Do Bugs Speak?

MESUT DURUKAL, Rapyuta Robotics

In this article, I will share how bug management and monitoring processes can be conducted in the best way to maximize the gain collected from the bugs by referring to my experiences. What type of aspects, dimensions or parameters hold valuable information, is investigated by mostly regarding the last project I was involved in.

1. INTRODUCTION

Defect counts have traditionally been an indicator of product quality and a criterion for a release decision. But this is a very limited way to deal with bugs. By embracing digital defect management systems and the opportunity to learn from the bugs found on a product over time you can gain insights and use these to improve your internal processes as well as the product itself.

This paper is my personal story as a QA professional of how I learned to appreciate bugs rather than see them as something negative. In other words, how I started to listen to the bugs and act on what they were telling me.

2. BACKGROUND

This story took place during 2 years when I worked as a technical lead for a test automation team in Siemens. I represented the Turkey location in a global organization whose responsibility was to ensure quality for MindSphere, a cloud-based open IoT platform. I managed 18 people organizing the tasks to automate E2E cases, implementing API and UI automation cases and integrating them into CI/CD pipelines to be able to perform continuous testing activities.

Siemens launched MindSphere, which is a cloud based open IoT Platform, to develop solutions for industry automation like collecting data from processes and detecting improvement points in the industry life cycle. *"MindSphere is a leading industrial IoT as a service solution developed by Siemens for applications in the context of the Internet of Things. MindSphere stores operational data and makes it accessible through digital applications to allow industrial customers to make decisions based on valuable factual information."* [1]

The project was driven with cutting edge technologies like XaaS: Everything as a Service (Both SaaS and PaaS), Cloud and EDGE Computing, Microservices architecture and IoT devices. It is developed by a globally distributed organization with teams in Germany, Hungary, Turkey, Austria, India and China.

In such a system with lots of subsystems, modules and units developed in a decentralized structure with new development approaches it is no surprise that we encountered a number of issues—bugs, change requests, weaknesses and risks were reported by the development, QA and operations teams and from our customers on the production environments. The bugs were typically coming from:

- Integration of subsystems or units
- Performance and scaling issues
- Configuration or deployment issues
- Wrong usage or testing
- Nature of design or architecture
- Reliability and security issues
- Missing documentation

All bugs were recorded and managed in our issue-tracking tool, Jira.

Author's address: Mesut Durukal, Koto-ku Tokyo Japan; email: durukalmesut@gmail.com

3. WHY I LEARNED BUG LANGUAGE

On the one hand, finding a lot of bugs feels annoying and disappointing. How could it not? Everyone in the team puts in an effort to develop the increment within a limited time frame. But as bugs are reported, development activities get slowed down by bug fixes. Moreover, the team may feel unable to produce working software.

However, bug-free software is inherently impossible. Some bugs were always expected. For sure, there is a cost associated with fixing them. But there is the other side of the medallion as well, the bright side: we can take advantage of the reported bugs that we have recorded.

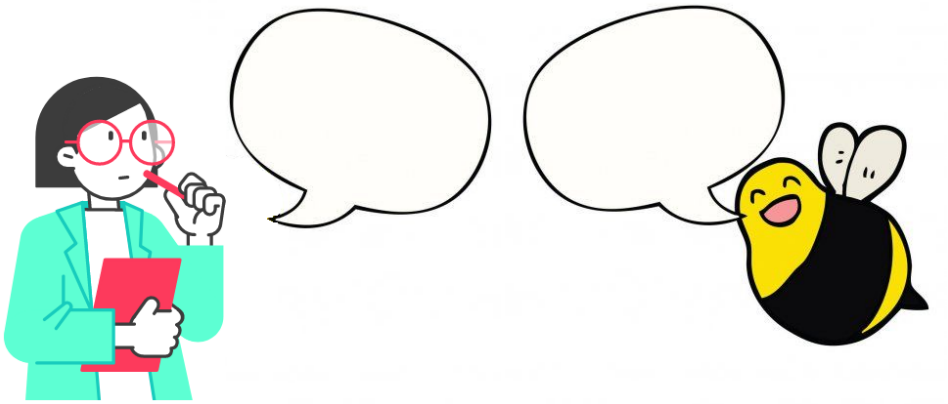


Figure 1. **Speaking with Bugs**

The issues are eventually resolved, sooner or later. But in addition to removing the bugs from the product, what can be done to make use of them? Can we turn them into beneficial feedback channels? This is possible by listening to what the bugs are telling us. It is always like this; we can't learn from someone or something unless we listen to them.

