Understanding Bohr-Mandel bugs through ODC Triggers and a case study with empirical estimations of their field proportion

Ram Chillarege

Chillarege Inc.

Raleigh, North Carolina, USA, www.chillarege.com

Abstract

This paper uses ODC Triggers as a means to estimate the Bohr-Mandel bug proportions from a software product in production. Specifically, the contributions are:

- *A conceptual articulation of how ODC Triggers can differentiate between Bohr and Mandel bugs*
- *A grouping of triggers to estimate the proportion of Bohr-Mandel bugs in the field.*
- *A case study that estimates Mandelbug proportions to range ~20%-40% which is comparable to the JPL-NASA empirical study.*
- *A measure of the distribution of Mandelbugs across components, impact groups and time.*
- *Creates a discussion, and raises questions in greater depth on the Bohr-Mandel definitions, implications, features and manifestations.*

I. Introduction

The subject of Bohrbugs, Mandelbugs (and/or Heisenbugs) is of interest to a few communities in software engineering. While these ideas have been around for a while, its importance grows as more software products ship with rejuvenation methods and tools. Also, large software service providers of cloud computing infrastructure need greater sophistication in the application of fault-tolerant methods at higher levels of abstraction. Thus, our need for a greater understanding of the underlying fault and failure phenomenons is vital.

There are several articles on this topic, starting from the famous article by Jim Gray [1] to several that further define the categories [2, 3]. There are articles on the technique of software rejuvenation [4] and methods that help estimate schedules towards higher availability [5]. The area also has a series of software engineering workshops that focus on these related topics.

In this article, we take a careful look at what might explain the phenomenon of Mandelbugs with the help of Orthogonal Defect Classification (ODC) Triggers. While some of the articles on the topic of Bohr-Mandel bugs mention ODC as a concept they do not get into the details of what it is and how to apply it.

In this paper we briefly explain what ODC Triggers [6] are and then discuss how this applies to the Bohr-Mandel bug theory. We also attempt to resolve some of the glossed over details in the Bohr-Mandel theory where there is a legitimate confusion between the fault and the failure. This arises, since one is defined in terms of the other: namely, the fault is ascribed a category based on its failure-mode.

The applicability of the ideas formulated in the paper is illustrated by a case study that helps put it in perspective of other studies. Specifically, we have chosen to estimate the measure that was also estimated by another empirical study on software defects from JPL-NASA space missions. This way, we are able to benchmark these estimates with each other, albeit the methods and the target system are different. Regardless, having comparable empirical data from different sources enhances our understanding of the subject.

The paper is organized thus: We begin by a discussion of ODC Triggers, and then do a quick review of a couple ODC attributes that we will use. This leads to the discussion on Bohrbugs and Mandelbugs to view them from the perspective of ODC. Next we discuss the approximation and estimation of Bohr/Mandel bugs using Triggers. Following this, we use our case study data to make measurements and compare them to the empirical study on JPL-NASA space mission data to draw parallels and insights. Several conclusions based on the data are discussed in context.

II. ODC Triggers

The Trigger is just one of the measurements in the Orthogonal Defect Measurement (ODC) system [8], but is the vital element for this discussion. The trigger is best explained by using the illustration in Figure 1 that shows two states – the fault and the failure. The failure is an event where the system's response is not according to specification. The cause of the failure is attributed to the fault [9]. This applies for both hardware and software. The important element to note is that fault can be either a code implementation that is incorrect, or a correct implementation of an incorrect design. Regardless, that fault can result in a failure where the response of the system is flawed. A further discussion on this topic is pursued in

[12]. The failure that occurs is a manifestation of that fault under the circumstances of its environment.

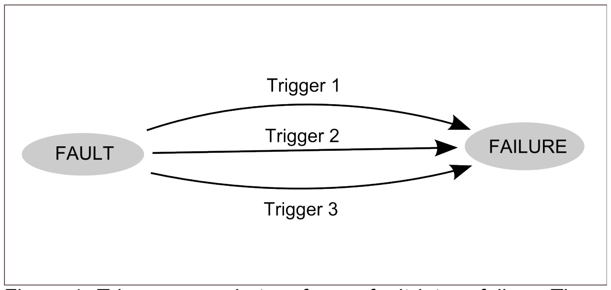


Figure 1: Triggers are what surface a fault into a failure. There can be more than one trigger that can activate a fault, and thereby resulting in different failure modes.

