Exploring, discovering and exterminating Bug Clusters in Web Applications

Kim Davis, SET SQA Director, My Virtual Model inc.
Kim@myvirtualmodel.com
www.myvirtualmodel.com

Robert Sabourin, President, AmiBug.com
Rsabourin@amibug.com
www.amibug.com

Abstract

In e-Commerce applications, time attrition, fleeting turbulent requirements and lack of documentation make testing an exercise in managing people who are entangled in constraints. Our experience shows that a good risk-based bug flow process is essential. However, tracking myriads of little web site bugs in a fast-paced environment is difficult in practice. The good intentions of web integrators, scripters and web managers often disappear, in the name of competition, in front of customers demands of an ever-faster site development.

Fortunately there may be a way of dealing with this difficulty without sacrificing features or quality in such an environment: *why not stop tracking the myriad little web sites bugs* – and *start going after the bug clusters!*

Bugs are social animals, they tend to associate by affinity of causes. They may come from a variety of causes. Major causes are usually related to requirements, be them incomplete, misunderstood, unclear etc., and turbulence, i.e. the effect of change of a project variable in a short timeframe.

This results in the e-Commerce application having areas of uncertainties, which may require large amounts of rework. These areas of uncertainties define clusters composed of the bugs that have natural affinities of causes.

So instead of tracking hundreds on individual bugs in an extremely short timeframe, which is not efficient, we instead focus on finding the important areas of uncertainties, which are easier to locate and track than individual bugs. We then perform fast probabilistic root-cause analysis on them to determine probable causes pairing developers and testers to identify and correct the defects causing them.

Motivation

80/20 Rule

We assume that generally the 80/20 rule can be applied to our testing activities. It is generally assumed that 80% of the important bugs are identified with 20% of the testing effort. It is unfortunate that we can not figure out in advance which 20% of the testing effort will find these important bugs. Can we save time and effort by focusing on larger structures of bugs, i.e. *Bug Clusters*?

One thing we know for sure is that we are severely restricted in the amount of time available for testing. So we want to focus our testing efforts on finding as many important bugs as possible in a short time, and present to management the state of the system under test so that informed business decisions can be made.

Apply successive approximations

Optimization techniques involve finding the best solution, or fit, to a problem by running a series of experiments, varying certain parameters each trial and then judging whether the results are optimal, or closer to optimal, based on our previous results and experiences. Purely statistical models are sometimes used in software testing (such as clean room based testing) to confirm that sufficient test cases have been run to prove or disprove a reliability hypothesis.

Bugs are caused by defects. Defects are introduced into a system by people using processes or tools. Defects may be introduced at any time of the development process, requirement, analysis, design, development, integration, deployment etc. The bugs found during testing are a very important input to the detection of defects in the system under test.

In the context of projects at MyVirtualModel, Inc. we know that we will fix the bugs deemed important from a business and technical point of view. Indeed the priority of bugs, indicating if and when it must be fixed, is a *business decision*.

If we find some bugs we may suspect that there is some a relationship between them. This can be considered a weak relationship which, as a result of more testing, and further bug identification, can be made successively stronger.

We use an iterative approach when testing the system. Each iteration is based on exploring the application to further enhance our knowledge of the software and validate any hypothesis about functional areas of the application exhibiting instability.

Identify the cluster

Ideally we would like to confidently identify a bug cluster without having to finding all of its individual constituent bugs! This would save us a lot of time and effort. But how can bug clusters be identified?

Bugs are social animals, they tend to associate by affinity of causes. Our practical experience tells us that after a project, when reviewing data about bugs identified and associated defects corrected, there is generally a series of bugs related to the same root cause. If we had only found one bug of a certain "family" or cluster, then we could have fixed the lot of them without even being aware of their existence! As Brian Marick defines in "Evaluation Test Suites", Conference Proceedings, Quality Week 2000, (with James Bach and Cem Kaner) a perfectly effective test suite is one that reveals at least one failure for every fault in the program.

Prioritize and Exterminate Bug Clusters

Once suspected bug clusters are identified we can prioritize them relative to other bug clusters in the system. We can make a cluster based decision which will apply to bugs which have not yet been identified.

Fix bugs that you did not find!

By fixing the fault in the cluster of bugs we correct failures which may have existed but were not identified and recorded.

My Virtual Model Community

My Virtual Model concept and community

My Virtual Model has created a network hosting a virtual community. Members of the community create electronic replicas of themselves, called Virtual Models. Shoppers can experiment with fashion, dressing their Virtual Models with virtual garments without the intimidation of store dressing rooms.

