the changes). Second, we examined the underlying application crashes and machines to identify possible causes for the changes. We examined application crashes (not application hangs or system crashes) because it was the most frequent and prevalent kind of failure, as discussed in Section 4.3. For example, to identify possible causes for failure rates differences in various sub-populations (i.e. language/geographic locales), we analyzed failing modules that were unique to those locale. Results of this investigation improved construct validity and increased confidence in our findings; furthermore, the results identified possible areas of future investigations.

4.5 Examine Usage Characteristic Differences

We evaluated differences in the important usage characteristics between pre-release and post-release. We performed descriptive statistical analysis and conducted statistical tests for the differences. We hypothesized possible causes for the differences based on Windows' experiences with beta programs. These findings indicated possible future investigations and suggested practical limitations for beta programs. For example, machines pre-installed by the OEM should not be considered exist in pre-release versions and is a limitation of Windows beta programs.

4.6 Report Experiences with Adjustments

We reported Windows experiences using *Usage Profile-based Reliability Measurement Calibration* (UPRMC) [22] to adjust field quality predictions. We first quantified discrepancies between field quality predictions and actual post-release quality. Quantifying discrepancies provided motivation for adjusting field quality predictions and served as references for evaluating improvements. We reported Windows experiences using the four step UPRMC algorithm, which adjusts users of a target population (e.g. a pre-release version) to match the users of a reference population (e.g. a post-release version) based on a set of usage characteristics. The four steps are: identification of key usage characteristics, quantitative usage profiling, bootstrap sampling, and computation of failure rate measurements. We presented instantiations and outputs of these steps for Windows. We quantified improvements in predictions resulting from the adjustments. The results increased confidence in the feasibility of field quality prediction adjustments.

5. USAGE CHARACTERISTICS

In this section, we discuss the usage characteristics we examined. We discuss the usage characteristics identified by Windows reliability experts and the subset of important usage characteristics that we identified using GLM analysis of data from post-release machines.

5.1 Usage Characteristics

We asked Windows reliability experts to identify usage environments and usage scenarios that may affect field quality. Usage environments were hardware related characteristics, e.g. number of processors or whether the machine was a laptop. Usage scenarios were action related characteristics, e.g. number of applications executed and system runtime. Some usage characteristics could be both usage environment and usage scenario related, e.g. overclocking.

The experts identified 27 potentially important usage characteristics. We then examined the available telemetry data and obtained measures for 18 of the usage characteristics; the other 9 were not yet collected by Windows reliability telemetry components. We discuss implications of not having measures for all usage characteristics in Section 9. Table 1 describes the usage characteristics relevant for this paper, why the experts believed they would affect field quality, and how we measured them.

The other usage characteristics that we examined were: number of storage drives, number of processors, size of the hard drive, whether the machine was overclock, whether the machine was a laptop (not a desktop), whether the machine was a netbook, number of hibernations, number of sleeps, calendar days since install, size of memory (RAM), speed of the processor, machine runtime, number of applications installed, and whether the machine was a virtual machine.

Table 1: Usage characteristics measured

Usage characteristic	Why it may affect failure rates	Measure
Number of applications executed	Failure rates may differ due to quality of the applications and associated hardware (i.e. specific hardware that works with specific applications)	Count applications receiving more than 1 minute of active usage
Install Type: • **Fresh install (Previous OS)** • **Pre-installed by OEM** • **Upgrade** • **Fresh install (Media)**	Failure rates may differ due to the kind of users that utilize various types of install, as well as quality of the drivers and applications on the machine	Five indicator variable, one for each possible install type, based on the OS installation event
Sub-populations: • **World-wide average (all locales except the top 6)** • **Locale 1** • **Locale 2** • **Locale 3** • **Locale 4** • **Locale 5** • **Locale 6**	Failure rates may differ due to regional specific differences. Locales are determined based on OS language and regional settings. The six locales represent top Windows markets; they are listed in random order and are obscured for business reasons.	Seven indicator variable, one for each of the seven possible locales
64-bit (not 32-bit) Windows	Failure rates may differ due to maturity and age of hardware and applications that run 64-bit Windows	Indicator variable, whether Windows is 64 bit

5.2 Important Usage Characteristics

We used Generalized Linear Model (GLM) to analyze changes in the usage characteristics and corresponding changes in field quality in the post-release population. GLM is a flexible and widely-used analysis approach used for examining the relationship between continuous variables and categorical variables (converted to indicator variables). Since we examined failure rates, we used the Poisson variant of GLM, as prescribed in references [19].

For each usage characteristic, we treated each failure type equally by running analysis for each: application crash, application hang, and system crash. We did this for two reasons. First, these three types of failures were inherently different phenomena (e.g. application hangs required user interaction) as discussed in Section 3.3. Second, application crashes were significantly more common than the other two types of failures; therefore, combining the counts of application crashes, application hangs, and system crashes together would have skewed the results. In the post-release data, the application crash rate was ~3.1X the application hang rate and ~27.1X the system crash rate (e.g. crashes in 3rd party drivers).

