Predicting the Reliability of Mass-Market Software in the Marketplace Based on Beta Usage: a Study of Windows Vista® and Windows 7®

Song Xue[+], Paul Luo Li[+^], Joseph P. Mullally[+], Mingtian Ni[+], Greg Nichols[+], Solom Heddaya[+], and Brendan Murphy*

Microsoft Windows Reliability Team[+],
One Microsoft Way
Redmond, WA USA
[songxue,pal,joemull,mini,grenich,solomhe]
@Microsoft.com

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

The Information School[^]
University of Washington
Seattle, WA U

ABSTRACT
The traditional approach for predicting post-release software reliability is to emulate customer usage environments and scenarios during in-house testing. However, this approach has limitations for mass-market software systems (MMSS), such as the Windows operating system, because it is impractical to emulate the usage environments and scenarios of millions of users. This paper presents and validates Usage Profile-based Reliability Measurement Calibration (UPRMC), a novel approach for producing accurate post-release reliability predictions using data from beta releases. This work leverages usage and reliability data from multiple releases of a large commercial operating system spanning more than 3 years and more than 3 million users. Results show that UPRMC produces accurate and credible predictions; it has been adopted by the Windows organization.

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

D.2.4 [**Software Engineering**]: Management – *Software Quality Assurance*

D.2.4 [**Software Engineering**]: Metrics – *Product Metrics*

General Terms
Management, Measurement, Economics, Reliability, Experimentation, Verification

Keywords
Mass market software, Windows, Customer experience improvement program, Windows Error Reporting, Reliability Analysis Component (RAC), Usage profile, Beta

1. INTRODUCTION
The most purchased and widely-used software systems today are mass-market software systems (MMSS), which differ from software systems of the past. Historical software systems were typically built for specific usage environments and scenarios, e.g. NASA space shuttle software [24] and AT&T telephone switching software [19]. Later software systems, like VMS [17] and Tandem [6], were designed for a limited set of usage environments and scenarios. Today, MMSS are designed to be used by a variety of users in various usage scenarios. Operating systems, such as Microsoft Windows® and Apple OSX®, are examples of MMSS. Nearly half a million unique devices and over 6 million applications have been run on Windows Vista®.

Ensuring that MMSS meet reliability expectations of customers has tremendous benefits. For software producers, reliability improvements can result in billions of dollars in revenue and in saving from repairs [21]. Reliability improvements also mean increased utility for millions of users.

Predicting software reliability in the marketplace (i.e. post-release) is critical to releasing quality software. Predictions enable software producers to identify high impact reliability issues and then take corrective actions before release. Software producers can then make economical and informed release decisions. Numerous previous studies have discussed benefits of reliability predictions, e.g. Musa et al. [19], Lyu [13], Murphy [18], Wood [26], and Paulk et al. [22].

The traditional approach of using data from in-house testing to predict post-release reliability is inadequate for MMSS due to usage environments and scenarios differences between in-house testing and post-release. It is often impractical for producers of MMSS to emulate all possible post-release usage during in-house testing; it is also difficult to determine which usage scenarios to test with each usage environment. Furthermore, usage environments and scenarios are continuously changing. Numerous previous studies have discussed these problems, e.g. Elbaum and Diep [1] and Li et al. [11].

Most MMSS producers today deploy beta versions (i.e. pre-release software deployed to real-world users) to assess reliability and other quality factors during actual customer usage. Microsoft and Adobe® are examples of MMSS producers that utilize beta programs. Reliability measured on beta users, after accounting for fixes, is commonly accepted as the predicted reliability in the marketplace.

This paper presents quantitative data showing that usage of beta versions can differ significantly from usage post-release. The corresponding reliability measurements can exhibit unexpected reliability regressions in successive beta versions despite reliability improvements. This anomaly hinders MMSS producers from predicting reliability. The usage and reliability data are collected via Windows telemetry systems for pre-release and post-release versions of Windows Vista, Windows Vista SP1®, and Windows 7®. The data spans more than 3 years and come from more than 3 million users. This paper addresses the problem:

How can producers of mass-market software use data from beta releases to accurately predict reliability in the marketplace?

This paper presents and validates *Usage Profile-based Reliability Measurement Calibration (UPRMC), an approach for predicting* *reliability in the marketplace for MMSS using data from beta releases*. UPRMC quantifies and statistically adjusts for usage differences between pre-release and post-release populations. UPRMC currently profiles usage along two dimensions: application coverage and usage duration. Experience shows that UPRMC removes measurement anomalies, reduces absolute relative error by up to 38%, and has been adopted by the Windows organization.

This paper is structured as follows. Section 2 discusses prior work in reliability prediction and usage profiling. Section 3 provides background information on processes and data used by Microsoft to assess reliability. Section 4 describes issues with pre-release usage and corresponding reliability measurements, and then provides insights into the solution. Section 5 describes UPRMC, the approach developed by the Windows Reliability Team to predict post-release reliability using data from beta releases. Section 6 provides validation using Windows data from Vista and Vista SP1. Section 7 discusses threats to validity. Section 8 contains the conclusion.

2. RELATED WORK

Reliability measurement and prediction are key aspects of software reliability engineering. It has been discussed in several books including Lyu et al. in [13], Fenton and Pfleeger in [2], and Musa et al. in [19], and it has been examined in industry, e.g. Nagappan et al. in [20] and Mockus et al. in [16]. It is software best practice as described in the Capability Maturity Model for Software [22]. There is general consensus in the software engineering community that pre-release predictions of post-release reliability help software producers meet reliability expectations of customers.

There are generally two classes of approaches for predicting reliability before release: software reliability growth modeling (SRGM) and statistical modeling. Both approaches use data from testing and development. SRGMs are time-based approaches that model failures occurrences as a mathematical function of time. Models include the Logarithmic model (i.e. the Musa Model [19]) and the Gamma model (i.e. the S-Shaped model [13]). Statistical models attempt to quantify the relationship between pre-release metrics and post-release failures. Approaches include logistic regression, classification trees, and neural networks. Li provides detailed discussions of these two classes of approaches in [10].

Both approaches make assumptions about similarity of usage environments and scenarios between in-house testing and post-release. SRGM assumes that usage is the same between in-house testing and post-release; therefore, pre-release reliability growth can be extrapolated to post-release. Statistical modeling assumes that the statistical relationship between in-house testing and post-release remains similar between releases; therefore, the relationship can be used to make predictions.