Users log on to the site to begin building their model. By answering specific questions, users can create a virtual replica of themselves with exact measurements, skin and hair color. Users can then try on various looks using their model in a virtual dressing room. Shoppers can choose to accept advice from a "Fashion Advisor" on the basis of their body type, coloring, and fashion goals.

Over 1.5 million models have already been created at MyVirtualModel.com affiliated sites.

Forrester Research described My Virtual Model™ as "the most viable alternative" among tools which support online apparel sales.

3D, Web and Fashion industries collide

Requirements for all My Virtual Model development efforts include 3D animation, the latest is Web GUI and interactive technologies. My Virtual Model clientele are members of the Fashion industry which operates under very tight seasonal delivery pressures. You cannot delay the spring fashion season due to a software bug! All projects have fixed timeframes.

Visualization

My Virtual Model provides the ability to mix and match garments and colors that are not only photo-realistic but are also right-sized to eliminate concerns of true fit or appropriate sizing. It includes :

- Photo realistic images of garments
- Close-up views of garments (Zoom in)
- Multiple views of garments (360 degrees rotation)
- Try-on of the garment on body with real measurements

Personalization – Virtual Identity

Virtual Models are personalized for each member of the community. All characteristics of a model can be adjusted as the end-user sees fit. If a new characteristic is modeled, then special consideration must be made for previously existing models to operate with the new characteristics. Backward compatibility is critical.

Users build what is called a Virtual Identity. The first layer is a physical representation in the form of the 3D Virtual Model, which is tied to the garment try-on service and the e-apparel market. Other layers tied to other services, for example financial, may be added later so that the model will resemble more closely the user in order to better serve it.

E-Commerce

My Virtual Model technologies are an integral part of the e-Commerce offering from client vendors. We must be able to provide specifics regarding the style, color, size and all other characteristics of garments purchased via B2B interchange of order information. Encrypted XML is used extensively to interchange the required information. My Virtual

Model must be non-intrusive to the client e-Commerce site and must be able to interact with all clients solutions.

Network of interoperating web sites – Mobility

My Virtual Model is implemented as a series of independent web servers. For each client there exists at least one of each of the following:

- Client e-Commerce
- Fashion Server Site
- Model Server Site
- Various Data Servers

A three tier architecture is used, using the familiar Web Server / Application Server / Data Server layering.

Virtual Models are mobile. Users can access their models on any of the affiliate sites, can travel from site to site, and can send their Virtual Model by e-mail. Mobility pumps the heart of the My Virtual Model Network.

Business risks

Business risks must be reviewed on a frequent basis to ensure that testing priorities are aligned with the corporate realities. We may have to react to a competitive threat or new short time frame opportunity.

Generally business risks are driven by:

- Fixed timeframe projects tied to industry dates for seasonal collections of garments
- Large and demanding User community (consistency, performance, reliability)
- Functionality always tied to market "$$" related revenues e.g. a new type of model can be introduced to target a new market (for example tall men)
- New business model is evolving (including revenue) as project continues!
- Organizational change
- Requirement turbulence

Typical Project Example

The typical project encountered has a tight time frame. From the moment clothing is in hand (hundreds of garments), to commercial deployment, takes a maximum of 12 weeks elapsed. Often projects require less than 8 weeks to complete.

The user community per customer includes 100s of thousands of users, all clients are large retailers, thus generating many hits fast!

The retail business is similar to the Web business in that customers offer no forgiveness! If the software or site is not operational they will move elsewhere.

It is very problematic to remove, or trading down, functionality in order to hit a release target. In fact quite the opposite happens and due to compelling business requirements the release date is often moved up!

Technical Risks

The projects involve considerable technical risks. Technologies used are continuously changing and evolving. The company addresses a segment of the market which continuously demands the latest and greatest web widgets!

So our projects experience fast technology churn. The technologies being used in production releases are new to the company, marketing team, developers, testers and system integrators. Developers and testers are learning about the limitations of the solutions on the fly and often have to adapt architectures in mid project.

Technologies are often used in a production environments with limited knowledge. We have limited time for training and for evaluation of the technologies.

Staff turnover can have a devastating effect on such projects. New programmers are not familiar with the existing code base and can accidentally miss something when adding a new feature or adapting to a new embedded technology.

Developers are often not familiar with the "live" production parameters. Often the customer has not decided whether parts of the server component will work on NT or Unix based computers.

Whenever requirements change there is a risk of changing development priorities which in turn will impact the code base. How do we elegantly drop what we are doing while not accidentally breaking something!

Changes in mid-project included:

- A new application server framework, at the business logic layer, moving to an Apache server with JServ to one with JRUN.
- A new JSP driven presentation layer.

Important technical risks were also introduced due to the reuse of existing code in a new context.