Since GLM is sensitive to the scale of the variables and the usage characteristics were on significantly different scales, we normalized continuous variables to be between zero and one. We subtracted by the minimum value and divide by the value at the 99% percentile. We did not divide by the max because the max was generally an outlier. This normalization also enabled us to use a mix of indicator and continuous variables in UPRMC, discussed in Section 8.

We evaluated absolute values of the z-value statistic (larger values are better) [19], which measure the strengths of correlations. For each failure type, the z-value of a usage characteristic was divided by the largest z-value among all usage characteristics. Then, for each usage characteristic, the normalized values were summed across the three failure types. We ordered the usage characteristics using this combined value. The z-values needed to be normalized (i.e. divided by the largest z-value among all usage characteristics) because of differences in the rarity of the failure types. It was easier to get higher z-values for application crashes, followed application hangs, and then system crashes (i.e. higher z-values for system crashes were harder to get). The usage characteristics we selected had the highest overall relationship across all three failure types.

The top five usage characteristics and their normalized combined values were: **Number of applications executed:** 3.00 (the highest possible across three failure type), **Install Type: Pre-installed by the OEM:** 1.01, **Sub-population: Locale: 2:** .84, **64-bit (not 32-bit) Windows:** .78, **Sub-population: Locale 6:** .68. Rest of the usage characteristics had lower combined values, though not by much (e.g. the sixth most important, number of storage drives, had a combined value of .65). We focused our analysis on the top five usage characteristics because of space considerations and because including additional usage characteristics in field quality predictions adjustments did not improve accuracy (including each of these top five did improve predictions), discussed further in Section 8. Software producers seeking to adjust field quality predictions should consider these five usage characteristics.

6. DIFFERENCES IN FIELD QUALITY

In this section, we present results of the descriptive statistical analysis of field quality differences corresponding to changes in the important usage characteristics (identified in Section 5), using data from post-release machines. We also identify possible causes for the differences by examining the underlying failures and machines.

For numerical usage characteristics, e.g. the number of applications executed in Table 1, we ordered the machines and then plotted the percentile of machines against the average failure rates for the percentile. For categorical usage characteristics, e.g. install type in Table 1, we plotted the average failure rate for machines in the category. For both types of data, we obscured the data for business reasons. We obscured the data by normalizing the failure rates; for numerical usage characteristics, we divided the failure rate of a percentile by the failure rate at the 99 percentile, and for categorical usage characteristics, we divided the failure rates by the failure rates of one of the categories.

6.1 Number of Applications Executed

Failure rates increased with increases in the number of applications executed, as shown in Figure 1. In Figure 1, the horizontal axis is the percentile of machines, ordered by the number of applications executed. The differences were 2.0X for system crashes, 3.1X for application crashes, and 3.5X for application hangs, between users at the 10% percentile (far left) and user at the 99% percentile (far right).

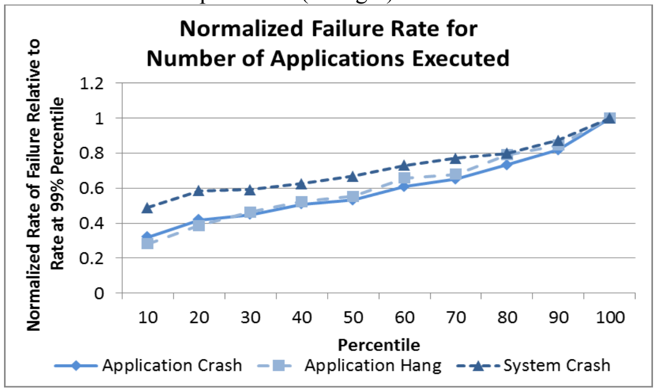

Figure 1: Changes in failure rate relative to apps executed

A likely cause was the diversity of applications executed. We examined the number of distinct crashing application modules for machines in the lower 25 percentile and the upper 25 percentile. Even though the numbers of machines were the same, the upper 25 percentile machines had failures in 1.8X more modules (~75K against ~42K). This finding aligns with existing reliability theory [17]. Execution of an application has some chance of failure, and more applications executed results in, on average, more failures.

6.2 Install Type: Pre-Installed by OEM

Machines pre-installed by OEM had the lowest failure rates, as shown in Figure 2. Relative to machines fresh installed from previous OS (i.e. where the fresh install is initiated from a running OS and not from media), the differences were .52X for system crash, .76X for application crash, .90X for application hang.