MMSS violates assumptions made by both classes of approaches. It is impractical to emulate all post-release usage environments and scenarios during in-house testing, and the relationship between in-house testing and post-release usage can differ between releases. Several previous studies have noted these issues, e.g. Murphy and Gent in [17], Gray in [6], Jeske and Akber-Quereshi in [9], and Li et al. in [11]. Consequently, many software producers today use reliability measured on beta versions as the predicted reliability in the market place, accounting for reliability improvements.

However, in section 4, we show that data from beta releases can produce misleading reliability predictions due to usage differences. We use data from Windows Vista, Windows Vista SP1, and Windows 7 to illustrate the problem.

This paper presents Usage Profile-based Reliability Measurement and Calibration (UPRMC), which automatically quantifies usage differences between versions and then statistically adjusts reliability measurements to account for the differences.

Previous work on usage profiling is mostly aimed at testing. Musa [19] is one of the first to discuss profiling usage in a systematic manner; however his approach is time consuming and depends on the availability of experts. Furthermore, there is no easy way to determine whether actual usage conforms to constructed profiles. UPRMC automatically generates usage profiles and is similar to profiling of web sites discussed by Goseva-Popstojanova et al. [5] and Weyns and Martin [25]. However, web site usage is less complex than operating system usage. Furthermore, profiling usage is easier for web applications because web usage entails sequential requests to the server containing full context of the usage. There has been work on automatically profiling application executions, notably the work by Elbaum and Diep [1]; however, such techniques have yet to be used on large scale production software systems.

The work in this paper is novel in three important ways. First, it is the first to quantitatively compare usage and reliability of pre-release and post-release software. Second, it provides a novel approach for producing accurate reliability in the marketplace predictions for MMSS. Finally, the data used to describe the problem, formulate the solution, and validate the solution come from millions of real-world users of a widely-used operating system.

3. BACKGROUND

Windows Reliability Team uses an iterative feedback process to drive reliability improvements. The process consists of four steps: (1) product deployment, (2) telemetry data collection, (3) reliability measurement and issue identification and prioritization, and then (4) issue resolution. After deployment, reliability metrics are computed from telemetry data and compared with reliability goals; the goals are typically established based on previous releases. The gap between current reliability status and reliability goals is quantified along with a list of reliability issues contributing to the gap. These issues are prioritized based on their overall reliability impact and then assigned to developers or third party engagement teams for resolution. Each iteration, the telemetry instrumentation can be enhanced to collect additional data to facilitate investigation of existing or newly discovered issues. An iteration can take months or years to complete depending on factors like phase of development (i.e. pre-release or post-release), code ownership (i.e. Windows or 3rd party code), the impact of the change (i.e. isolated or interconnected components), and the deployment mechanism.

Windows has several reliability related telemetry systems, each with a different focus. The most widely-known system is Windows Error Reporting (WER) [15], which combines and replaces two earlier systems: Online Crash Analysis (OCA) and Dr. Watson [4]. WER automatically activates when an application crashes or hangs or when the operating system experiences an unrecoverable failure. WER collects debugging data to help developers diagnose and resolves issues. WER collects data with the user's consent and only when necessary (i.e. if sufficient information has already been collected from other users to diagnose an issue, no additional data is collected). WER was first included in Windows XP and is in all subsequent versions of Windows. Ganapathi et al. [3] and Glerum et al. [4] discuss WER in detail. However, since WER is designed to help developers diagnose and fix issues in an efficient manner, it has several key design limitations. WER only activates upon failures and does not capture information on installs, hardware re-configurations, or successful execution. Consequently, WER data do not produce normalized reliability metrics that can be compared across releases and time, such as the failure frequency metric discussed in Section 3.4. To compute these reliability metrics and to enable other reliability evaluations, Windows has introduced a new telemetry system in Windows Vista: the Windows Reliability Analysis Component (RAC). Most of the data in this paper come from RAC. The data collected by these telemetry systems are likely representative of general users (i.e. not enterprise or government users that have telemetry systems disabled by policy). The rest of this section describes RAC and RAC data.

3.1 Reliability Analysis Component (RAC)

RAC is an opt-in Windows component that collects, analyzes, and reports reliability telemetry data. It surfaces reliability information to the user via the Reliability Monitor. It randomly samples machines in the Customer Experience Improvement Program (CEIP) [14] to send telemetry data to Microsoft. The data are anonymous, RAC attempts to avoid collecting personally identifiable information (PII), and data are sent with consent of the user.

RAC was developed to assess reliability and usage pre-release and post- release. Pre-release, RAC provides data on Microsoft internal self-hosting and external deployments. The Windows Reliability Team uses the data to monitor build-to-build progress and to predict reliability of the final product in the marketplace. Post-release, the Windows Reliability Team assesses the reliability status of the Windows ecosystem, prioritize reliability improvements, and establishes baselines for future versions of Windows.

3.2 Data

RAC provides four types of data:

- Machine profile: machine hardware information (e.g. devices, memory, processor, hard drive, laptop/desktop, manufacturer, and model) and Windows OS information (e.g. version, locale, time zone); this data enables analysis along machine dimensions.
- Discrete system events: application failure, WER events, application and device installations, etc.; this data enables in-depth investigation into specific events of interest, for example, relationships between application installations and reliability issues.
- Metrics: there are 4 types of basic metrics: OS metrics, service metrics, application metrics, and event metrics. Metric data consists of counts of occurrences (e.g. system boot, application launch) and time durations in certain states (e.g. OS uptime, application uptime). Metrics are computed on pre-defined intervals (e.g. every 3 days). These metrics are the foundation of reliability measurement and reporting.
- State transitions: launches, terminations, and sleep transitions for the OS, services, and applications. Each state change is recorded with an associated timestamp and the appropriate parameters.

Not all data is collected from every machine. An updateable set of data collection rules is shipped with Windows and defines the sub-set of data collected.

RAC receives anonymous machine centric data globally. Every data point has an associated anonymous machine identifier. The machine identifier is randomly generated. RAC has data from all major consumer markets and segments. However, since RAC samples only a subset of machines where the users have explicitly opted into CEIP, some small groups, enterprises, and government agencies are not fully represented. For example, RAC may not have data on machines assembled by a custom machine builder.

RAC randomly selects a few million machines from the CEIP population. The number of machines can be adjusted to accommodate analysis needs and deployment sizes. The sample is optimized to provide adequate number of machines for analysis of major markets and market segments.

3.3 Failure Definition

The Windows Reliability Team focuses on three types of failure: operating system crashes, application crashes, and application hangs. An operating system crash is a catastrophic failure that prevents the system from performing its function and requires a complete system reboot [23]. Operating system crashes are often caused by device drivers or hardware failures. An application crash is when an application terminates with an unexpected exit code. Heap corruption is a common cause of application crashes. An application hang occurs when an application fails to process UI inputs for at least five seconds and is closed by the user, usually by their selecting the "X" close button on the top right corner of the window. Waiting on I/O on the UI thread is a common cause of application hangs. All three types of failures disrupt users and may be accompanied by data loss. The causes of the failures can be diagnosed and fixed using data collected via WER.

