

2009

PACIFIC NW SOFTWARE
QUALITY
CONFERENCE



MOVING
QUALITY
FORWARD

OCTOBER 27-28, 2009

Conference Paper Excerpt

From the

CONFERENCE
PROCEEDINGS

Permission to copy, without fee, all or part of this material, except copyrighted material as noted, is granted provided that the copies are not made or distributed for commercial use.

Retrospective Analysis and Prioritization Areas for Beta Release Planning Improvement

Ajay Jain
ajjain@adobe.com
Adobe Systems Incorporated
Noida, India

Abstract

The beta release of a product is an official pre-released version of the product. The product at this stage i.e. at the beta testing stage is generally considered to be fully functional and in a “close-to-release” state. Typically, the participants in beta testing are a small group of classified users who are invited by the product company. In some cases, however, the beta product is open for public scrutiny.

This paper does a retrospective analysis by taking a project and product as a case study and evaluates the testing effort invested by the testing team in beta or prerelease testing and makes a statistical comparison with that of the testing effort invested in regular feature testing during a complete product cycle. The data variables used are in terms of resources, the time spent in bug triaging, beta readiness certification, and documentation. The data is then analyzed to measure the effectiveness of the beta or prerelease testing program in terms of the product quality improvement achieved through the beta or prerelease testing program. Post-analysis recommendations are also suggested to help a beta testing program achieve a high quality product without overshooting the testing cost.

The strategies discussed in the paper shall help a test manager in achieving a much higher ROI (Return on Investment) from the beta release test program not only in terms of testing effort spent but also in improving the product functionalities and optimizing the organization cost.

About the Author: Ajay Jain

Ajay Jain has more than nine years of industry experience in testing and validation of mobile and desktop publishing software. He is currently working with Adobe Systems, Inc. as Quality Engineering Manager, leading and managing the Instantiation (Deployment, Provisioning and Services) QE team for Adobe Creative Suites products. Prior to Adobe, Ajay Jain worked with industry majors like Lucent Technologies (Bell Labs Development Center) and Skyworks Inc, where his experience ranges from building a startup testing team from scratch to a resource optimized, efficiency driven team certifying multiple product lines.

Ajay has an active interest in knowledge sharing on best processes and practices and has written and published several papers for *Quality Matters*, Adobe Quality Newsletter (Internal), and hosted birds of feather session at Adobe QE Summit. Two of his papers, “*Power of Glide Path: Statistical approach for controlling and adding Predictability in a testing project*” and “*Well processed, well done – Suggesting 5 light weight processes for optimizing software test management*” earned him speaker invites to the 8th All India Software Testing Conference 2008, and SPICE Conference 2009 respectively. He also has a patent invention–Meritorious Disclosure award to his credit.

Ajay holds a B.Tech (Engineering) degree from Delhi Institute of Technology and a Specialized Diploma in Business Administration from the Institute of Management Technology. Reach him at ajjain@adobe.com.

Introduction

Beta testing is a strategic milestone in a product life cycle and helps in building and securing a strong product positioning among the potential end users or consumers. Therefore, it is extremely important to plan and manage the testing effort in bringing a product to a beta testing stage.

Releasing a quality product on time, while restricting the cost to a minimum level, is a challenge for the test manager. She has to make sure that the right resources are invested at the right time on right activities. There is almost a zero scope for any testing effort getting wasted or going into unnecessary activities.

For a test manager, the beta testing phase involves complex planning and meticulous execution. A product's beta (a successful one) is a major milestone for any team and organization. When the product wins accolades and receives positive feedback from potential customers, it is also a big win for the testing team because a successful beta establishes the testing coverage and product stability achieved while doing the testing of the product.

Primary reasons for releasing a product for beta testing are:

- Platform to introduce the product and features to wide audience.
- Receive bugs and compatibility issues on existing functionalities from the end users.
- Receive feedback and suggestions on existing features and new features requests.

Testing is always a crucial, challenging, and interesting activity for a test manager especially if she is managing multiple time bound projects with a limited set of testers. She cannot afford to lose the quality of any project, she cannot afford to delay any project, and she cannot afford to burn out the team.

