Personal Reading: Cost of Defects

Mats Stijlaart - University of Amsterdam, The Netherlands

March 6, 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 researchers 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.
- 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 is on the moment defects are fixed, to which Boehm dedicated a part of his research.

The Fuzz Around The Cost Of Defects

In his 1976 paper 'Software Engineering' [4] and similarly in his 1981 book *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 paper and the book show that the costs are in average 60 times greater for solving a bug in the operations phase compared to solving them in the requirements phase (this is my read of the graph). Boehm presents his argument with a graph containing a logarithmic distributed y-axis, which is nearly impossible to read accurately. He lacks to provide the data points in a convenient table. In his 1976 paper, Boehm gives an example to support his statement: "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 lies clearly above between 0.2 and 0.5, and the cost to fix a defect in the operations phase lies between 10 and 20. The 1 to 100 ratio that Boehm states in his example, could only be made on outliers, which indicate a poor conclusion or an 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 In this book, the writer shows that the statement originally made by Boehm lacks the required data and is proven to be inadequate in different studies [6, pp. 98,102]. Bossavit also points out that Boehm attenuates his original ratio of 100:1 to 5:1 in a paper he wrote in collaboration with Basili [2, p. 426].

Although, Boehm's 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].

In 2000 Kent Beck wrote the book *Extreme Programming Explained: Embrace Change* [1]. Beck writes about an event he encountered in college related to the cost of change [1, p. 21]. Resulting Beck 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. According to Bossavit, Kent Beck or Beck's teacher in college

misinterpreted Boehm's statement [6, p. 97]. However, Bossavit assumes that 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 Bossavit does not consider at all the possibility that the "Cost of Change" might only have been a figment of Beck's teacher [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 Bossavit and in Beck's work insubstantial to conclude that the ideas relate.

What Do We Actually Know?

To conclude shortly: We know nothing. The provided 'important' literature - which has that status mainly due to it being cited a lot by others - 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 properly for every context and situation? I believe we cannot, and maybe we should not try to pursue this. For example, every project has a different environment. Fixing a bug in an Erlang program when it is in the operation phase may be much easier than fixing a bug in a Cobol program when it is in the operation phase [2], just because it may take less time to schedule a release.

What Is Its Importance?

As shown, so far no suitable method to determine the cost of defects. But if we had such a method and we could determine the cost and we can eliminate the cost, it would allow 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 we may not able to drop a scientific model on it, 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 since there is a continuous interval.

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 to do next. Beck argues convincingly that even if you know for sure that fixing defects in an operations phases has higher costs, you may not want to fix the bug right away [1, p. 22]. To build on this argument, let's assume the following situation:

- Some customers are waiting for a feature;
- If this feature is not delivered, they may decide to move to the competition;

[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.

- The company has detected a defect in one of the obscure error paths in their system.
- This error will only occur on one day in a leap year.

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 a logical choice. A lot of money may be at stake for a problem that can be fixed later. In conclusion: 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

Let's resign by the fact that we cannot know the exact ratio between the costs of defects in different phases of the project. With common sense we can still 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 a new version of the software may be required, which leads to the overhead of putting your software in production
- After a while programmers lose context. Not only the understanding of the domain, for example a financial module that was implemented a few months ago, but also the understanding of the bug may decrease over time.

Another interesting point we can reason about is the adjustment on the cost of defects 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 determine the cost of defects, I do believe there is something to the decrease in that ratio. For example, when looking at the second point of the above bullet list, 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 has decreased in the past years. Also, nowadays one can make use of 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: *IEEE Trans Computers* (1976), pp. 1226–1241.

[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.