This is the moment when I decided to speak with bugs and collect information from them by listening to them. Because I noticed that there can be lots of valuable information that can be gathered over bugs. This is the ***"value of the bugs"***, apart from the defect cost.

4. INSIGHTS FROM BUG TALKS

Don't you believe that bugs speak? Let me share some conversations that I had with them.

4.1 Story of an old bug

Once a very aged bug said Hi to me. I met her after a tough root cause analysis. I sat together with developers to understand the root cause of an issue. After lots of findings, at the end of the way, she was sitting over there and looking at us:

"Why didn't you come to me in the first place? I am here for a long while. If you called the most experienced guy who knows most about this issue, you would immediately see me."

At that time, this beautiful bug was not considered as a major issue, but in time she got aged and had descendant bugs. The value she taught me is the importance of analyzing all the potential effects of bugs instead of forgetting them even if they are not major issues at that time.

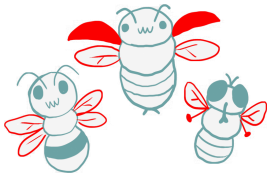
4.2 The Sweet Escape

Another story is from a bug who somehow succeeded in escaping to production. He found me in the evening of a long deployment day.

In this particular deployment, fortunately all tests passed and we did not have to fix issues and repeat the activities. At least, we thought that was the case! But what is it!?! After the deployment was done to the customer site, the system was not up and running.

How was it possible? When I visited the customer site, the bug welcomed me and explained everything in detail! He said that "Even though all the features work properly under certain configurations, some other conditions can appear under different configurations."

4.3 Best Friends Band!



In a sprint, when visiting bugs, they introduced themselves one by one. After knowing the first one and learning that he is coming from "security land", the second told that he is a friend of the first and also coming from the same place.

At the end, I realized most of them are somehow connected to each other and arise from the same reason. They were almost like: ***"Hey, we are all best friends and hang out together. We all disappear or appear again under the same conditions."***

5. ADVANTAGES OF BUG CONVERSATIONS

Bug analysis is very important for QA teams, and especially for QA managers. When the bugs are investigated in detail, various insights can be collected concerning not only the quality of the product, but also the whole system, pipelines, processes and requirements.

Bug conversations can go deep into root causes and help understanding the true state of our product. Individual bugs tell us a lot but even more value do we get from bug trends and bug collections. A comprehensive analysis of all created bugs can provide precious insights about the product. For instance; if we notice that a bunch of bugs heap together on a feature, we can conclude that the feature should be investigated and cured. We can learn from the bugs what kind of requirements are expected from the customers – if they are making mistakes then our system has usability issues and/or gaps in the functionality. Or we can make some observations about the severity or assignee of similar bugs. Therefore, there are potential patterns to be discovered.

In my project, we constructed lots of dashboards and monitoring graphs to trace the progress of resolution of bugs and see other trends. We had a separate dashboard to focus on escaped bugs. Finally, using Machine Learning (ML), we achieved to collect insights in a much more efficient and faster way. After the investigation of all bug categories and clusters, we have reached numerous observations like:

- After having some issues, we have found out that the root cause was already reported a very long time ago. But why isn't it resolved yet? That, no one knows.
- Similar issues were arising again and again since the root cause is not always analyzed.
- While the percentage of escaped bugs per the number of bugs found in testing activities was high for some features, it was very low for some other features.
- The trends for different bug categories were directly proportional. They were increasing or decreasing together along the sprints.
- The number of bugs stemming from the misunderstanding of the usage of the system or feature was going down whenever we generate user manuals.

6. DESIGNING A BUG ANALYSIS PROCESS

After deciding the questions that we wanted to answer, and what valuable information we therefore needed to collect about our bugs, the next step was to clarify the data collection channels.