In software, faults are primarily latent and they all do not cause failures. Failures occur only when the fault is activated. Triggers activate a fault. A trigger is that mechanism which surfaces the fault as a failure. Thus triggers are not a cause but it is only a force of the software environment that exercises a fault condition.

The fault when activated, creates an incorrect state, corrupts the output, or slows operations, or some such symptom which amounts to the failure. In theory, one could define the incorrect system state as a failure. However, in practice, one must note that not all failures get noticed. That is, not every failure puts the system into a state where the manifestation of the failure is obvious. For instance, when data is corrupted it is rarely noticed immediately. However if the system crashes then that event is far more observable and therefore may gain the attention of the operator or a user.

This gets into finer semantics of the failure operation, and its associated latencies. While that discussion is related to our topic, it's just outside the scope of our current discussion.

For our purposes we need to crystallize what the trigger is and how that relates a fault to the failure. Essentially, the set of triggers in ODC V5.11 is a broad enough set that it allows us to span the variety of different forces at play from requirements to field usage. Thus, ODC Triggers become a measurement through their distribution, on the verification and validation process. The details of this are discussed in several ODC papers [10].

Same Fault, different Trigger

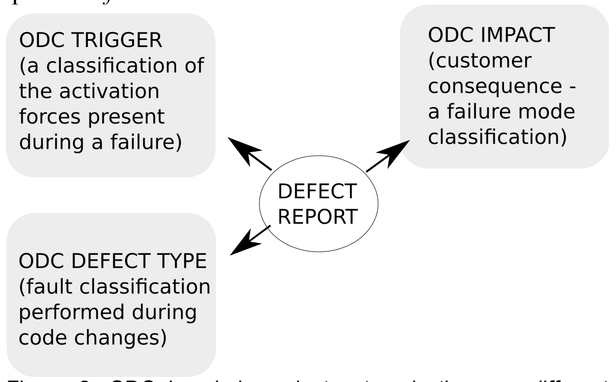
What may not be so obvious but important to note is that the same fault can be triggered by more than one trigger. Thus, triggers are not unique in activating a fault. For instance, most faults can be detected by inspection alone. However, inspection effectiveness is often poor and therefore formal testing is needed to screen for those escaped faults. Similarly, the testing process is not perfect and many faults escape and are left to be detected during field usage.

The triggers that surface faults during customer usage are often different from those within the verification and validation processes. This is primarily due to the environmental conditions of the customer or production environment being different from that of the test environment. In practice, this difference can be so larger that there have been movements in the software industry to encouraging early release of buggy software so that customers can do the debugging for the developer. It works within limits - so long as customers are gracious enough to deal with the onslaught of failures.

These tradeoffs in the development and testing strategy (and business strategy) are tied to ODC Triggers. Thus their importance from an analytical perspective. This is what makes the management of the testing and verification process inherently complex and exciting for researchers.

The ODC categories

ODC data captures several attributes from the defect, the Trigger being only one of them. Figure 2 shows a representation of a few of the other attributes. These are drawn around the circle that represent a defect. ODC extracts the semantic information from the defect and turns it into a measurement on the process. The different attributes in ODC measure different aspects of the development process. Using the terms from the earlier section, the Defect Type is a *fault* classification whereas the Impact is a *failure* classification.


ODC TRIGGER (a classification of the activation forces present during a failure)
ODC IMPACT (customer consequence - a failure mode classification)
DEFECT REPORT
ODC DEFECT TYPE (fault classification performed during code changes)
Figure 2: ODC has independent categorizations on different aspects of a defect incident. In our papers we use the Triggers and Impacts, but not the Defect Type. However it is shown here for completeness to distinguish between fault classifications and failure classifications.

The Defect Type and Trigger categories have been carefully designed in ODC so that they provide a measurement on the development process and the testing process in a software development process. The ODC literature discusses these aspects in considerable detail [10].

