Personal Reading: Cost of Defects

Mats Stijlaart - University of Amsterdam, The Netherlands

March 06, 2015

What Is Cost Of Defect About?

The cost of defect is a research topic that focuses on the classification of the costs of defects. In these studies research try to find the relation between different aspects related to fixing the defect. For example:

- Who introduced the bug and the one that fixes the bug. Whether it is the same person or a differently classified person.
- In what stage of the process the bug was resolved. This relates to the time between the introduction of the bug and the moment it is resolved [4].

In this reading assignment the main focus in on the moment defects are fixed, on which Boehm dedicated a part his research.

The Fuzz Around The Cost Of Defects

In his 1976 paper *Software Engineering* [4] and similarly in his 1981 paper *Software Engineering Economics* [5], Boehm shows that there is a significant difference in the costs of defects in relation to the phase in which the defect is fixed. The papers show that the costs are in average 60 times greater for solving a bug in the operations phase compared to the requirements phase (this is my read of the graph). Boehm presents with a logarithmic distributed y-axis, which is nearly impossible to read accurate. He lacks to provide the data points in a convenient table. Boehm states in his first of the two earlier stated papers that "it pays off to invest effort in finding requirements errors early and correcting them in, say, 1 man-hour rather than waiting to find the error during operations and having to spend 100 man-hours correcting it" [4, p. 4], which is a poorly chosen example because the graph says differently (this is one of the things that can be abstracted from the data, using the median). The cost to fix a bug in the requirements phase is lies clearly above between 0.2 and 0.5, and the cost to fix a defect in the operations phase lies clearly in between 10 and 20. The 1 to 100 ratio that Boehm states in his example, could only be made on outliers, which may indicate a poor conclusion or a exaggeration of the results.

The notion of Cost of Defects is treated in *The Leprechauns of Software Engineering* [6] written by Laurent Bossavit. In this book, the writer shows that the statement originally made by Boehm lacks the required data and is proven to be inadequate in different studies [6, pp. 98,102]. The writer also presents that Boehm in collaboration with another writer attenuate Boehm's original ratio of 100:1 to 5:1 [2, p. 426].

Although, Boehm original statement is put into perspective, the damage is already done. Different papers have adopted the statement, without the validation of Boehm's statement, and sometimes inflect the results. For example, by adding currencies to and/or removing the notion of a median from the graph [6].

One of the most interesting 'findings' (I'll elaborate later on) of Bossavit, is that Kent Beck or Beck's teacher in college misinterpreted Boehm's statement [6, p. 97] [1, p. 21]. Resulting Kent to write about "Cost of Change" in his book *Extreme Programming Explained: Embrace Change* [1] and stating that the form of the Boehm's curve is no longer valid. Also this work, just as Boehm's work in 1981, was very influential [1] [5].

However, what Bossavit does not show is that there the "Cost of Change" from Beck is connected to the "Cost of Defects" from Boehm [6]. There are no citations from Beck to Boehm's work and the possibility that "Cost of Change" was a figment of Beck's teacher is not considered [1]. The point made, I agree with Bossavit that the research of Boehm and Beck seem to lack the required data, but I find the provided information by Beck insubstantial to conclude the ideas relate.

What We Actually Know?

To conclude shortly: We know nothing. The provided 'important' research (read: cited a lot), lacks proper empirical research as presented in the previous section. But can we actually create a 'Holy Scale' that allows us to read the cost of defects from and holds for every team? I believe there is not, and maybe we should not try to pursue this. The reason why I do not believe this is possible, is because every project has a different environment. Fixing a bug in an Erlang program when in operations may be much easier than fixing a bug in a Coboal program when it is in operations [2], just because it may take less time to schedule a release.

What Is Its Importance?

As shown, we do not actually know a lot of how much defects are going to cost us. But if we would know and we can eliminate the cost, it allows us to better reason about the benefits of code reviews, test-driven development (TDD), fault detection and other practices. For example with code reviews it is measurable how much bug report differ when applied, it is measurable how much time is spend on code reviews, but the unknown factor would be the actual costs of bugs.