I first collected bugs created by end users both from the real environment and testing environment. To achieve our goals, we needed a very clear model for the contents and a process where the contents are correct and relevant. We maintain monitoring tools and platforms with the development and SRE teams, and there were already prepared dashboards and charts to categorize and cluster bugs in our issue tracking tools. For the consistency of the study, by manual inspection I made sure the information in the issue tracking tools was up to date and ready to be analyzed. As the whole team, we were trying to put all the comments, updates, changes and other relevant information into issues so that we would collect as much valuable information as we could.

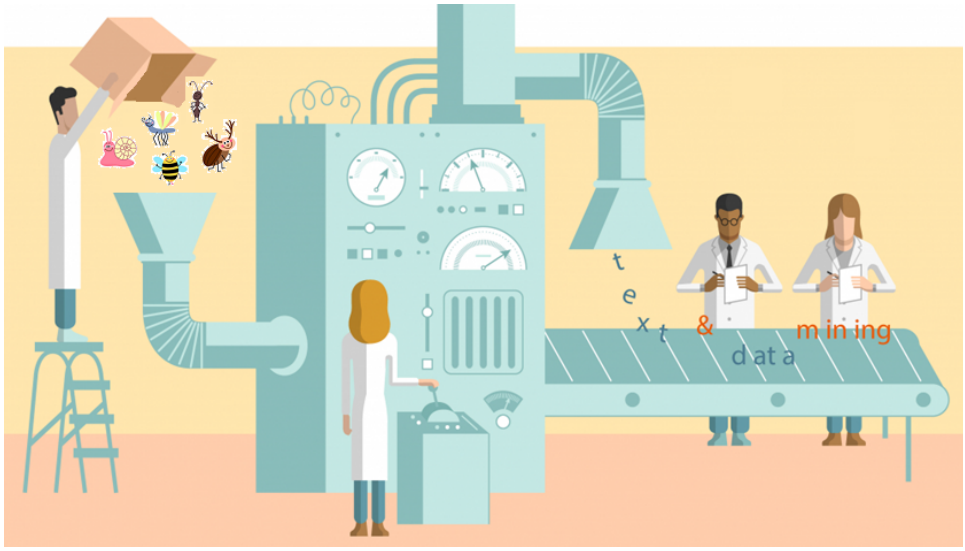


Figure 2. **Setup for analyzing bugs**

The activities I performed step by step were:

- ✓ Building a bug life cycle
  After realizing that there were inefficiencies in the workflow, we customized the state transition flow and added some extra states such as reopening a bug or final customization. Figure 3 shows our bug lifecycle.

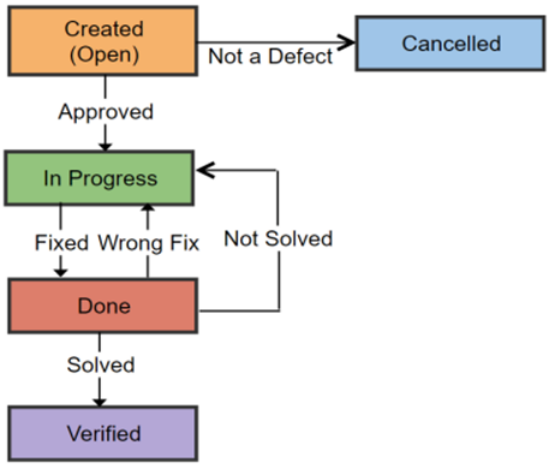


Figure 3. **Bug State Transition Diagram**

- ✓ Automated queries
  To reduce the manual effort for the monitoring activity of bugs, I implemented code to automatically perform queries from APIs of the issue tracking system. In this way, bug resolution duration, bug ages, open/closed/resolved counts were pulled into the analysis.
- ✓ Basic bug distribution dashboard
  To concentrate on the bugs having highest priority first, I constructed a dashboard including bug distribution across severity levels. In each sprint, we monitored the distribution of bugs with high priority among others.
- ✓ More advanced bug distribution dashboard
  We added different distribution of bugs across various parameters such as component or testing type. For instance, if most of the bugs heap together on 'uploading feature', we would check the health of deployment of the relevant component. Or in a sprint, if most of the bugs are related to documentation testing, we could get insights about the process of documentation or whether yaml generation is broken.
- ✓ Escaped bugs
  We have made a special study to concentrate on Escaped bugs and reduce their number.
- ✓ Machine Learning
  I adapted Machine Learning into bug management processes, this is explained in section 7.