We touch upon the ODC Impact in this paper, and therefore a few words on this are in order. The Impact category is constructed so that it measures the nature of the pain that is forced on the customer. The impact categories also separate them into functional and non-functional consequences.

Usually a separate severity measure on an ordinal scale of 1-4 is used to quantify the degree of the impact.

With respect to this paper – be it a Bohrbug or a Mandelbug the impact categories do not change, since the Bohr and Mandel are failure phenomenons which all need to be characterized by the Impact categories.

The two attributes on the left side of Figure 2 are associated with cause. The "cause" in the process sense is associated with development and test. The attributes on the right side are associated with effect. The "effect" is with respect to the customer. Thus, each defect creates a link between aspects that are associated with cause to those with effect.

Collectively the ODC attributes capture an enormous amount of information from individual defects. The most common use is in-process and post process quantitative analysis, a replacement of the classical root cause analysis process, measurements of test effectiveness, risk reduction, process diagnosis, etc. Separately, ODC data is also a valuable research tool, such as the work being reported in this paper.

III. Bohrbugs and Mandelbugs

The late Jim Gray, coined the term Bohrbug while the term Heisenbug has been in usage longer. These terms have since been refined and various subtleties added to the equation. In a nutshell, the Bohr bug is one that had a much more deterministic type of behavior in its failure mode. Whereas, the Heisenbug, or more appropriately the Mandelbug demonstrates a far more non-deterministic failure behavior [2].

But, herein also lies the Achilles heel of this form of categorization. Is it a fault characterization of a failure characterization?

Hidden in this formulation is the assumption that certain types of faults necessarily have a failure mode of one kind or the another. This is hard to swallow for those of us who see a lot of defect data and understand ODC, defect containment, triggers and the statistics of escape analysis and its reasons. But, like many a dicey issue, there are arguments on both sides, and a certain amount of truth in either position.

Linus's law, "given enough eyeballs, all bugs are shallow" creates an argument that defeats the proposition that faults fall distinctly into two classes – Bohr/Mandel, since, by this argument, the ODC Passive Triggers can uncover a bug – thus turning a Mandelbug into a Bohrbug. The counter example is enough to muddy the waters that bugs fall into distinct classes, be it Bohr or Mandel.

The subtlety is in the nature of the "escape process" in the software development life-cycle. What this amounts to is that if one of the sub-processes (such as inspection, or unit test) fail to catch the bug, the probability of this bug being caught downstream rapidly decreases per unit effort. The downstream testing or verification process create newer triggers, whose effectiveness changes by the fault types. We are usually able to predict the responses to recommend the most suitable methods, given a particular condition. While that may not be the focus in our discussion here, it does reveal the underlying dynamics, of fault types, triggers, and the probability of surfacing a failure.

What this implies for the Bohr/Mandel theory is that there will be a finite number of bugs whose likelihood of being observed distinctly as a Bohrbug (or a Mandelbug) increases as one heads towards the later stages of development, and consequently in field usage and production too.

Triggers resolve the Issue

There is a good discussion in [2] on the classification of software faults as to what might render a fault as a Mandel bug. The discussion is along the lines of additional factors that influence the behavior of a fault be it hardware or the software. We are in agreement on this part of the discussion, in the subjective arguments set forth. However, we would like to take it a step further to crisp up the factors, and allow for more ready quantification.

What distinguishes the Mandelbug phenomenon from the Bohrbug phenomenon is the Trigger. This is the glue that helps resolve the confusion on how or whether the Bohr-Mandel bug theory is tied to the fault or the failure.

To be crystal clear – we are not saying that it is purely a Trigger classification. What we are saying is that if a defect escapes to a later stage of development such that it gets revealed only by the more difficult class of Triggers, then it behaves as a Mandelbug. Now, that same fault could have been captured at a different (may be earlier) phase of the development process and then it could have been labeled a Bohrbug.

This is a very different presentation from one that portrays them as faults that are mutually exclusive and either belong to the set of Bohrbugs or Mandelbugs.

Mapping Triggers to spot Bohr-Mandel bugs

So what are the statistics and characteristics of these Bohr/Mandel bugs? This is an important question to ask since, the more we know of them, the better we can set a mouse trap – be it Rejuvenation or some alternative means of achieving high availability and fault-tolerance.