In this paper, *we focus on application crashes and hangs*, since they are the most frequent. Windows Vista SP1 data from March 2009 indicate that application crashes and hangs are 20-50X more common than operating system crashes (e.g. failures in 3rd party device drivers). However, we acknowledge that different types of failures may have different characteristics, as discussed in section 7.

3.4 Measurement

The Windows Reliability Team measures failure frequency per day per machine during the initial 15 calendar days after install. At the individual machine level, failure frequency is defined as

$$failure\ frequency_{Machine} = \frac{1}{15}\sum_{i=1}^{15} failures\ in\ day_i$$

At the machine sample level, the trimmed mean is computed over all the machines. Machines are ordered according to their failure frequency. The top and bottom 1% are trimmed, and then the mean is computed over the remaining 98% of the machines. The trimmed mean is more resilient against outliers, which is important for beta versions. Beta versions often have machines running automated tests, which can generate a large number of failures in a short amount of time; untrimmed, data from these machines can pollute the mean.

Three reasons lead the Windows Reliability Team to measure failure frequency during the initial 15 calendar days after install:

- First impressions are set within the first two weeks: Market research shows that most customers establish their first impressions of a product within the first two weeks of use. This first impression persists for a long time and is difficult to reverse. Therefore, reliability during the first two weeks is critical to the success of a software product.
- Reliability changes over time: Reliability of MMSS changes over time due to myriad factors, such as configuration changes, usage scenario changes, training, and avoidance of known problems. Consequently, the mix of time periods within a sample can affect reliability measurements. Constraining the period to the initial 15 days mitigates this affect and helps to ensure comparability across samples.
- Timeliness of the results: MMSS producers need to quickly assess reliability, address issues, and make business decisions pre-release. The first 15 days is a reasonable compromise between the amount of data collected and timeliness of the data.

4. PROBLEM DESCRIPTON AND HYPOTHESIS

Windows deploys multiple beta releases to the general public pre-release. These beta releases help original equipment manufacturers (OEMs), independent software vendors (ISVs) and independent hardware vendors (IHVs) prepare for the new operating system. Beta releases also provide real-world feedback data to help Windows fine tune the system for release. The underlying assumption is that the beta releases are being used with applications, devices, and scenarios that approximate post-release. Consequently, reliability measured on data from beta users will be predictive of reliability in the marketplace.

However, the Windows Reliability Team discovered that, for Windows Vista, the reliability of beta releases were not predictive of post-release reliability. Despite deployment to millions of users and thousand of reliability improvements, reliability measurements appeared to exhibited counter-intuitive reliability regressions in successive versions. This section illustrates this problem with quantitative data and describes efforts by the Windows Reliability Team to diagnose the underlying cause.

4.1 Windows Vista

Figure 1 shows reliability measurements for successive beta releases of Windows Vista. The measurements, showing regression from Beta2 to RC to the final version, i.e. Release-To-Manufacturer (RTM), were misleading because Windows reliability had improved through successive beta releases from fixes to issues identified through failure and debugging information collected on the beta releases. The observed reliability regressions should not be possible. These misleading data hindered Windows from evaluating reliability progress.

The Windows Reliability Team hypothesized that usage differences lead to the anomaly and selected two facets of usage to assess the problem: application coverage and usage duration. Application coverage is the number of applications exercised on a machine during the study period. Only applications that were executed were counted. Only the top 1000 most popular applications that were not shipped with Windows Vista (i.e. non-Windows applications) were considered. Usage duration was the cumulative runtime of a machine during the study period. Other factors could have been considered; section 7 discusses this issue.

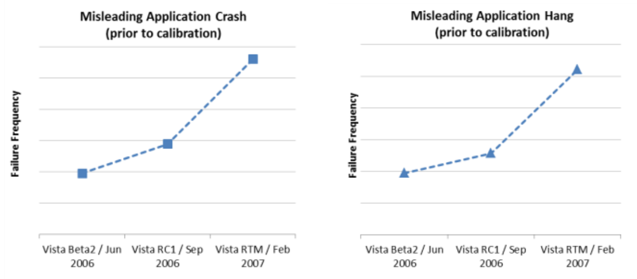

Figure 1: Misleading application crash and hang frequency of Windows Vista beta versions and post-release, prior to calibration

Figure 2 and 3 compare the application coverage distributions for Vista Beta2 machines installed in June 2006 and Vista post-release machines installed in February 2007, immediately after Vista was generally available. The Beta2 distribution was heavily skewed to the left, indicating a large proportion of machines with low application coverage: 18% of machines did not run any non-Windows application; 45% ran between 1 and 3 applications and only 6% ran 10 or more applications. Vista post-release showed two phenomena. First, 24% of machines ran no applications during the first 15 days after installation. This was likely due to high numbers of demo machines, with February being close to general availability; this phenomenon disappeared after a few months. Second, application coverage was higher for machines that exercised at least 1 application: 6% of machines ran between 1 and 3 applications and 44% ran 10 or more applications.

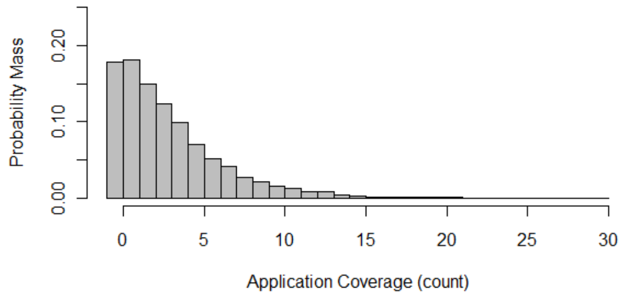

Figure 2: Application coverage of Vista Beta2

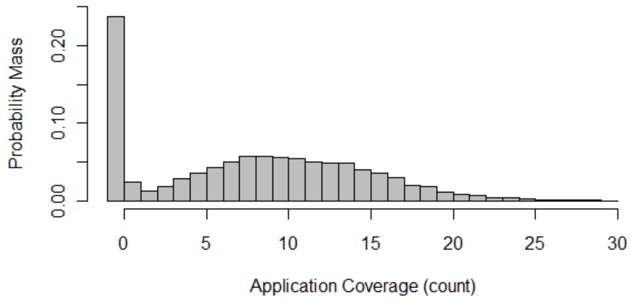

Figure 3: Application coverage of Vista post-release

Figure 4 and 5 compare the usage duration distributions. The Beta2 distribution was bimodal; a large proportion of machines had low usage duration while a small proportion of machines ran almost the entire observation period: 30% of machines accumulated less than 1 day of running time or less than 1.5 hours per day. In contrast, the post-release distribution showed high overall usage time. Only 8% of machines had less that 1 day of use; the mode was between 3 and 5 days or 5 to 8 hours per day.