Below are examples of the various dashboards, some of them monitoring the sprint-based numbers and some others monitoring the cumulative numbers. Figure 4 depicts distribution of bugs across different parameters, Figure 5 shows investigation of ages of bugs and Figure 6 is showing created versus resolved bugs.

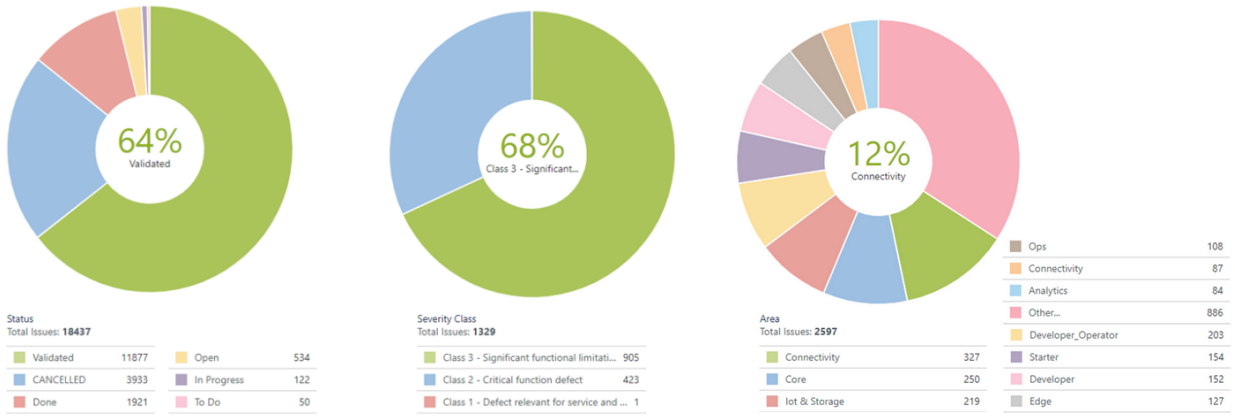


Figure 4. **Status, Severity, Related Module Distributions**

Severity Class	0-30 Days	31-60 days	>60 Days	Total ages
1	0 issues	0 issues	1 issue	1 issue
2	19 issues	2 issues	23 issues	44 issues
3	105 issues	47 issues	204 issues	356 issues
4	46 issues	24 issues	164 issues	234 issues
5 → 7	21 issues	4 issues	49 issues	74 issues
[EMPTY]				0 issues
Total	191 issues	77 issues	441 issues	709 issues