Given these discussions we have an opportunity to use the Triggers for the purposes of gaining a little deeper understanding of the Bohr-Mandel bug phenomenon.

Measurement of triggers is far simpler and more readily available than the measurement of the environment or the combination of a fault in a particular environment. This distinction is vital because one cannot tell a Bohr bug or a Mandel bug from the looks of the Defect Type alone. For instance an assignment fault, which is essentially an update of the variable to a static or dynamic value may be the fault but it may not surface because of the way it is used. The usage pattern of the same variable under different circumstances could render widely different failure modes.

In Table 1 we divide the triggers into two groups. The first group approximates the Bohr bug phenomena and the second group approximates the Mandel bug phenomena.

Table 1. Grouping of Triggers to identify and estimate Bohrbugs and Mandelbugs.

Bohrbug Triggers	**Mandelbug Triggers**
Lateral Compatibility	Concurrency
Backward Compatibility	Rare Situations
Coverage	Side Effects
Variation	Interaction
Design Conformance	Sequencing
Logic Flow	Recovery/Exception
Normal Mode	Workload/Stress
Complex Path	
Hardware Configuration	
Software Configuration	

The reason for this choice is that there are certain triggers that represents a complex operation when this is experienced in the field. For instance Workload and Stress are triggers that are quite commonplace in the field and often create situations that are not easily replicable. This fits the overall description of a Mandel bug. Similarly the Sequencing and Interaction triggers represent forces where there are several tasks running in parallel and the timing conditions they create makes certain bugs surface. This choice is also conditioned by our case study that is restricted to field data. Thus, not all the ODC Triggers need to be present in Table 1.

IV. Case Study

The choice of this case study was motivated by the JPL-NASA empirical study on space missions software [11]. In this study, the proportion of Bohr-Mandel bugs is estimated, by individually categorizing the defect reports.

In software engineering empirical work there is tremendous value in having related studies since it enlarges our understanding of these large complex systems that almost behave like organisms with a life of their own. That said, these systems are different, and the methods used are also different. However, any parallels recognized helps us gain insights and a world view that is larger than the individual study.

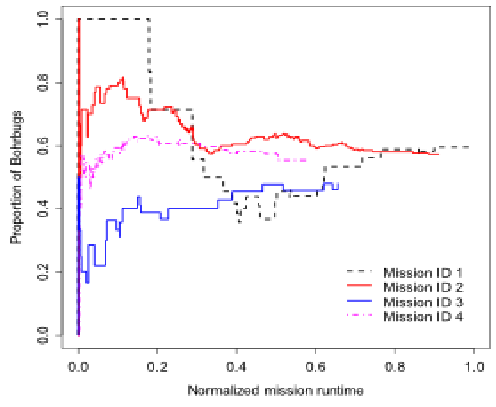
We picked a case study project that is reasonably close to the JPL-NASA study, albeit from a different industry. The software comes from a high-reliability product that is fairly widely distributed and belongs to a class where there is significant focus by the development team to produce a good product that will scale well.

The JPL-NASA study used about ~600 defects from several missions. Therefore, we too chose to look at only field operations or the equivalent of production runs, and chose a multi-year sample that also happens to be in the range of ~600 defects.

Extract from the JPL-NASA Study

The JPL-NASA study categorized ~600 of the defects from the anomaly reports that were related to the flight software of 18 missions. These are thus categorized as Bohrbugs and Mandelbugs. The latter is differentiated as being aging related or not. Finally, each of these faults are thus labeled, and the analysis studies several aspects of its manifestation reporting the fault type proportions along a normalized mission duration. For reference, we include one of the figures from this study to help us compare some high level takeaways.

Figure [3] shows the proportion of Bohrbugs for four of the missions. The x-axis in this graph shows a normalized mission time and the y-axis displays an estimate of the proportion of faults that are Bohrbugs for a particular mission. This is updated with every additional fault event and stabilizes as the mission progresses. For a high level takeaway, we read that the proportion is around 60-40, and the numbers across the different missions vary around this number, but not widely.