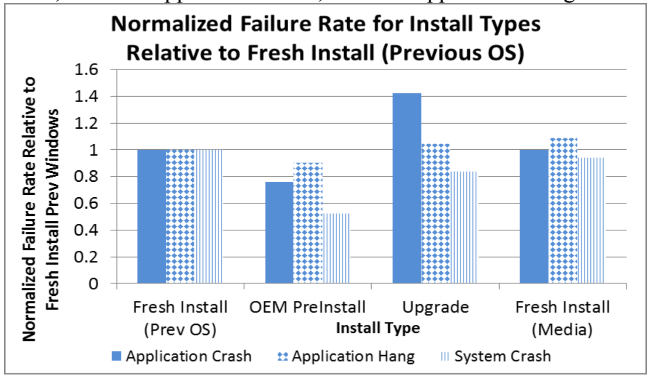

Figure 2: Failure rate for install types

The likely cause was the behavior of users that performed a fresh install. Users of machines that were fresh installed (from previous OS) tended to use their machines more extensively than users of machines that were pre-installed by the OEM. We examined applications executed and applications installed as proxies for how extensively the users used their machines. For machines that were fresh installed (from previous OS), the median applications executed was at the 39.2 percentile and the median applications installed was at the 18.6 percentile. For machines that were pre-installed by the OEM, the median applications executed was at the 35.2 percentile and the median applications installed was at the 8.0 percentile. These differences in how extensively the users used their machines likely resulted in the differences in failure rates, as discussed in Section 6.1.

6.3 Sub-Population: Locale 2 and Locale 6

Sub-populations are specific language/geographic locales. We examined locales corresponding to the top six Windows markets, as discussed in Table 1. The locales are obscured for business reasons.

Locale 2 had better failure rates than the world-wide average for all failure types, while Locale 6 had worse application crash and application hang rates but better system crash rate, as shown in Figure 3. The differences for Locale 2 were .69X for application crash, .61X for application hang, and .57X for system crash. For Locale 6, failure rates were 1.13X for application crash, 1.39X for application hang, and .77X for system crash.

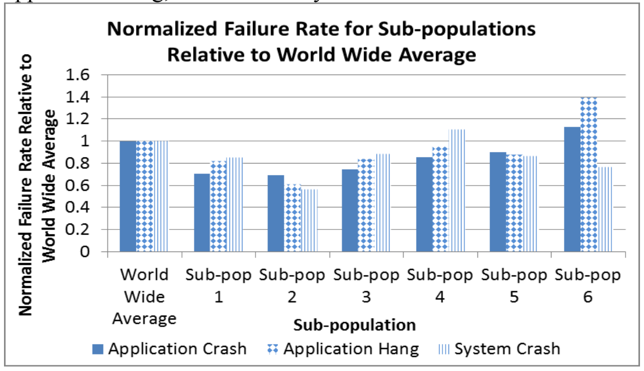

Figure 3: Failure rate for top locales

A likely cause was the localized nature of the software ecosystems. Both of these locales used double-byte languages (e.g. Chinese). We examined crashing application modules not present in rest of the world (i.e. localized). For Locale 6, 35.2% of modules (~8 thousand) were localized. For Locale 2, 25.0% of modules (~3 thousand) were localized. Quality of the locale specific software ecosystems likely dictated the failures rates.

6.4 64-Bit (not 32-Bit) Windows

Machines running 64-bit Windows had lower failure rates than machines running 32-bit Windows, shown in in Figure 4. The differences were .80X for application crash, .90 X for application hang, and .72 X for system crash.

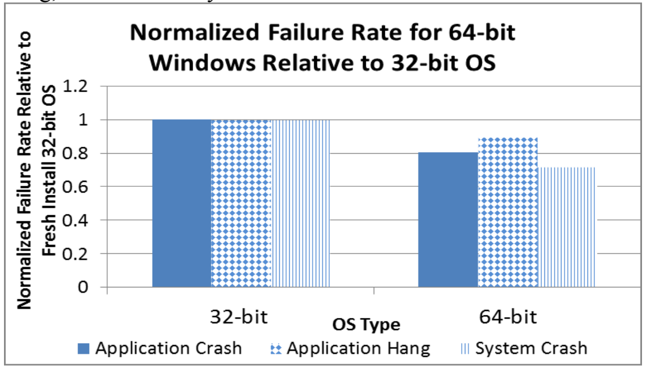

Figure 4: Failure rate for 64-bit and 32-bit Windows

The likely cause was age of the machines. Newer machines tend to have fewer hardware related issues, reducing application crashes and system crashes. Also, newer machines tend to have better performance, reducing application hangs. Although, we did not have a direct measure for the age of the machine, we used memory size and processor speed as proxies. For machines running 64-bit Windows, the median memory size was at the 46.0 percentile and the median processor speed was 68.3 percentile. For machines running 32-bit Windows, the median memory size was at the 22.6 percentile and the median processor speed was at the 65.3 percentile.