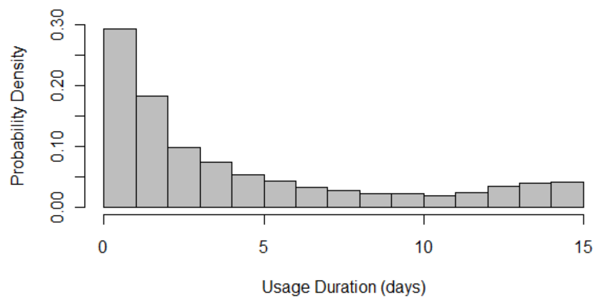

Figure 4: Usage duration of Vista Beta2

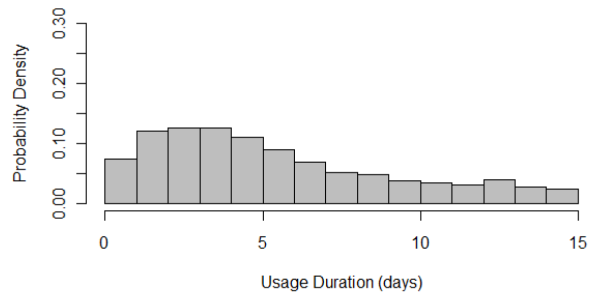

Figure 5: Usage duration of Vista post-release

Figure 6 and 7 show the two usage factors together in scatter plots. Beta2 machines clustered around two areas: bottom left and top left. The bottom left were machines with low application coverage and usage duration; these machines were minimally used. The top left were machines with low application coverage but high usage duration; there machines were likely left idle after install or used by OEMs/IHVs/ISVs to run automated tests. The right half of the chart was sparsely populated, indicating few machines with high application coverage. As expected, application coverage and usage duration were higher post-release; the distribution had a larger proportion of machines on the right half of the chart and no cluster at the top left corner.

Vista data provided two insights. First, the usage differences between pre-release and post-release could be quantified. Second, some beta users, though few in number, were using the beta releases as they would the RTM version. Based on these two insights, the hypothesis was that *if beta users had post-release usage, as measured by the two facets, then reliability measured on those beta users would be predictive of post-release reliability*.

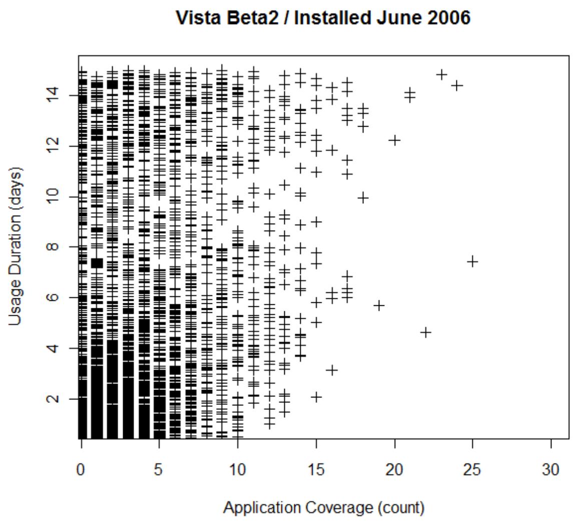

Figure 6: Overall usage of Vista Beta2

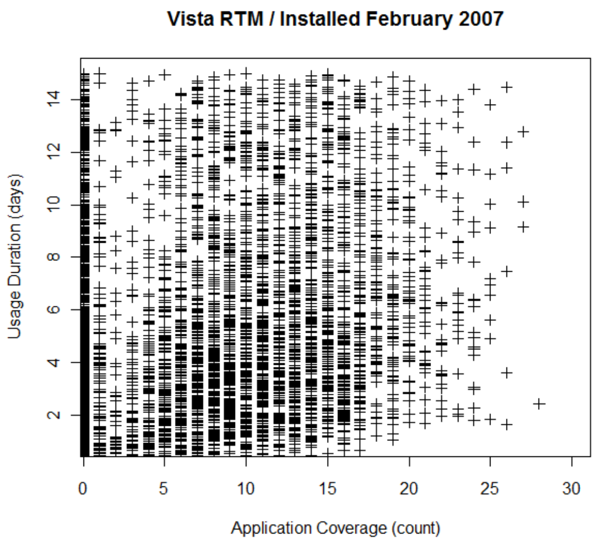

Figure 7: Overall usage scenario of Vista post-release

4.2 Windows Vista SP1

Data from Vista SP1 confirmed the hypothesis. The Windows Reliability Team examined usage and reliability data from Vista SP1 RC machines installed in Jan 2008 and Vista SP1 RTM machines installed in Sep 2008. Figure 8 compares reliability of the two machines samples. The reliability of Vista SP1 RC and Vista SP1 RTM were similar, as were their associated usage.

Figures 9 and 10 contain the application usage comparison, figures 11 and 12 contain usage duration comparison, and scatter plots in figure 13 and 14 examine both dimensions together. The insight was consistent across all three: Vista SP1 RC usage was very similar to Vista SP1 usage post-release.

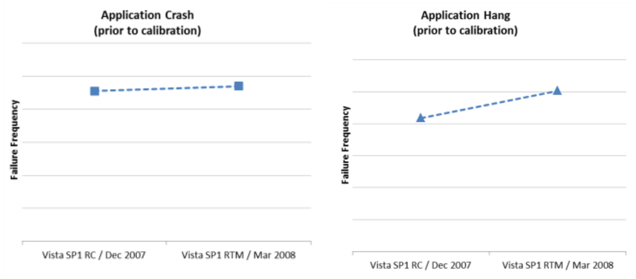

Figure 8: Application crash and hang frequency of Vista SP1 Beta and RTM, prior to calibration