Proportion of Bohrbugs for Mission IDs 1 to 4
M. Grottke, A. Nikora & K.S. Trivedi Paper, on
18 JPL/NASA space mission. DSN, 2010.

Figure 3: We compare some of our results to these from the NASA study on Bohrbugs and Mandelbugs [11].

Data in Case study

The number of defect events in this project runs into thousands especially when examined over multiple years. However, not all these events are of interest for us in this particular study. Therefore, we restricted the data to only those from a field deployment or production environment. This allows us to make it a closer comparison point to the JPL-NASA study.

Some other differences are that there is no concept of a mission in this product. These are everyday products that have a long deployment cycle in the field that runs in to several years. However, since they are large products they also have several components, which allow us to study the variability by component.

For this study, we have thus chosen field data from multiple components across a few years. Each of these defects are categorized by ODC. We do not need all the attributes of ODC for this study and therefore restricted our analysis to what is meaningful.

Figure [4] shows us the overall Bohr-Mandel proportions along with an Impact distribution. The Bohr bugs and Mandel bugs are determined by the Trigger assignment as shown in Table 1. Since this is field data, and not development data we believe that the estimates of Mandelbugs is about right, and the errors if any are on the higher side. This is because the Triggers associated with the Bohrbugs are most definitely Bohrbugs, whereas there may be a few of those in the set of Mandelbugs that could possibly be rendered as a Bohrbug, but unlikely the other way around.

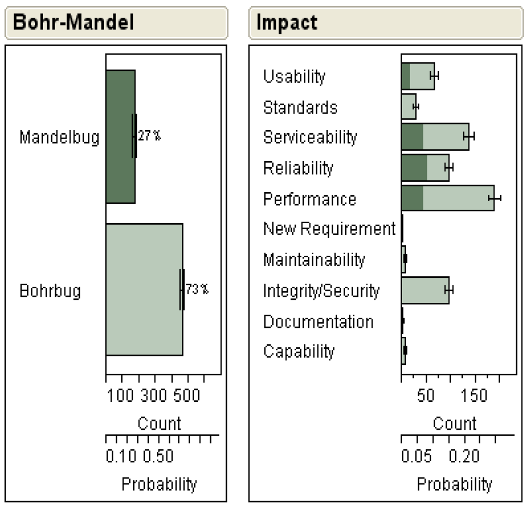
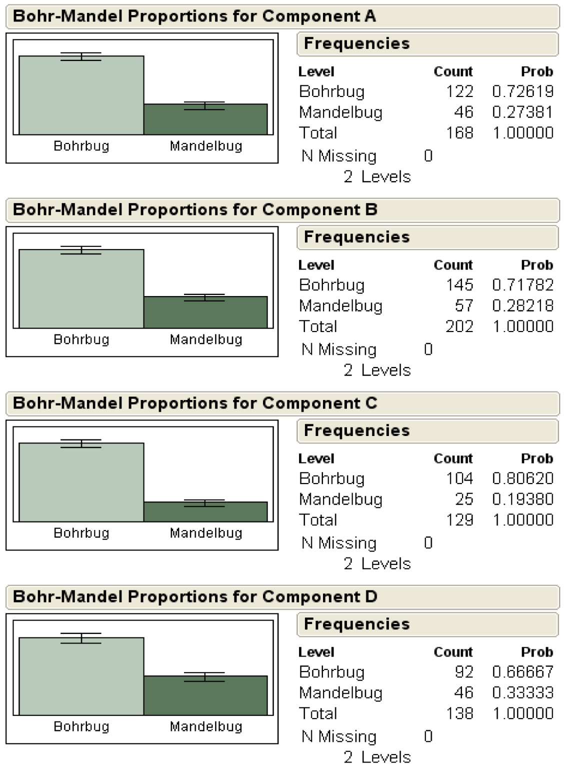


Figure 4: The Bohr-Mandel proportions are computed for the entire product and all the data over multiple years. The Impact distribution is highlighted so as to illustrate the relative Bohr-Mandel contributions to each impact category.

A few observations on these data are in order, before we slice this data further.