As it is no science, I would advice companies to start measuring on project level. For example, iterative processes lend themselves to detect effects of the exertion of TDD on the number of defects. An iterative approach like Scrum creates more continues flow of releases, which allows tracking of when defects i.e. the x-axis of Boehm's curve would have a more accurate distribution when it was applied on Scrum.

And Then We Know

Assume a team can measure what the costs of a defect may be and they have encountered a defect, the question remains what we will do next. Beck makes an argument in his book where he explains that you cannot fix every bug (although the known 'cost of defects') [1, p. 22]. To build on this argument, let say:

- Some customers are waiting for a feature;
- If this feature is not delivered, they may move to the competition;
- And we have detected a defect in one of the obscure error paths in our system;
- This error will only occur on one day in a leap year.

[1] In 2003, Boehm responded to Beck and changing his own 'theory' into "Cost of Change" just as Beck stated [3], but this still does not show that Beck's original statement was based on Boehm's work.

[2] Erlang's virtual machine BEAM has a hot module reloading system, which allows an application to stay online during a redeploy of the software.

Will we fix this defect? For many companies the answer is "No, not right away. Customers are waiting for this feature for this feature". This is of course logical. A lot of money may be at stake for a problem that can be fixed later. The point made: Knowing that bugs exist, may not always be an argument to fix them although you may know that it might cost you a bit more in the future.

A Bit of Common Sense

If we resign by the fact that we cannot know the exact ratio between the costs of defects when defects are fixed in an early stage of the project or in an operations phase. With common sense we can reason why it may be more expensive to fix software when it runs in production.

- You may lose customers when they encounter bugs.
- Deploying software may take some planning, when a bug has to be fixed in production a new version of the software may be required and thus the overhead of putting your software in production
- After a while programmers lose context. Not only about the domain, for example an financial module that was implemented a few months ago, but the also the understanding of the bug may decrease over time.

Another interesting point we can reason about is the adjustment Boehm made over the years (from 100:1 to 5:1). Although Bossavit has presented that the studies were insufficient and in the previous paragraph I have stated that we cannot know, I do believe there is something to the decrease in that ratio. For example, when looking at the second point of the previous analysis, deploying software in general got easier and more reliable due to tooling as Puppet[3], Chef[4] and Ansible[5]. The normal analysis (understanding the bug and finding it) may not have changed, but the overhead of putting the software live, that decreased. Also there is a movement to put software in the cloud, for some software vendors this means that maintenance and upgrades can be performed over night at a continuous pace. This allows them to bring the new software to the end user much easier.

Conclusion

To conclude: We still do not know if there is an actual 'curve' on the cost of defects such as Boehm described. We do not know if there exists a formula that we can use that will give us approximation of the costs. The only thing I believe, is that companies should start measuring on project-basis to get an insight on what is happening. We should not generalize on this matter. I do believe that the cost of fixing defects in an operations phase is higher than during a development or requirements phase, but I will not enounce a ratio because I do not believe it is possible.

[3] https://puppetlabs.com/
[4] https://www.chef.io/chef/
[5] https://www.ansible.com/

References

[1] Kent Beck. *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[2] Barry Boehm and Victor R Basili. "Software defect reduction top 10 list". In: *Foundations of empirical software engineering: the legacy of Victor R. Basili* 426 (2005).

[3] Barry Boehm and Richard Turner. *Balancing Agility and Discipline: A Guide for the Perplexed, Portable Documents*. Addison-Wesley Professional, 2003.

[4] Barry W Boehm. "Software Engineering". In: (1976).

[5] Barry W Boehm et al. *Software engineering economics*. Vol. 197. Prentice-hall Englewood Cliffs (NJ), 1981.

[6] Laurent Bossavit. *Leprechauns of Software Engineering: How Folk Law Turns Into Fact [Electronic Resource]*. Lean Publishing, 2013.