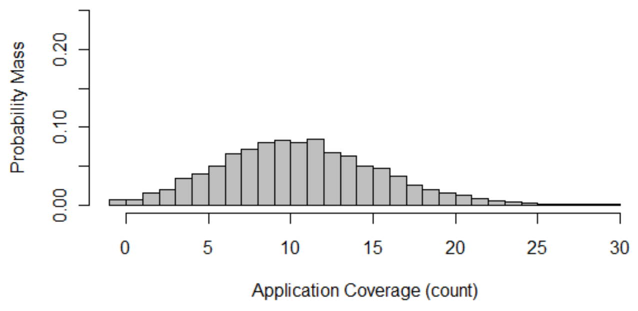

Figure 9: Application coverage of Vista SP1 RC

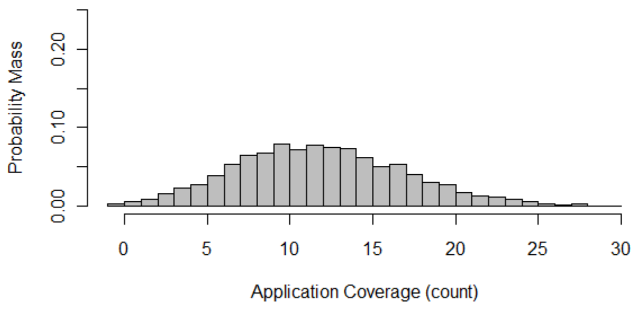

Figure 10: Application coverage of Vista SP1 post-release

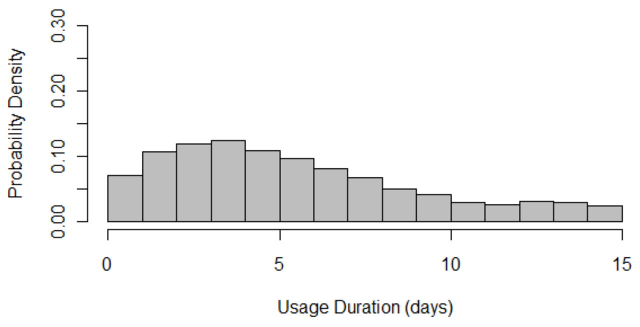

Figure 11: Usage duration of Vista SP1 RC

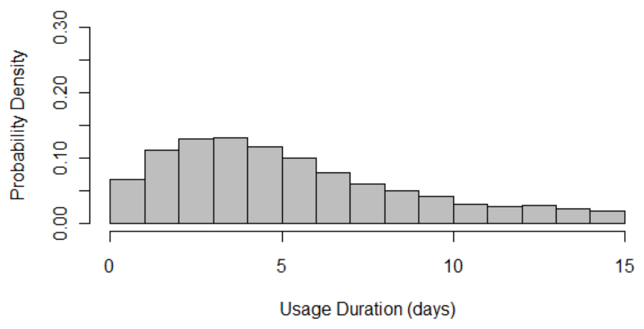

Figure 12: Usage duration of Vista SP1 post-release

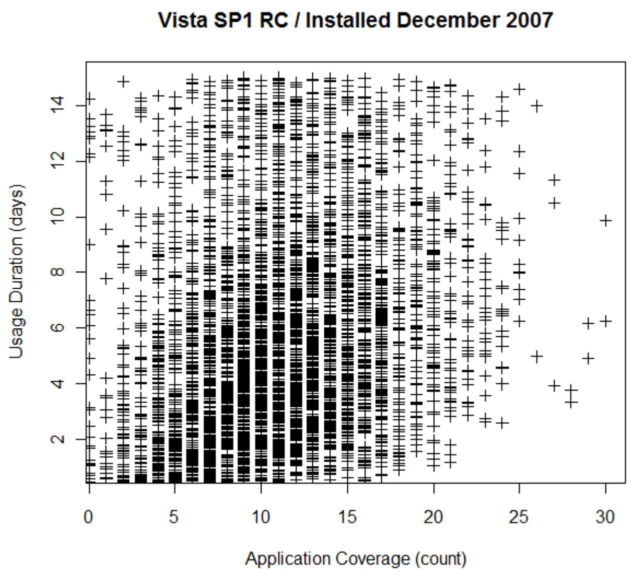

Figure 13: Overall usage of Vista SP1 RC

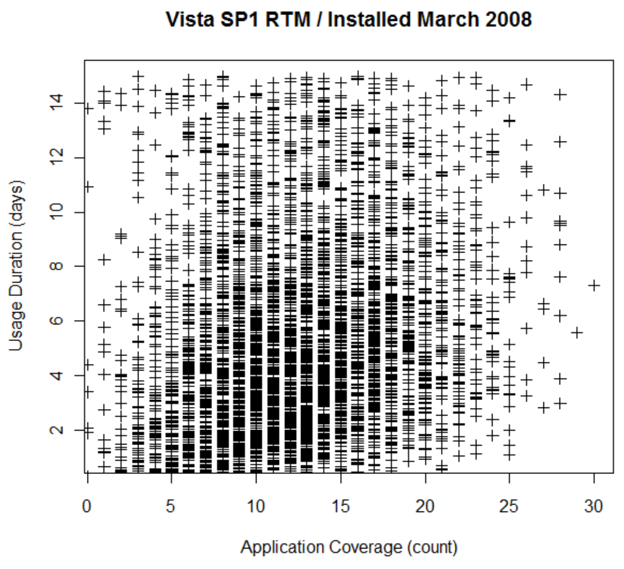

Figure 14: Overall usage of Vista SP1 post-release

Data from Vista SP1 *validate the hypothesis that if beta users had post-release usage then reliability measured on the beta users would be predictive of post-release reliability*. Windows could use this knowledge to recruit only beta users that would use beta releases as they would the RTM version; however, adoption of beta releases were voluntary and uncontrolled. Controlled adoption would be impractical, for example, stopping file sharing sites. Furthermore, it would conflict with other objectives of the beta program, e.g. enabling OEMs, ISVs, and IHVs to verify their devices and applications. Consequently, *the logical approach was to construct a sample of beta users that matches post-release usage*. Section 5 describes the approach used by the Windows Reliability Team to profile usage of each user and to statistically construct a sample of beta users that matches post-release usage.

4.3 Windows 7

Windows 7 data reaffirmed the need to adjust for usage differences. Without calibrating for usage differences, reliability measurements for Windows 7 RC exhibited misleading regressions relative to Windows 7 Beta, as shown in figures 15.

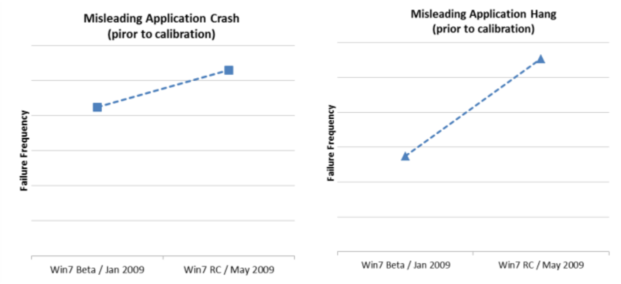

Figure 15: Misleading application crash and hang frequency of Windows 7 Beta and RC, prior to calibration

Pre-release usage of Windows 7 Beta was different from post-release usage (compared to Vista SP1 RTM, the version in the marketplace concurrent with Windows 7 Beta release in figure 14 of section 4.2) and Windows 7 RC usage in figures 17 and 19.