- The population of Mandelbugs are fewer than the Bohrbugs. This is similar to the JPL-NASA study.
- While the error-bounds (indicated on the charts) on these measure are fairly tight, we think that this ~70/30 split is just the nature of this product and environment, as compared to the ~60/40 split in the case of the JPL-NASA data
- The Impact distribution alongside the Bohr-Mandel distribution is revealing in that there seem to be a representation of Mandelbugs in each of the major impact groups, with the notable exception of the Integrity/Security class of impacts.
- We recommend that the lack of Mandelbugs in the Integrity/Security group should not be over interpreted. There are many reasons for this, and unless we conduct a full ODC analysis the conclusions can be misleading. It is far better to note that of the 5 large impact groups, 4 of them had Mandelbugs in them, and leave it at that.

We now look at the Bohr-Mandel proportions for each of the components in the system. We have chosen to study four major components where each has a significant contribution to the whole. There were a few other components that we ignored since the numbers in those groups are far too small to be meaningful.



Bohr-Mandel Proportions for Component A

Frequencies

Level	Count	Prob
Bohrbug	122	0.72619
Mandelbug	46	0.27381
Total	168	1.00000

N Missing 0
2 Levels

Bohr-Mandel Proportions for Component B

Frequencies

Level	Count	Prob
Bohrbug	145	0.71782
Mandelbug	57	0.28218
Total	202	1.00000

N Missing 0
2 Levels

Bohr-Mandel Proportions for Component C

Frequencies

Level	Count	Prob
Bohrbug	104	0.80620
Mandelbug	25	0.19380
Total	129	1.00000

N Missing 0
2 Levels

Bohr-Mandel Proportions for Component D

Frequencies

Level	Count	Prob
Bohrbug	92	0.66667
Mandelbug	46	0.33333
Total	138	1.00000

N Missing 0
2 Levels

Figure 5: The Bohr-Mandel bug proportions are shown for four components, where each has a sizable contribution to the whole, and the counts are meaningful to compare.

A few observations from these data follow:

- The Mandelbug proportions are uniformly smaller than the Bohrbugs in each of the four components.

- The confidence interval for the Component A estimate of 0.27 Mandelbugs runs from 0.21 to 0.34 at an Alpha of 0.95. That said, Mandelbug proportions point estimates range from 0.19 to 0.33 across the four components.
- At a high level, it is reasonable to state that the Mandelbug range we see in this study is in the ~20%--40% window. In our case, this translates to a ~60%--80% spread for the Bohrbugs.
- Comparing this to the JPL-NASA study (see Table 1 in their article) the Bohrbug range there is from ~50%--80% . Bohrbugs are a better comparison since we do not perform the aging related sub classification of Mandelbugs.

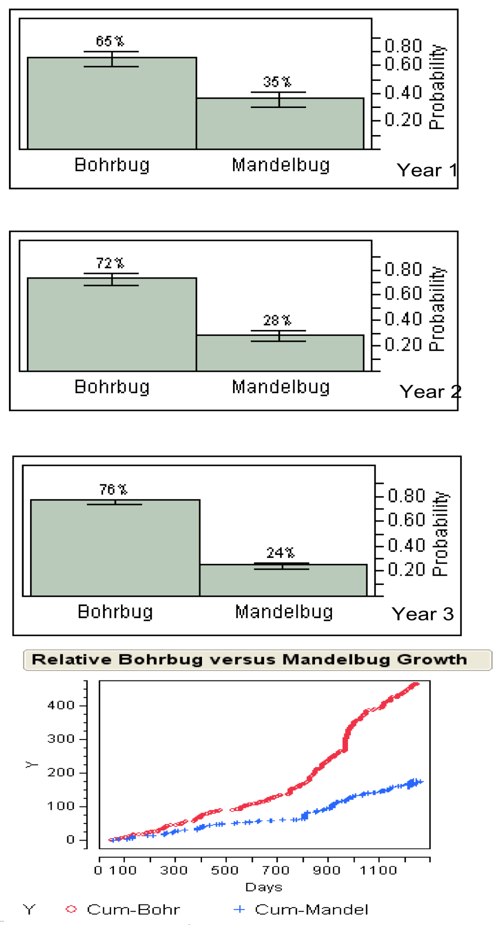


Figure 6: Proportions of Bohr-Mandel bugs across a time line. Data is partitioned by year showing changes in the first three years. The lowest chart shows the cumulative growth of Bohr-Mandel bugs against time, illustrating some of the dynamics at play.