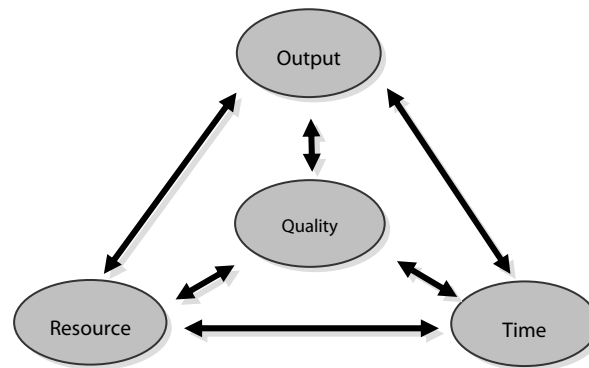


Figure: The Dimension of Productivity concept

This paper takes a case study of beta testing done for Adobe Creative Suites 4 family of suites. The manager managing the complete testing team owns validation of six products with seven head count of test engineers in her team. The testing cycle of each of these products is 12 months. She is now conducting a retrospective analysis of the last projects done and aiming for an effectiveness evaluation of the testing effort of the beta or prerelease testing program.

Time and effort investment in beta testing calculation

Beta testing involves the following major activities:

1. Beta Readiness Certification

Before selecting a build candidate for the beta release, the product or application always goes for a full testing cycle (with respect to offered features) within the internal team. This full cycle covers all stages starting from requirements validation to feature test case execution. A good strategy is to create a profile of the end user's machine, platforms, and the most likely products already installed and available on the user's machine. This strategy helps to design and observe the product behavior in application coexistence scenarios. An application coexistence scenario unearths several unexpected issues because of non-compatibility of two different applications. For example, the installation of Adobe Creative Suite (where Adobe Photoshop is one of the 10 products constituting the suite) on a machine where the Photoshop trial version is already installed might result in some kind of an installation conflict. A common area where conflict implementation is important and interesting is the coexistence of prerelease and trial versions with a full licensed version.

The readiness certification also includes validating major bug fixes that were reported on the feature. The aim of the certification is to make sure that the application is working fine on all major platforms and is sufficiently stable in all general and exceptional conditions. Once a substantial confidence is achieved on the product's functionalities, only then the product is selected as a beta candidate.

The test manager now does an effort calculation exercise to determine the effort that the test engineers have invested while certifying the beta build candidate.

Effort Calculation Category: Beta Build Readiness Certification (per Product, per platform)
Testing effort – 6 person days
Number of products - 6*

**Note: Test manager is managing six products in her team. The data above is based on the execution time for beta build test cases. The feature testing effort is considered part of the regular testing effort and is not included in the beta testing effort. The beta testing effort is the dedicated effort that the QE (tester) had put for determining the candidature of a specific product build as the beta build.*

2. *Documentation for beta testers*

Documentation for beta testers is generally referred to as release notes. A specific release note should be included with each and every beta build. Release notes should be prepared with utmost care because this is the first document that will help the user to get the product rightly installed on their machines. Any misinterpretation at the release notes level may result in the flow of unnecessary and invalid bugs from the tester's side.

A *release note* should carry the following information:

- a) Detailed installation or setup instructions
- b) Description of existing known issues, limitations, and other relevant information
- c) Hardware and software configuration, platform, system, and other technical requirements, and specifications for product installation or functionalities
- d) Available features and features recommended for testing
- e) Methods (Bug Database/Tools/Forums) for reporting bugs and issues
- f) Disclaimer*

**Because the product is being released only for beta testing, there is no guarantee of a bug-free product. There may be cases where the user's machine might face operation disruption, or BSOD (Blue Screen of Death) or other critical crashes originating might be because of the beta product. It is always advisable to have a Disclaimer section, clarifying no responsibility for data loss on the user's machine.*

Effort Calculation Category: Documentation (per product, per platform)
Testing effort – 0.5 person days
Number of products - 3

3. *Bug triaging*

Bug triaging consumes a major part of a tester's time. As soon as the product (or application) is out for beta testing, there is always a big influx of bugs from the beta testers.

Bug triaging includes the effort spent by the tester in:

1. Reviewing a new bug, categorizing (or re-categorizing) the bug against the right product areas in the bug database
2. Simulation of bug scenarios or failures in order to get the logs, especially if the bug description or log files are incorrect or incomplete
3. Update and closure of bugs with right details

Based on the average time spent on bug reviews, bugs are divided into 3 categories:

Category 1 bugs – These are bugs that take less than 10 minutes to triage; these bugs are generally not new from the product's perspective and are already there in the team's bug database. In our case of creative suites, the number of Category 1 bugs is 220.

Category 2 bugs – These are bugs that take close to 30 minutes for the complete review. Typically, these bugs have good bug description based on which they can be simulated or reproduced in-house, even without log files. In our case of creative suites, the number of Category 2 bugs is 130.

Category 3 bugs – These bugs take the maximum time (sometimes several hours) to simulate. The beta testers reported incomplete bug descriptions and did not attach log files. The in-house tester has to extract details from the beta testers on the exact reproducibility conditions. As a result, Category 3 bug troubleshooting is tough and time-consuming. In our case of creative suites, the number of Category 3 bugs is 50.