Windows 7 data reaffirm the need for adjustments and provided confidence that the approach was valid. Since pre-release usage of Windows 7 differs from usage post-release, the need to account for usage differences was real (i.e. behavior on previous releases was not a coincidence). Without accounting for usage differences, reliability measurements would have been counter-intuitive. Windows used the approached described in the next section to calibrate reliability measurements and to accurately predict reliability for beta releases of Windows 7.

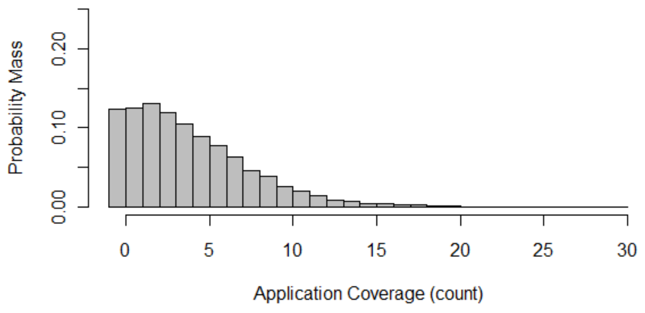

Figure 16: Application coverage of Windows 7 Beta

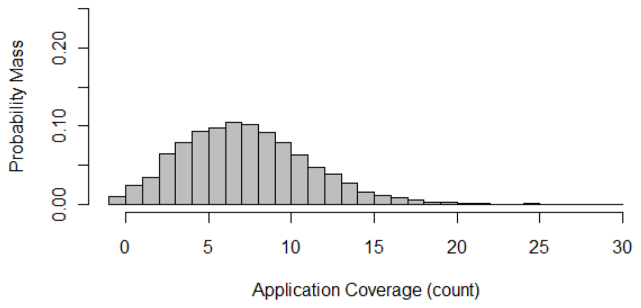

Figure 17: Application coverage of Windows 7 RC

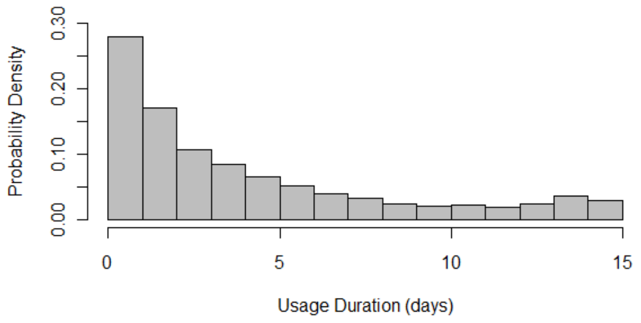

Figure 18: Usage duration of Windows 7 Beta

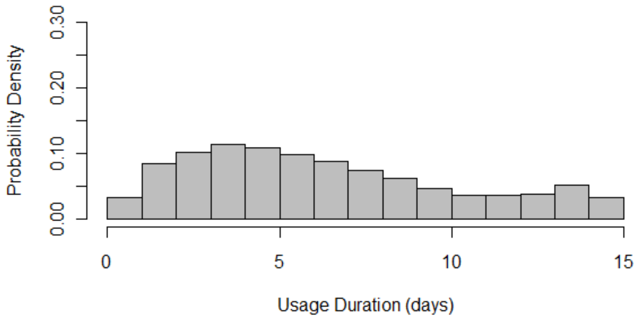

Figure 19: Usage duration of Windows 7 RC

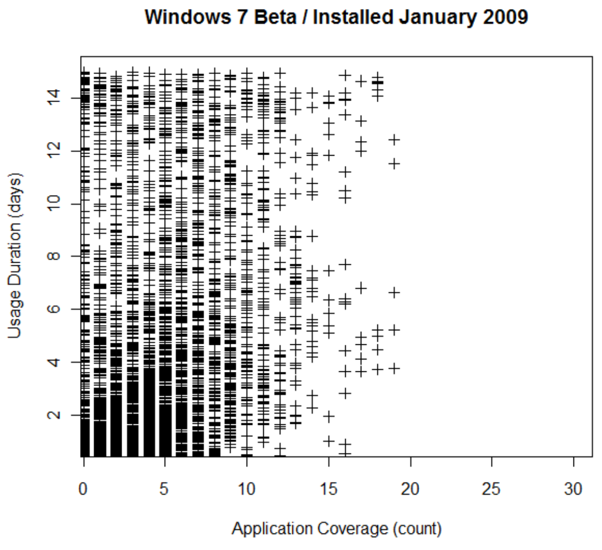

Figure 20: Overall usage of Windows 7 Beta

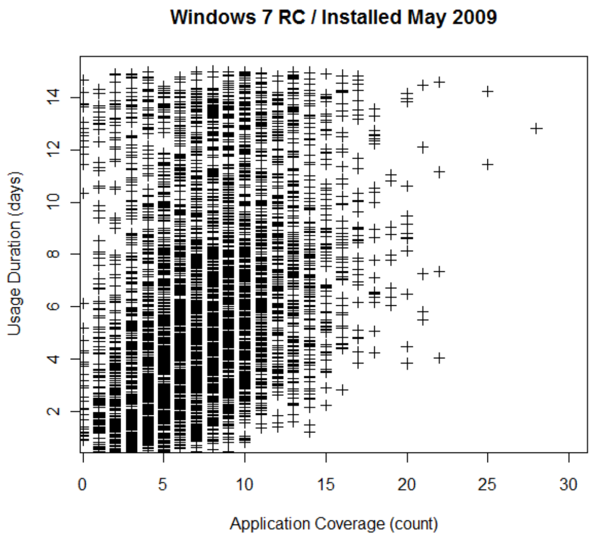

Figure 21: Overall usage of Windows 7 RC

5. Usage Profile-based Reliability Measurement Calibration (UPRMC)

This section describes Usage Profile-based Reliability Measurement Calibration (UPRMC), which uses the insight in section 4 to calibrate reliability measurements. The insight is that *if beta users had post-release usage then reliability measured on the beta users would be predictive of post-release reliability*. UPRMC characterizes the usage profile of a target machine sample using a set of usage factors. Then, for the given reference usage profile, UPRMC adjusts the composition of the target sample to match the usage pattern of the reference sample. Finally, reliability measurement is taken on the calibrated machine sample. The calibrated reliability measurement can be compared to the reliability of the reference sample and to the reliability of other machine samples calibrated to the same reference usage profile.