7. PRE-RELEASE AND POST-RELEASE USAGE CHARACTERISTICS

In this section, we present descriptive statistical analysis of differences in the important usage characteristics identified in Section 5 between pre-release and post-release machines. Also, we present results of statistical tests for differences (and similarities) and identify possible causes for differences based on Windows' experiences with beta programs.

We used two statistical tests for usage characteristics differences: the two-sample Kolmogorov-Smirnov (K-S) test for continuous usage characteristics and the proportion test (Pearson's χ^2 test) for categorical usage characteristics. The K-S test is a non-parametric test of equality of empirical distributions [19]. It rejects the null hypothesis (two distributions are equal) based on differences in the cumulative distribution functions (i.e. how many observations are below a given value). We used the K-S test because the distributions of machines with usage characteristics generally do not follow known distributions. The proportion test is a test for equality of proportion of successes and failures [19]. With sufficient number of observations, the test rejects the null hypothesis that the proportion of successes (e.g. users running 64-bit Windows in the post-release population) is the same as another sample (e.g. users in pre-release version 1) based on the χ^2 distribution. Since we had millions of observations, this test was appropriate for examining the categorical usage characteristics. The statistical tests allowed us to statistically verify whether usage characteristics differed between pre-release and post-release.

7.1 Number of Applications Executed

Pre-release machines executed more applications during the first 7 hours of active usage compared to post-release machines. In the plot of pre-release and post-release distributions in Figure 5, the curves for pre-release versions are to the right of (higher percentiles) the post-release version (i.e. more machines with higher numbers of applications executed). Results were statistically significant at the 95% confidence level (K-S test statistics of .0542 and .0742)

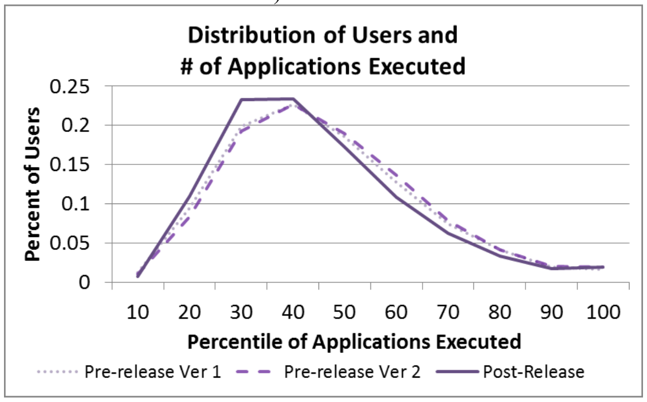

Figure 5: Distribution differences for applications executed

A likely cause was the proportion of people who are skilled with computers in pre-release populations. Pre-release populations generally have higher proportions of people who are skilled with computers and confident with trying applications. The average users are generally less inclined to experiment with software, which may not be fully functional and may contain bugs.

7.2 Install Type: Pre-Installed by OEM

There were fewer machines pre-installed by OEM pre-release, as shown in figure 6. The differences were statistically significant at the 95% confidence level based on the proportion test (χ^2 statistics of 255.13 and 102.8241).

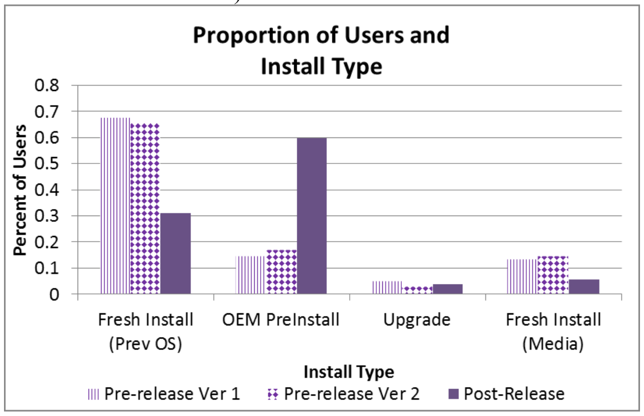

Figure 6: Proportion differences for install type

In theory, machines pre-installed by the OEM did not exist pre-release. OEMs generally prepared machines with up-to-date drivers and applications *after* the OS was ready. Pre-release machines installed by the OEM were likely testing the machine setup process. These machines likely did not have up-to-date drivers and applications (i.e. they were not similar to post-release machines). For field defect predictions, the Windows Reliability Team excluded these machines, discussed in Section 8.

7.3 Sub-Population: Locale 2 and Locale 6

Locale 2 had fewer machines pre-release; Locale 6 had more machines pre-release. Plot of the proportions are Figure 7. The differences between pre-release and post-release were statistically significant at the 95% confidence level based on the proportion test for both Locale 2 and Locale 6 (χ^2 statistics of 714.3097 and 779.2834 for Locale 2; 251.4201 and 43.968 for Locale 6).