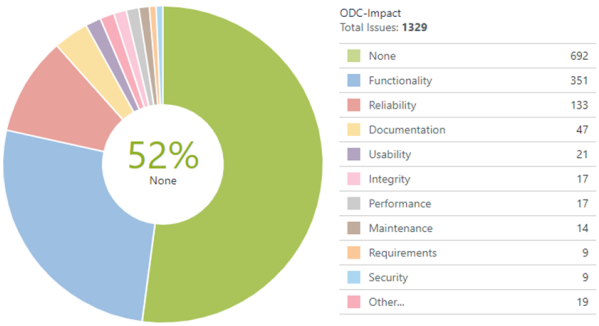


Figure 5. **Type of the Bug and Age vs Severity**

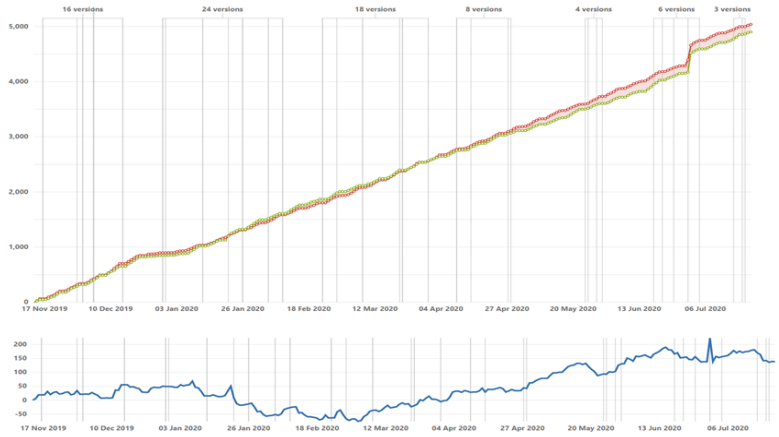


Figure 6. **Created vs Resolved**

7. MACHINE LEARNING ADAPTATION

I have applied ML to predict the severity levels of bugs by reading their descriptions.

7.1 Why Use Machine Learning?

Bug triage is important, because if you give a lower severity value to a bug than it deserves, it means most probably the fix will be delayed since it is not properly prioritized. On the contrary, a severity which is higher than needed blocks the other tasks.

When setting the severity values of bugs, we do not check the other bugs created by other people. This results in inconsistencies. One can think a bug should be with S2, where another person can think the same bug should have S4.

Instead, if the assignment is done by machines, it can be much more consistent. Detecting the hidden patterns and valuable information can be achieved much more easily with Machine Learning algorithms [3].
What is difficult for human beings:

- Dealing with ***large data*** sets. When creating a new bug, we do not read all the previously created bugs to understand the approach for the triage of the bugs.
- There is no time to read 1000 bugs! It means there will be ***inconsistencies***. Since everyone does not read other bugs, triage depends on the person.
- Mining valuable information from large data manually is almost ***impossible***.

What can be done easily by Machines?

- Process large data in a very ***short time***.
- Which means the triage can be done ***consistently***?
- Since machines do not get tired as humans, they are ***not biased***.

7.2 How is Machine Learning applied?

As for ML algorithms, first training is performed and then according to the learnings, the outcomes of the upcoming samples are predicted; in our project I collected training data. I exported the bugs from our issue tracking system and mainly concentrated on the description of the bugs. The purpose was to construct a model by exposing the relationship between descriptions of the bugs and their severity levels.

Thus, once the model is constructed, whenever a new bug is raised it would be possible to predict the correct severity level according to the description. Since we are trying to generate a mathematical model from the descriptions of the bugs, written in texts; we somehow had to convert strings to numerical vectors. For this purpose, we used the Bag of Words [2] technique. In Figure 7, it can be seen that Bag of Words method provides a solution to vectorize texts and converts them into numerical arrays.

```
Training texts: ["This is a good cat", "This is a bad day"]

=> vocabulary: [this, cat, day, is, good, a, bad]

New text: "This day is a good day"   -->   [1, 0, 2, 1, 1, 1, 0]
```

Figure 7. **Bag-of-Words Model**

In addition to classification, clustering can be done to group similar bugs to manage them in a more efficient way. Figure 8 shows detected clusters.

```
Top terms per clusters for: (2, 2)

Cluster 0:
 error message
 returns 504
 wrong error
 gateway error
 create agent
 mindsphere gateway
 service returns
 event management
 visual explorer
 different queries

Cluster 1:
 internal server
 server error
 500 internal
 returns 500
 management returns
 endpoint returns
 responded 500
 request responded
 events returns
 asset management

Cluster 2:
 timeseries data
 fleet manager
 yaml file
 wrong response
 stream deployment
 time series
 returns 400
 bad request
 response code
 information yaml
```

Figure 8. **Bug Clusters**

7.3 What is gained?

Bug triage is done much more consistently by means of ML. Since a model is generated after observing all data, for the future predictions, it means that the decision is taken by considering all past samples. That is not the case for human predictions. People generally set a severity according to their understanding without considering others.

Additional insights are gained after similar bugs are collected in clusters. For instance, it was observed that there were some bugs clustered around text 'gateway error'. These clusters gave us chance to group bugs to evaluate them all together. Additionally, we detected the possible weaknesses in the product by analyzing cluster to see if some bugs heap on specific features.