5.1 Identification of Key Usage Factors

Selecting a set of usage factors is a critical step for two reasons. First, the wrong set of factors can produce ineffective calibrations and may lead to more misleading predictions. Second, the factors must be credible in order to be accepted by an organization, as discussed by prior work in software process improvement [7].

The current version of UPRMC uses two usage factors: application coverage and usage duration. Three reasons contributed to their selection. First, data from Vista machines in the field indicate that application failures rates are more closely related to application usage and machines usage than to machine configurations. Second, examining applications usage in aggregate rather than usage of individual applications is more robust because individual application usage can fluctuate significantly. Finally, the two factors are the consensus choice among the reliability experts in the Windows. Additional factors may be important; section 7 discusses investigations planned by the Window Reliability Team.

5.2 Quantitative Usage Profiling

UPRMC profiles usage scenarios using two-dimensional k-Means clustering over application coverage and usage duration. The k-Means clustering algorithm partitions the machine sample into a specified number of clusters, each having similar application coverage and usage duration. UPRMC currently uses 10 clusters; other cluster sizes were examined but noticeable benefits were not observed. 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 with C_j defined as <application usage$_j$, usage duration$_j$>

Table 1: Usage scenario based on characterization of Vista post-release sample

	Cluster Centers		
Cluster ID	*Application Coverage*	*Usage Duration*	*Percentage*
1	7.0	2.7	18.0%
2	17.0	3.8	10.4%
3	21.5	7.6	5.3%
4	15.4	13.2	4.8%
5	7.8	13.1	5.4%
6	3.9	7.2	6.9%
7	2.4	2.1	9.8%
8	11.8	3.0	17.8%
9	9.1	3.8	11.3%
10	14.0	7.4	10.3%

For example, consider the Vista post-release usage profile from February 2008 in Table 1. The first cluster is centered at 7 applications executed and usage duration of 2.7 days; 18.0% of all machines should belong to this cluster. The reference usage profile can be constructed from any sample. For Windows 7, the Windows Reliability Team uses a Vista SP1 sample taken over one calendar year. This is because Vista SP1 is the most recently release and one calendar year accounts for seasonality effects.

The Windows Reliability Team has considered evenly-spaced bins and classification trees, but found the k-Means clustering algorithm to be better. The k-Means clustering algorithm is more tolerant of outliers, works well for both large and small samples, and can be easily generalized to more than two factors [8].

5.3 Bootstrap Sampling

UPRMC constructs a calibrated sample by re-sampling (i.e. reusing) machines from the original sample to ensure that the calibrated sample exhibits the same usage profile as the reference sample (i.e. percent of machines in each clusters matches the percentage in the reference sample). For the calibrated sample, no cluster is over- or under-represented. 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).

$$\text{Calibrated Sample} = N'_{1..m} \text{s.t.} \forall N_{1..n} \ni N'_{1..m} \text{ and } P_{N'}^{C_i} = P_R^{C_i} \text{for} C_{1..n}$$

Where

$P_R^{C_j} = percentage\ of\ machines\ in\ the\ reference\ sample\ in\ C_j$

$P_{N'}^{C_j} = percentage\ of\ machines\ in\ the\ calibrated\ sample\ in\ C_j$

5.4 Computation of Reliability Measurement

UPRMC computes the calibrated reliability measurement as reliability measured on the calibrated sample.

6. VALIDATION

Vista SP1 data in section 4.2 validate the hypothesis that if beta users have post-release usage then reliability measured on the beta users would be predictive of post-release reliability; UPRMC utilizes this hypothesis to profile users and then statistically constructs a sample of beta users that matches post-release usage. For UPRMC to be valid, it needs to meet three criteria:

- No reliability regression anomalies: reliability is known to have improved between successive beta releases due to numerous reliability fixes; consequently, reliability regressions should not occur. This assumes that there are users using the beta release as they would post-release; we discuss this assumption in section 7.
- Improved prediction accuracy: the calibrated reliability measurements should be closer to the actual reliability in the marketplace. Since the actual reliability of beta releases in the marketplace is not known, reliability of the RTM version is used as proxy. Reliability of the beta release closest to release is compared to the RTM version. Absolute relative error, a commonly used measure of accuracy [10], is used.
- Buy-in from experts in Windows: for successful adoption, a new approach needs buy-in from experts within an organization, as discussed by previous work in software process improvement e.g. Herbsleb and Goldenson in [7]. Therefore, buy-in from experts in Windows is important; these experts typically make a holistic evaluation of the approach.

Figure 22 shows the original and calibrated reliability measurements for pre-release and post-release versions of Windows Vista. The reference sample is Vista RTM machines taken over an entire year. UPRMC removes the counter-intuitive trend in successive beta releases. The absolute relative error, measured from RC1 to RTM, improves 26.6% for application crashes (48.5% to 21.9%) and 38.0% for application hangs (50.6% to 12.6%). The real accuracy improvement is likely more. RTM contains reliability fixes; therefore, the actual RC1 reliability in the marketplace is likely closer to the calibrated RC1 reliability.

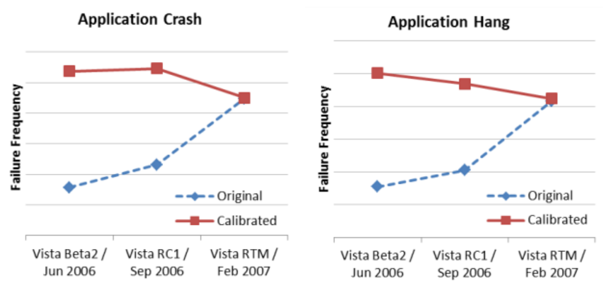

Figure 22: Original and calibrated application crash and hang frequencies of Vista beta versions and post-release

Vista SP1 data in figure 23 verifies that UPRMC can be applied even when beta releases already have usage similar to post-release usage, that is, UPRMC does not adversely affect already accurate measurements. Figure 15 shows calibrated reliability measurements for Vista SP1. The calibrated and original measurements differ by only 8.1% and 7.1% for application crashes and application hangs respectively. The absolute relative error improves 0.5% for application crashes (3.1% to 2.6%) and 6.7% for application hangs (16.9% to 10.2%).