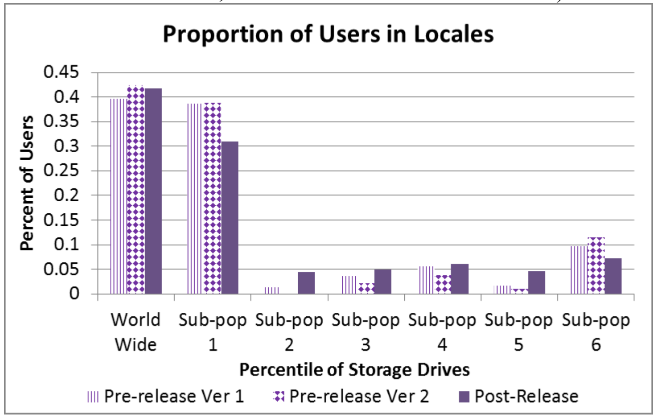

Figure 7: Proportion differences for locales

The availability of language packs was likely *not* a cause for the differences. Pre-release version 1 was available in languages of both Locale 2 and 6. Pre-release version 2 was available in the language of Locale 2 (but not for sub-population 6). Yet, the proportion of machines in Locale 2 actually decreased from pre-release version 1 to pre-release version 2, while the proportion of machines in Locale 6 increased.

A likely cause for the differences was cost. Both pre-versions were effectively free for users. Users in Locale 6 might have been more inclined to try free, albeit pre-release, software.

7.4 64-Bit or 32-Bit Windows

Fewer machines ran 64-bit Windows pre-release, as shown in Figure 8. The differences were significant at the 95% confidence level based on the proportion test (χ^2 statistics of 10753.177 and 11463.96).

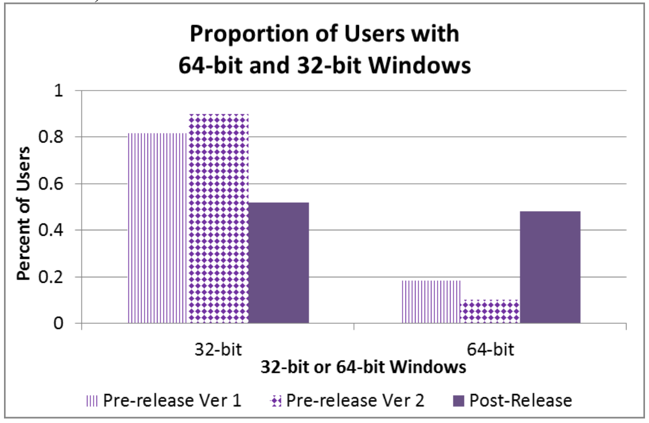

Figure 8: Proportion differences between 64/32-bit Windows

A likely cause was post-release users running new hardware. Post-release, users generally obtained Windows with the purchase of a new computer. New machines tended to have newer hardware that runs 64-bit Windows. Nonetheless, there were new computers running 32-bit Windows (e.g. netbooks) and older machines being upgraded; thus, not all post-release machines ran 64-bit Windows.

8. PREDICTION ADJUSTMENTS

In this section, we report experiences of the Windows Reliability Team using UPRMC to adjust field quality predictions, accounting for usage characteristics differences between pre-release and post-release machines. First, we quantify the discrepancy between field quality predictions and actual field quality. Then we describe results of instantiating the four steps of UPRMC for Windows. Finally, we report results of adjustments. The Windows Reliability Team considered adjustments because misleading field quality predictions could lead to sub-optimal release actions, discussed in Section 3.3.

8.1 Prediction Discrepancies

Current practice takes failure rates measured on pre-release versions, adjusted for fixes, as the predicted field quality; we compared failure rates of the pre-lease versions and the post-release version. We estimated fixes using failure rate changes on Windows internal self-host machines. Since usage characteristics remained largely the same for internal self-host machines (i.e. machines and the usage scenarios remained relative unchanged), comprised of machines from ~2000 Windows engineers, the observed differences in failure rates is an approximation of fixes.

Shown in Figure 9, the absolute relative errors in predictions were ~36% for pre-release version 1 and ~89% for pre-release version 2, averaged across the three types of failure. These errors were statistically significant based on 95% bootstrap confidence intervals (i.e. sampling subsets of the data to determine the variation in the data), indicated by the error bars in Figure 9.