Next, we look at a different view of these data – along the time dimension. Our case study data spans a little over three years and we can study the effect of the time domain, or aging of the product in the field.

Figure [6] shows us the Bohr-Mandel bug proportions for three consecutive years. Below the three charts, we also have plotted the cumulative growth of these individual groups, which shows us their relative growths.

A few observation from these data are:

- There is a steady decrease in the proportion of Mandelbugs. What starts off as 35% of Mandelbugs in year 1, drops to 28% and 24% in year 2 and year 3.
- The growth curve below the three charts illustrates this phenomenon better. There seems to be an uptick in the Bohrbugs in year three, whereas the Mandelbug contributions do not rise as fast. This can be due to a variety of phenomena connected with increase deployment, sales as is the case with many products. But, what does stand out is that while there is some corresponding uptick in the Mandelbugs, they seem to be leveling out.
- This tells us that it is likely that Mandelbug population does not disappear in time. While the numbers and proportions may fall, the trend line seems to suggest that they will remain, probably closer to the lower bound of the estimates.

Summary

This paper has made the following contributions:

1. We discuss the Bohr-Mandel theory of bugs and argue that a greater insight and differentiation is possible through the use of ODC Triggers.
2. We conduct a case study to measure the proportion of Bohr-Mandel bugs using ODC Triggers.
3. Our case study results are compared to the JPL-NASA study on Bohr-Mandel bugs.
4. Our estimates of Mandelbug proportions ranges (~20-40% range) are very comparable to those found in the JPL-NASA study.
5. We find that the Mandelbug presence is uniform across multiple impact groups and components of the product.
6. We found that in our case study, as the product ages in the field, the proportion of Mandel bugs seemed to drop. While this finding cannot be generalized, it does alert us to watch this phenomenon in other case studies. However, given the numbers it is clear that methods to alleviate the impact of Mandelbugs early in the lifetime of a product will be of significant value.

References

[1] J. Gray, Why do computers stop and what can be done about it?, in *Proc. Fifth Symposium on Reliability in Distributed Systems*, 1986.

[2] M. Grottke and K.S. Trivedi, A Classification of Software Faults, in *Proc. Sixteenth International IEEE Symposium on Software Reliability Engineering* , 2005

[3] wikepedia.com   Search on Bohrbugs http://en.wikipedia.org/wiki/Unusual_software_bug

[4] Y.Huang, C. Kintala, N. Kolettis, and N. Fulton, Software Rejuvenation: Analysis, Module and Appplication, in *Intl. Symp. On Fault-Tolerant Computing*, 1995.

[5] T. Dohi, K. Goseva-Popstojanova, and K.S. Trivedi, Estimating software rejuvenation schedule in high assurance systems. *The Computer Journal*, vol. 44, no.6. 2001

[6] R. Chillarege, et. al., Orthogonal Defect Classification – A Concept for in-Process Measurements, *IEEE Transactions on Software Engineering*, Vol 18, No. 11, Nov 1992.

[7] M. Butcher, H. Munro and T. Kratschmer, Improving software testing via ODC: three case studies.(Orthogonal Defect Classification), *IBM Systems Journal* March 01, 2002

[8] R. Chillarege and K.A. Bassin, Software Triggers as a Function of Time - ODC on Field Faults, *Dependable Computing and Fault-Tolerant Systems Vol. 10,* IEEE Computer Society. Also, DCCA-5, Urbana Champaign, 1995

[9] A. Avizienis, J.C. Laprie, B. Randell, and C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, *IEEE Transactions on Dependable and Secure Computing*, 2004

[10] ODC articles: http://www.chillarege.com/odc

[11] M. Grottke, A. Nikora and K.S. Trivedi, An Empirical Investigation of Fault Types in Space Mission Systems Software, in *Proc. Dependable Systems and Networks,* 2010, Chicago, IL.

[12] R. Chillarege, What is Software Failure?, *Commentary in IEEE Transactions on Reliability,* Vol. 45, No.3, September 1996