<p><i>Effort Calculation Category: Bug Triaging (Including all prerelease/beta drops)</i> Total beta bugs reported: ~400 (Category 1: 220, Category 2: 130, and Category 3: 50) Total bug triaging effort: $220 * 10 \text{ minutes} + 130 * 30 \text{ minutes} + 50 * 120 \text{ minutes}$: ~12000 minutes. <i>Average Bug Triaging effort:</i> (Total triaging effort / Total beta bugs reported) : $\sim 12000 / 400$: ~30 minutes per bug</p>

The data is being rounded off to a whole number in some cases to simplify calculations.

4. Communicating with beta testers on forums/e-mails

Keeping the beta testers up to date on the status of issues, providing a quick clarification to their posted queries, is always desired from an effective beta program. Forums provide a platform where the beta tester community comes together. In the project discussed in this article, beta testers had added 250 posts on the beta forum. The average time spent by an in-house tester on a forum post was 5 minutes. However, some posts required around an hour of investigation and discussion. Also, the nature of participation was analyzed and it was found that only 5-7 out of the 50 beta testers were regularly posting questions.

<p>Total Posts: 250 Average time spent (per forum post): 5 minutes Total time spent on the forum response: $250 * 5 = 1250 \text{ minutes} = \sim 12 \text{ days}$</p>

Assumptions:

- 1) Time invested in management-level meetings is not added to the data estimation.*
- 2) 1 person day = 8 working hours*

Total effort invested in beta testing:

Table 1: Summarized view of testing effort invested in beta testing

Effort Calculation Category	Time Spent (in person days) (Per product / Per Platform)	Number of Products	Number of Platforms	Total Time (in person days)
Beta Readiness Certification (a)	6	3	2	36
Documentation (b)	0.5	3	2	3
Bug Triaging/Troubleshooting (c)				25
Forum Responses (d)				12
Total Time (a + b + c + d)				~75

Therefore, the total time invested on each beta or prerelease testing is 75 person days.

If a business unit or product team is planning for 3 beta releases, the time investment will be 75*3 person days i.e. 225 person days.

Now, let us do an analysis of the beta reported bugs/failures/issues. Here are the different charts derived from the ~400 bugs logged through the beta or prerelease program.

Chart 1: Beta program bugs classification as per their final closure state (Fixed/ Withdrawn)

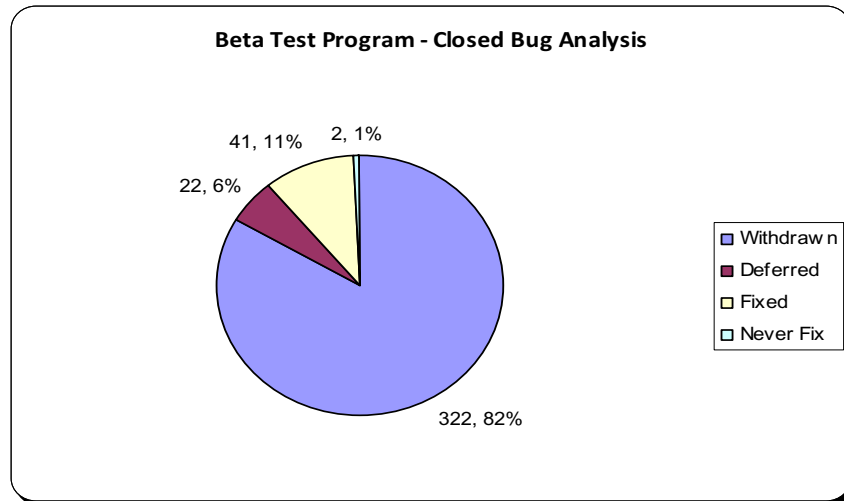
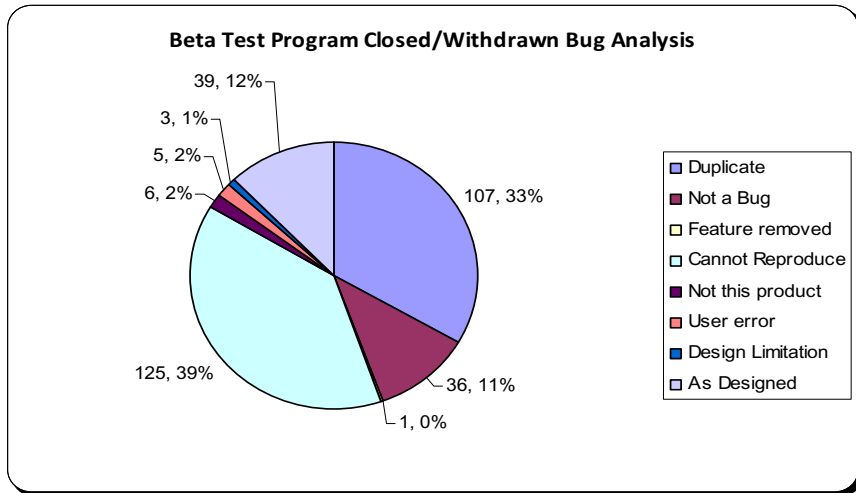