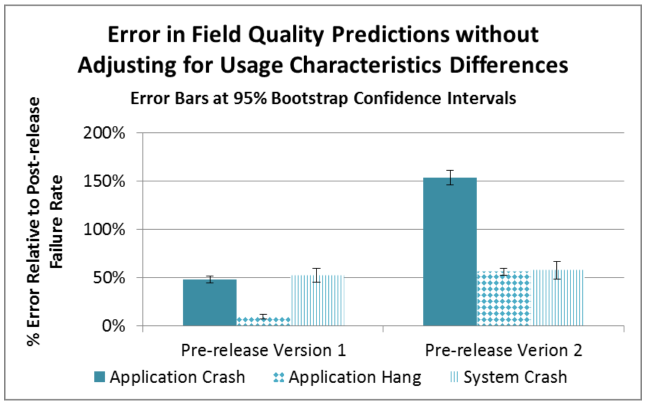

Figure 9: Error in field quality predictions

The likely causes for the discrepancies were differences in usage characteristics as discussed in the previous sections. The important usage characteristics we identified in Section 5 were shown to be related to failure rates in Section 6, and we identified differences in proportions of machines with these usage characteristics between pre-release and post-release in Section 7.

8.2 Instantiation of UPRMC

To adjust for the differences, the Windows Reliability Team used UPRMC [22]. Instantiation, explanation, and results of the four steps of UPRMC are:

1. Identification of key usage characteristics

This step identifies the usage characteristics to profile. The Windows Reliability Team used the important usage characteristics identified in Section 5.

I. Number of applications executed
II. Whether the machines was pre-installed by OEM
III. Whether the machine was from Locale 2
IV. Whether the machine was running 64-bit Windows
V. Whether the machine was from Locale 6

Using a statistical cut-off to select important usage characteristics would not have been appropriate because the data did not follow known distributions. Instead, an empirical approach was taken: iteratively including more variables and then evaluating predictions. The Windows Reliability Team observed improved prediction accuracy with the five usage characteristics, but not with additional usage characteristics.

Since it was not possible to have machines pre-installed by OEM before release, discussed in section 7.1.2, machines pre-installed by OEM were sampled out both pre-release and post-release.

2. Quantitative usage profiling

This step profiles usage using k-Means clustering [19] over the usage characteristics. For categorical data, e.g. Install Type, all categories were transformed into multiple indicator variables (i.e. 1 for being of the category and 0 for not being in the category) as discussed in Section 5. This approach is used by existing statistical packages, e.g. WEKA [9]. The k-Means clustering algorithm is then used to partition the machine sample into ten clusters, each having similar usage characteristics. K-Means minimizes the following function:

$$J = \sum_{j=1}^{10} \sum_{i=1}^{n} \left| X_i^{(j)} - C_j \right|^2$$

Where $\left|X_i^{(j)} - C_j\right|^2$ is the squared Euclidian distance between a machine, $X_i^{(j)}$ and the cluster center C_j.

The Windows Reliability Team produced ten clusters using the k-Means algorithm. The Windows Reliability Team tried using more than ten clusters but encountered feasibility issues: there were no machines in some niche clusters. Fewer clusters did not improve accuracy of predictions. Using ten clusters appeared to be a good rule-of-thumb.

The cluster centers C_j were defined as <Number of applications executed $_j$, Locale 2 $_j$, 64-bit (not 32-bit) Windows $_j$ Locale 6 $_j$>. The cluster centers are presented in Table 3. For example, ~35.7% of the post-release Windows users were centered on the 28 percentile for number of applications executed, were not from Locale 2 or 6, and ran 32-bit Windows.

Table 3: Cluster centers for post-release users

Number of Applications Executed	Locale 2	64-bit (not 32-bit) Windows	Locale 6	Percent
0.28	0.00	0.00	0.00	**35.7%**
0.51	0.00	0.00	0.00	**22.1%**
0.34	0.00	1.00	0.00	**18.7%**
0.69	0.00	1.00	0.00	**7.7%**
0.40	0.00	0.00	1.00	**5.5%**
0.87	0.00	0.00	0.00	**4.8%**
0.79	0.00	0.00	1.00	**2.4%**
0.43	1.00	0.00	0.00	**1.3%**
0.51	0.00	1.00	1.00	**1.1%**
0.50	1.00	1.00	0.00	**0.6%**

3. Bootstrap sampling

This step constructs a calibrated sample by re-sampling machines (i.e. including multiple times) to ensure that the calibrated sample exhibits the same usage profile as the reference profile (i.e. percent of machines in each clusters matches the percentage in the reference sample). The reference profile is the output from Step 2, shown in Table 3. For the calibrated sample, every machine is counted at least once, and the maximum cardinality difference between any two machines within each cluster is one (i.e. no machine is over-weighted within a cluster). The Windows Reliability Team constructed new machine samples for both pre-release versions using this methodology.

4. Computation of failure rate measurements

This step computes the calibrated failure rates, which are the failure rates calculated on the calibrated sample. The Windows Reliability Team used the same calculation method described in Section 3.3. The failure rates for the individual machines remained unchanged; however, the adjusted sample contained additional re-sampled machines having the desired usage characteristics.