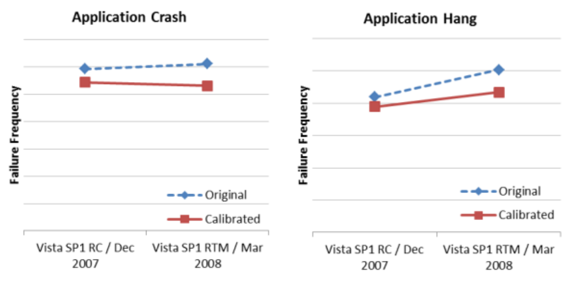

Figure 23: Original and calibrated application crash and hang frequencies of Vista SP1 RC and post-release

Finally, experts within Windows agree that the calibrated measurements are more credible than the raw reliability measurements. Windows has adopted UPRMC and has used it to measure the reliability of beta releases of Windows 7.

7. THREATS TO VALIDITY

This section discusses the four major threats to validity. The Windows Reliability Team has taken steps to mitigate their effects or is planning future investigations to address the issues.

The measurement period may introduce bias. Windows currently measures reliability in the 15 calendar days after release; machines that do not have 15 days of usage are not considered. This introduces possible under-representation of users with problematic initial experiences. For example, a beta user who cannot get a display driver to work and then abandons the beta release after only one day; this machine would not be counted. Windows is currently working on approaches to incorporate all feedback data; however, it is generally accepted that no data will be received from some users.

The usage factors may be insufficient. UPRMC currently characterizes usage using applications coverage and usage duration; however, other factors are available. We hope to investigate other factors, such as hardware configurations, and active application usage (i.e. when there is user interaction). Other measures of reliability, e.g. operating system crashes, may be more related to other factors. In addition, we plan to explore different granularity of factors. Currently, we do not distinguish between applications; however, issues associated with specific applications or classes of applications may not surface if users are not using those applications. We plan to examine application groupings (e.g. office productivity, internet, and games) and individual applications. With additional factors and more granular factors, there will be a bias-variance and feasibility tradeoff. We plan to investigate the right balance of factors and stability of predictions.

UPRMC may not be effective if beta usage is too different from post-release usage (i.e. there is a limit to how much the data can be adjusted). We have evidence that UPRMC is not effective for the Microsoft internal self-host population. We believe that Microsoft employees are not representative of post-release users due to standardization of hardware and software configurations. For example, one anti-virus product is standard for the entire Windows organization. We are working to quantify the operational limits UPRMC.

The issues and solutions described in this paper may be applicable only to operating systems. However, given that Windows is widely-used, this work is of value. In addition, we are collaborating with other Microsoft teams, e.g. Internet Explorer and Office, to investigate the applicability of the approach to other MMSS.

8. CONCLUSION

The ability to exhaustively test mass-market software through in-house testing is limited by the number of usage environments and scenarios. A common approach to predict post-release software reliability is to distribute pre-release versions of the software (i.e. betas) and to monitor its behavior. Using usage and reliability data collected from pre-release and post-release versions of Windows Vista, Vista SP1 and Windows 7, this paper shows that usage of pre-release versions may not reflect post-release usage, resulting in misleading reliability predictions. This paper presents UPRMC, an approach that allows MMSS producers to accurately predict post-release reliability by leveraging usage information to calibrate reliability measurements. Results show that UPRMC produces credible and accurate results and has buy-in from experts in Windows. UPRMC has been adopted and used by Windows, and it is being investigated for use by other Microsoft products.

9. ACKNOWLEDGEMET

Thanks to Gabriel Aul and Michael Fortin for their support of this work, Jason Garms, Vince Orgovan, and Harald Gall for their input and feedback, and members of the Windows Vista and Windows 7 Reliability Team for providing foundational work for the analyses: Scott Graham, Raymond Fingas, Sung-eok Jeon, Yuriy Vasyleha, Lily Yang, Arvind Visvanathan, Daniel Martinez, Elmar Langholz, Valera Zakharov, Desmond Lee, Gretchen Loihle, David Grant, and Ryan Kivett.

10. REFERENCES

[1] S. Elbaum, M. Diep: Profiling Deployed Software: Assessing Strategies and Testing Opportunities, *IEEE Tr. On Software Engineering*, Vol 31, No. 4 April 2005, pp 312-327

[2] NE. Fenton, SL. Pfleeger: *Software Metrics: A Rigorous and Practical Approach.* PWS Publishing Coo. 1997

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

[4] 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 2009*

[5] K. Goseva-Popstojanova, A. Singh , S. Mazimdar , F. Li Empirical Characterization of Session-based Workload and Reliability for Web Servers, *Empirical Software Engineering Journal*, Vol 11, No 1, Jan 2006, pp 71-117

[6] J. Gray: Why Do Computers Stop and What Can We Do About It. *Intl. Conf. on Dependable Systems and Networks*, 1986, pp 3-12.

[7] J. Herbsleb, D. Goldenson: A Systematic Survey of CMM Experience and Results. *ICSE*, 1996, pp 323-330

[8] A.K. Jain, M.N. Murty, P.J. Flynn: Data Clustering a Review. *ACM Computing Survey*, Vol 31, No 3, Sep 1999, pp 264 - 323

[9] D.R. Jeske, M. Akber-Quereshi: Estimating the Failure Rate of Evolving Software Systems. *ISSRE*, 2000, pp 52-61

[10] PL. Li: A Catalog of Techniques that Predict Information about the Count or Rate of Field Defects. *CMU tech report CMU-ISRI-06-122*

[11] PL. Li, R. Nakagawa, R. Montroy. Estimating the Quality of Widely Used Software Products Using Software Reliability Growth Modelling: Case Study of an IBM Federated Database Project. *ESEM*, 2007, pp 452-454

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

[13] M. Lyu: *Handbook of Software Reliability Engineering.* McGraw-Hill, 1996.

[14] Microsoft Corp, http://www.microsoft.com/products/ceip/EN-US/default.mspx.

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

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

[17] B. Murphy, T. Gent: Measuring System and Software Reliability Using an Automated Data Collection Process. *Quality and Reliability Engineering International*. Vol 11, pp 341-353.

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

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

[20] N. Nagappan, T. Ball, Z. Zeller. Mining Metrics to Predict Component Failures. *ICSM 1998*, pp 24-32

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

[22] MC. Paulk, CV. Weber, B. Curtis, M.B. Chrissis: *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison Wesley, 1995.

[23] M. E. Russinovich, D. A. Solomon, Windows Internals (5th Edition), Microsoft Press, 2009.

[24] N.F. Schneidewind, T.W. Keller, Applying reliability models to the space shuttle, *IEEE Software*,Vol 9,No 4, Jul, 1992, pp 28-33

[25] K. Weyns, H. Martin: Sensitivity of website reliability to usage profile change. *ISSRE*, 2007, pp 3-8

[26] A.P. Wood: Software Reliability from the Customer View, *IEEE Computer*, Vol 36, No 8, Aug 2003, pp 37-42