Chart 2: Beta program bugs classification as per their Closed-Withdrawn reasons



Here are the charts derived from the ~1400 bugs logged during the regular feature test cycle.

Chart 3: Regular feature test program bugs classification as per their final closure state (Fixed/Withdrawn)

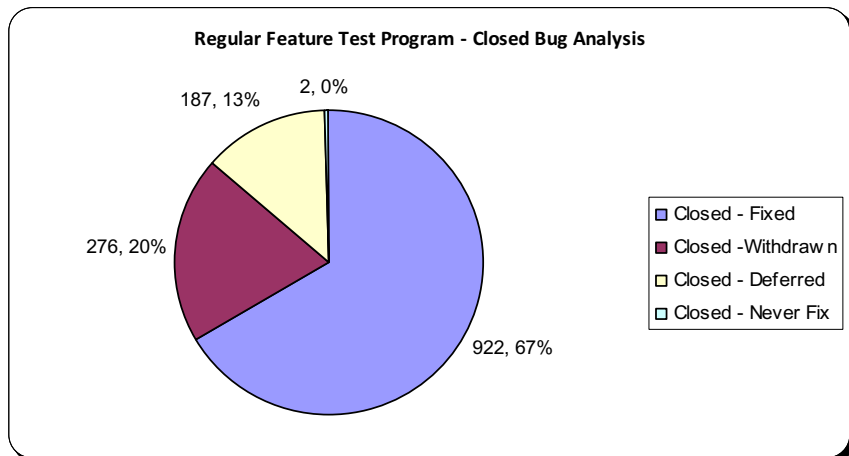
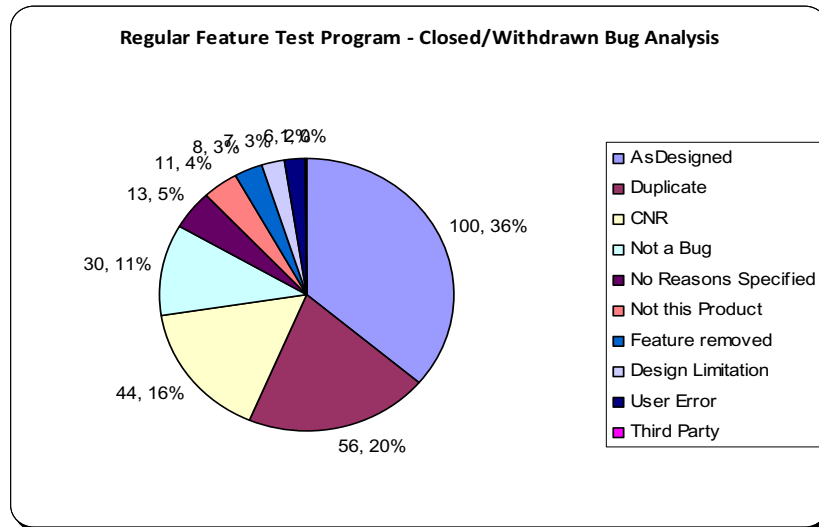
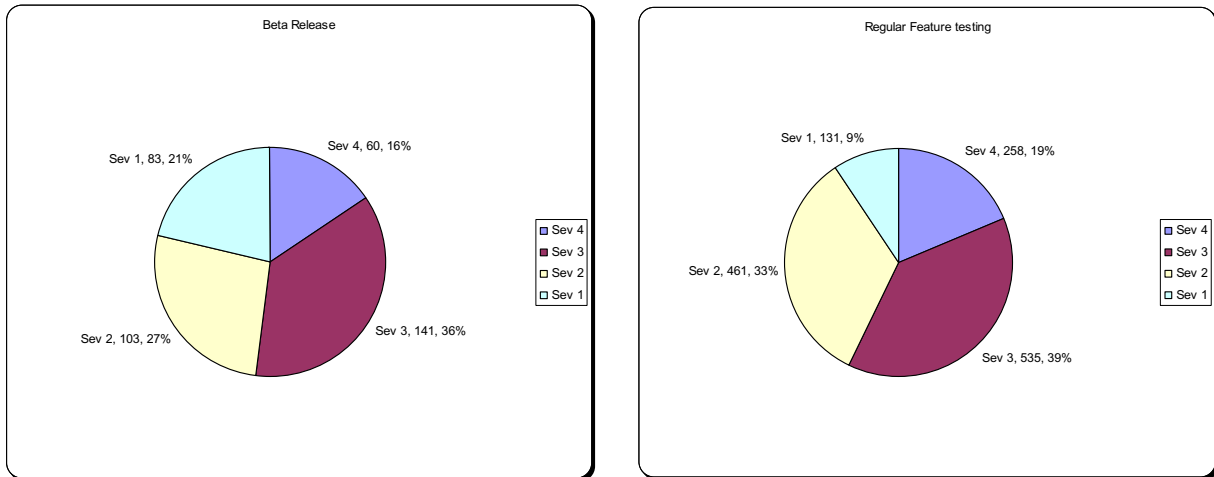