By adjusting for the five usage characteristics (sampling out OEM Pre-install machines and using UPRMC to calibrate for the other four), accuracy of predictions improved by ~59%, as shown in Figures 10 and 11. Accuracy of predictions improved by ~78% overall for pre-release version 1 and ~51% overall for pre-release version 2. Both improvements were statistically significant based on 95% bootstrap confidence intervals.

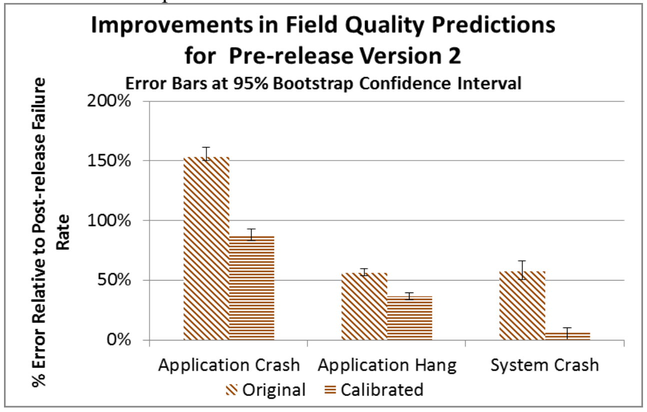

Figure 10: Improvements for Pre-release Version 1

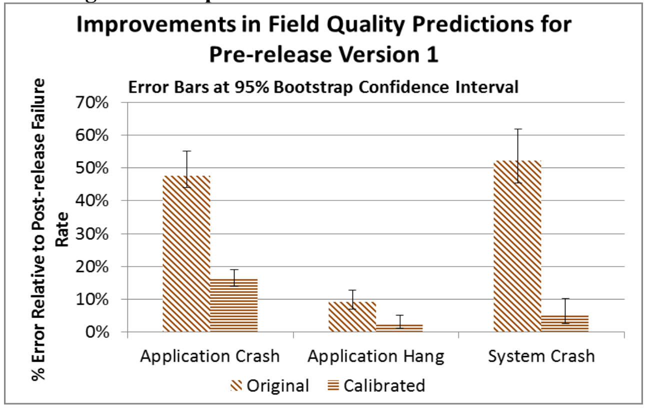

Figure 11: Improvements for Pre-release Version 2

9. CONCLUSIONS

Many software producers today utilize data from beta programs to predict field quality and fix failures; however, this paper shows that usage environments and usage scenarios can differ between pre-release and post-release machines, leading to misleading field defect predictions. For two pre-release versions of the Windows 7 operating system, the absolute relative errors of predictions (adjusted for fixes) were ~36% and ~89%.

We identified 5 important usage characteristics based on their relationship with field quality for Windows: number of applications executed, whether the machine was pre-installed by OEM, whether the machine was from one of two double-byte language locales, and whether the machine is running 64-bit Windows. For Windows 7, the proportion of machines with these important usage characteristics differed significantly between the two pre-release versions and the post-release version.

We reported the Windows Reliability Team's experiences and results using UPRMC and information on the important usage characteristics to adjust field defect predictions. The Windows Reliability Team obtained improvements in accuracy of field defect predictions of ~78% and ~51% for the two pre-release versions of Windows by adjusting for differences in the five usage characteristics.

Based on the findings, we believe that software producers running beta programs should consider adjusting for usage characteristics differences between pre-release and post-release. This is especially true for software that interacts with many other software applications and is released world-wide. Software producers can use two approaches to account for the differences: 1) selectively deploying pre-release versions to match post-release usage or 2) statistically adjusting for usage characteristics differences. In practice, Windows has found it impractical to control deployment of pre-release versions. Many partners use pre-release versions to test and verify their own software products. These tests often do not represent real-world usage but are critical to improving the overall quality experience for users. Furthermore, it is also impractical for Windows to control the distribution of pre-release versions (e.g. through bit-torrent). Consequently, software producers may want to focus on statistically approaches.

Our findings suggest that software producers should have localized quality improvement efforts to obtain high field quality world-wide. Our analysis shows that individual locales have highly localized software ecosystems. For two locales with two different double-byte languages, ~25% and ~35% of the crashing applications modules were seen only in that locale. These locale specific failures may be missed if failures are aggregated and prioritized world-wide.

Our findings may be specific to Windows and may change over time as hardware and software evolve. Nonetheless, since little quantitative information is available today, this paper can be useful to other software producers and can be a mile marker for gauging future changes. Furthermore, since many software producers build on top of the Windows platform, the information is relevant to their quality improvement efforts.

There may be other usage characteristics that significantly affect field quality not included in our analyses. We were not able to obtain measures for all usage characteristics identified by the Windows reliability experts, and there may be other important usage characteristics beyond their knowledge. For example, after conducting the analyses, we came to believe that age of the machine might be a hidden variable that underlie several other usage characteristics, as discussed in Section 6.4. Also, Section 6.1 suggests that highly skilled users are likely different from average users, improved detection of these users may also improve predictions. We hope to investigate age of the machine and additional usage characteristics in future studies.