Testing Objectives & Risk

Summarizing the specific technical and business risks associated with a project lead to the identification of primary testing objectives. We were able to prioritize these based on our knowledge of the areas of highest risk. Testing effort was spread across testing assignments proportional to the associated risk.

We considered the potential areas of instability to be:

- Functions particularly related to new technologies
- New stuff or changed stuff
- New functions added due to business

Exploratory Testing Approach

An exploratory testing technique was used to help find bugs, and identify problem areas of the application. To quote James Bach, *"In operational terms, exploratory testing is an interactive process of concurrent product exploration, test design and test execution."*

Exploratory testing is a systematic approach. It allows for the concurrent design and execution of tests. As the application is explored a detailed record is kept of the areas explored. Information is captured on standardized templates. Information gathered includes:

What was tested?
How was testing done?
What was discovered about the application?
What bugs were identified?
What oracles were used in an attempt to validate results?
What new discoveries may be of interest to future exploratory testers!?

A senior test lead is responsible for triaging test assignments to various members of the testing team.

The test lead uses knowledge gathered in testing combined with the risk assessment and the potential areas of instability to define the next series of test assignments.

The test lead is operating at the testing mission control. Different testers act as emissaries exploring along the directions indicated by the test lead. The test lead collects all results and saves them in a repository (file system) as a map which is being built up as the application is further explored.

Testing assignments include specific objectives. Testers must use their own judgement in following them. If a tester uncovers an area of instability on the route he may explore it or document it for future exploration. The decision has to be made on the spot.

Bug Cluster Identification

Mapped areas of instability define clusters composed of the bugs that have natural affinities of causes. As the test lead collects results from each "chunk" of exploration, two important assessments are done with his peer in development.

1- All new bugs are reviewed to determine if they are indicative of a pattern, are they due to similar or potentially related faults? Is there a correlation between them?
2- Are newly identified bugs potentially related to previously discovered bugs? Is a pattern emerging?

When a suspected cluster is identified the testing team works closely with the development team to find the probable root cause of the problem.

Clusters can be observed in a variety of ways. Here are four examples of "distance metrics" in bug space:

Functionality:	several bugs related to same functionality are discovered
Reliability:	different functions fail in similar way
Efficiency:	several operations are inefficiently using system resources in a similar manner
Time:	several content sources are out of sync

Prioritize based on business impact

Once a bug cluster is identified it must be prioritized. Just like you would prioritize an individual bug, a cluster of bugs must be prioritized. We assigned clusters one of four priorities:

P1 - Fix immediately, next build
P2 - Fix before commercial release
P3 - Fix in some future commercial release
P4 - Do not fix

Certainly the decision to correct the defect had a lot to do with the severity and business importance of the associated cluster. If for example a memory leak is discovered on a server application which causes an occasional failure which results in a very short term delay in service, but without any loss of data, then it may be prudent not to correct it. If however the same memory leak is aggravated by increased concurrent use of the site then it may critical to fix it before going live.

By working on bugs in a collection we can make decisions more efficiently.

Practical results

The techniques developed lead to health communication and team work between developers and testers. Developers and testers enjoyed working together to try and identify probable root causes of sets of seemingly related bugs.

Developers were much more aware of the role of testing in the process. This is especially beneficial as one of the teams goals is also to improve the Personal Software Process (PSP) of individual team members. Empirical knowledge about the processes and people in place can help since weaker points are sometimes inferred from past projects.

Management and team members all bought-in to approach. Combining exploratory testing with detective work makes sense especially when time is tight and we are always working with incomplete data. People were in sync.

We found that we were able to identify an additional 30% more software problems with this approach. More specifically we measured for every 10 bugs reported the correction of 13 defects in the code base. It is impossible to say however whether the correction of all of these defects would have been necessary.

The method is flexible and can use several alternatives as feeding mechanism to the Bug Cluster Identification phase. The one we chose was Exploratory Testing but more detailed and analytical methods may be used. This will be explored in future work.

Our three projects were released on time with minimal field reported defects. All critical problems were corrected during the final integration phase and hundreds of thousands of users are live using this software as we write this article.

We plan to continue using this technique, improving our understanding of the model and generalizing it as we continue. The effectiveness and efficiency, as well as the limits of the method will be further explored as well. This model is used in a commercial and very turbulent e-Commerce environment. It will evolve. All indications are positive and it is presently being used in several projects under development.

Some Web References

www.satisfice.com

James Bach's web site, exploratory and risk based testing

www.testing.com

Brian Marick's web site, articles about exploratory testing

www.amibug.com

Robert Sabourin's web site, various presentations

www.mvm.com - www.myvirtualmodel.com

My Virtual Model community Web site

www.stickyminds.com

Much relevant content.