Chart 4: Regular feature test program bugs classification as per their Closed-Withdrawn reasons



CNR stands for “Can not reproduce” i.e. bugs which the core test engineer is not able to reproduce in house.

Chart 5: Severity distribution of bugs



Calculation of time/effort investment in regular feature testing

Total person days invested in feature testing = Number of testers * product cycle (number of months) * time spent on feature testing
= 7 * 15 * 22 days * 0.7
= ~1600 person days

Assumptions:

- 1) A tester is spending 70% of his/her time in writing and running test cases, remaining 30% is being invested in handling bugs.
- 2) The testing team is of 7 testers.
- 3) The product cycle is of 15 months.

Comparative analysis: Prerelease beta test program effort vs. regular feature test effort

S. No.	Areas	On Prerelease/Beta test	On Regular Feature test
A.	Number of days spent by the tester	225	1600
B.	Number of bugs detected	387	1387
C.	Number of valid bugs (Fixed + Deferred)	63	1109
D.	Number of bugs closed as Fixed	41	922
E.	Average ROI per fixed bug (D/A)	0.18	0.57
F.	Bug success rate (D/B)	0.11	0.66

Findings:

As per the data, it was found that during a *regular feature test cycle*:

- 1 *valid (Fixed + Deferred) bug* is detected after every 1.1 days of QE effort.
- 1 *product improvement (Fixed) bug* is detected after every 1.73 days of QE effort.

On the other hand, it was found that during a *beta test cycle*:

- 1 *valid (Fixed + Deferred) bug* is detected after every 3.57 days of QE effort.
- 1 *product improvement (Fixed) bug* is detected after every 5.48 days of QE effort.

Beta testing is a strong platform which sometimes brings in critical bugs which were not detected during the months of regular feature testing.

This data helps the test manager to re-evaluate the process that is being followed while doing the beta testing. She should look out for areas to optimize in order to get maximum ROI from each QE resource being invested to the product and project.

Some of the prioritization areas that she can consider for improving the beta test phase ROI are:

- Optimizing the beta test program frequency. In the case discussed here, if the test manager reduces the number of beta releases by 1, she might be able to save an average of 30%* of QE effort. She can re-route the saved QE effort to other critical activities like regular feature test cycle.
- Defining a good communication method for beta testers. The above data shows 33%* count of the Closed or withdrawn bugs are of duplicate nature. This might reflect a case where the testers are not coordinating or communicating with each another before reporting bugs. Many of the beta testers are logging the same failure as different bugs and therefore increasing the triaging effort of the product QEs. The test manager can now design a

mechanism for logging a single bug, allowing other testers to add their votes to the same bug. This way, the bug count will remain one, QE effort will remain as a single unit, and all communication regarding the bug will be done through a single channel.

**Direct mapping of the cost saving is not an ideal case because each bug differs in its criticality and severity. The above data mapping is done to suggest a need for process improvement indicating a save and optimization of resources and time.*

A quick look at the bug severity distribution for regular feature testing bugs and Beta release reported bugs are as follows

With these strategies in place, a test manager achieves much higher ROI on the QE effort during the complete product cycle. Another way of looking at the benefits is that this strategy ensures the success of both the beta program and the regular testing cycle while keeping a perfect balance of organizational cost. While beta testing plays a significant role in a product life cycle and especially in cases where internal bug count slows down, it is extremely necessary to plan out the testing in all effective means to gain benefits at all levels across bugs, features getting tested, resources that are being used etc.

References:

1. *Dimension of Productivity concept: Metrics and Models in Software Quality Engineering, IInd Edition, Stephen Kan.*