The findings and experiences reported in this paper can help software producers with current practices and guide future research to improve experiencers with software systems for billions of users world-wide.

10. ACKNOWLEDGEMET

Thanks to Gabriel Aul and Michael Fortin for their support of this work, Jason Garms, Vince Orgovan, and Tony Voellm for their leadership and feedback, Gretchen Loihle, David Grant for their expert input, and the Windows 7 Reliability Team for providing foundational work: Song Xue, Mingtian Ni, Raymond Fingas, Yuriy Vasyleha, and Arvind Visvanathan.

11. REFERENCES

[1] V. Augustine, A. Podgurski, Corroborating User Assessments of Software Behavior to Facilitate Operational Testing. *ISSRE,* pp. 61-70, 2007.

[2] R. C. Cheung, A User-Oriented Software Reliability Model. *IEEE Trans. Software Engineering*, Vol. SE-6, No.2, pp.118-125, March 1980

[3] M. Diep, M. Cohen, S. Elbaum, Probe Distribution Techniques to Profile Events in Deployed Software. *ISSRE*, pp. 331-342, 2006.

[4] S. Elbaum, M. Diep, Profiling Deployed Software: Assessing Strategies and Testing Opportunities, *IEEE Trans. Software Engineering*, Vol. 31, No. 8, Aug. 2005.

[5] A. Ganapathi, V. Ganapathi, D. Patterson, Windows XP Kernel Crash Analysis. *Large Installation System Administration Conference*, pp. 149-159, 2006.

[6] K. Glerum, K. Kinshumann, S. Greenberg, G. Aul, V. Orgovan, G. Nichols, D. Grant, G. Loihle, G. Hunt, Debugging in the (Very) Large: Ten Years of Implementation and Experience. *SOSP,* pp. 103-116, *2009.*

[7] A. E. Hassan, D. J. Martin, P. Flora, P. Mansfield, D. Dietz, An Industrial Case Study of Customizing Operational Profiles Using Log Compression. *ICSE*, pp. 713-723, *2008.*

[8] H. Le Guen, R. Marie, T. Thelin, Reliability Estimation for Statistical Usage Testing using Markov Chains. *ISSRE,* pp. 54-65, *2004.*

[9] K-Means Clustering in WEKA http://maya.cs.depaul.edu/classes/ect584/WEKA/k-means.html retrieved on 2011-02-01

[10] P. L. Li, J. Herbsleb, M. Shaw, B. Robinson, Experiences and Results from Initiating Field Defect Prediction and Product Test Prioritization Efforts at ABB Inc. *ICSE*, pp. 413-422, 2006.

[11] P.L. Li, M. Ni, S. Xue, JP. Mullally, M. Garzia, M. Khambatti, Reliability Assessment of Mass-Market Software: Insights from Windows Vista. *ISSRE* pp. 265-270, *2008.*

[12] Microsoft Corp: http://www.microsoft.com/products/ceip/EN-US/default.mspx, retrieved on 2001-02-01

[13] Microsoft Corp.: Introducing Windows Error Reporting. http://msdn2.microsoft.com/en-us/isv/bb190483.asp.

[14] A. Mockus, P. Zhang, P.L. Li, Predictors of Customer Perceived Software Quality. *ICSE*, pp. 225-233, *2005.*

[15] J. C. Munson, S. Elbaum, Software Reliability as a Function of User Execution Patterns. *International Conference on System Sciences*, 1999

[16] B. Murphy, Automating software failure reporting, *ACM Queue*. Vol 2, No 8,pp. 42-48, Nov. 2004.

[17] J.D. Musa, A. Iannino, K. Okumoto, *Software reliability: measurement, prediction, application,* McGraw-Hill, 1987.

[18] National Institute of Standards and Technology. The economic impacts of inadequate infrastructure for software testing. *Planning Report 02-3*, Jun. 2002

[19] R Project for Statistical Computing, http://www.r-project.org/, retrieved on 2011-02-01

[20] W. Wang, M. Tang, User-Oriented Reliability Modelling for a Web System. *ISSRE,* pp. 293-306, 2003.

[21] K. Weyns, M. Host, Sensitivity of Website Reliability to Usage Profile Change. *ISSRE,* pp.3-8, 2007.

[22] S. Xue, P.L. Li, J.P. Mullally, M. Ni, G. Nichols, S. Heddaya, B. Murphy, Predicting the Reliability of Mass-Market Software in the Marketplace Based on Beta Usage: a Study of Windows Vista and Windows 7. *MSR Tech Report*, MSR-TR-2011-2, 2011