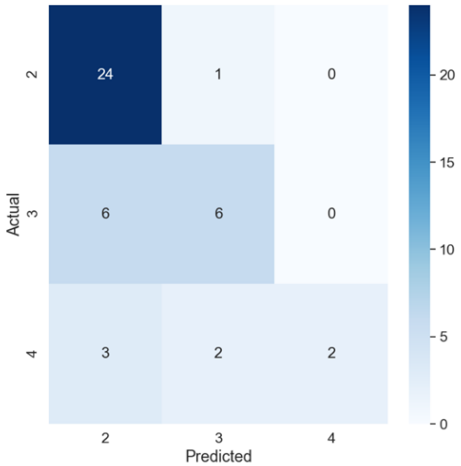
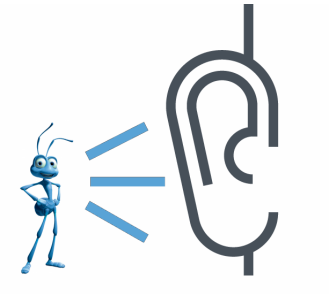


Figure 9. **Confusion Matrix for Classification of Bugs across Severity Levels**

In Figure 9, the confusion matrix is shown [4]. Actual labels are the decisions made by people and predicted labels are the results of our algorithm. Since the reference values are personal decisions, evaluating the success of the algorithm is a bit tricky. But at least it was seen that the decisions made by the algorithm were close to the decisions made by people. The algorithm was aligning the human decisions with the general trend, that is why there are some differences. Therefore, it can be said that the benefit is the consistency and even if we need a final human confirmation, the algorithm can be used for the initial triage.

8. CONCLUSION

8.1 Summary



After interesting information was whispered into my ears by some bugs, collecting ideas by categorizing them gave me the idea to perform more categorization and classification.

With a proper bug management process, we managed to:

- ✓ reduce the gap between opened vs resolved bugs
- ✓ avoid very aged bugs
- ✓ reduce the number of escaped bugs
- ✓ increase test coverage by root cause analysis of bugs from the production environment

When the process was mature and we were able to gather lots of various insights from different classifications, the final step was to adapt ML, since the classification and clustering can be done in the best way by Machine Learning algorithms.

8.2 How we utilized the observations and outputs

From each story told by the bugs and observations made from charts or trends, we have taken some action items to cover the improve rooms. We continuously tracked the progress and observed that various processes have improved in time.

One of the first things we have done is, after realizing that some bugs were open for a long time, which was making customers upset, we started to track bugs per their ages. After constructing 'bug distribution across age' dashboards, the CCB (Change Control Board) started to monitor them each sprint and did not let bugs to be open for a long time.

What we learnt from 'bug distribution across age' tables are, especially certain types of bugs were open for a long while. For instance, security bugs were not taken into consideration with high priority, maybe since they were not visible to customers. They are critical though. The observation was the underestimation of certain types of bugs. Similarly, by checking bugs across priority tables, we were understanding the distribution of bugs across each level. We have tried to motivate people to concentrate on the most prior cases first and not to allow critical bugs to go production.

The gap between opened vs resolved bugs was clearly giving an idea whether we had to halt new features or not. When we notice that the gap is getting wider, some sprints were assigned to only bug resolution. (POs did not define new features in such a situation) Thus, another value gathered was the sprint planning decisions.

Finally, what did not work is, we tried to use the number of found bugs as a performance criterion. But in time, it turned out to be a 'bad metric'. People tended to increase their "fixed bug" numbers by focusing on the cosmetics bugs. We gave up tracking the number of bugs per person, concluding that it was not a perfect decision to take this metrics as a performance criterion.

9. ACKNOWLEDGEMENTS

I am honored to work with my coach Lise Hvatum. This paper would not have come together without her keen insights, questions, and edits: *Thanks, Lise, I couldn't have done it without you!*

REFERENCES

[1] https://en.wikipedia.org/wiki/MindSphere
[2] https://en.wikipedia.org/wiki/Bag-of-words_model
[3] https://www.infoq.com/news/2021/03/machine-learning-testing/
[4] https://en.wikipedia.org/wiki/Confusion_matrix
[*] Cartoons: https://www.clipartmax.com