Unit test coverage targets are harmful for software. Never request them!

Zizka, Jan (NSN - CZ/Prague)

Exported on 04/01/2024

Table of Contents

1 Who is Jan Žižka

This is for those who do not know me. I take it as very important to tell about my background so that this page would be taken as just a quick though. What I'm sharing is based on work I have behind me. I don't have any official titles, I'm not architect or chief engineer nor manager. I'm just an engineer with passion in software and many lines of code behind me.

My name is @Jan Zizka (Nokia) I'm **principal technical leader** in Nokia Mobile Networks (MN) in Edge Cloud Platform (ECP) in Radio Cloud Platform (RCP) leading Linux OS, FOSS, development practices area.

I have **35 years of programming** experience and I'm **24 years at Nokia**.

I was honored to become **Distinguished Member of Technical Staff** (DMTS) at Nokia Bell Labs.

I have received **Nokia Business Excellence Award**.

With Commit Reviewer Apprenticeship[1] project created and lead by me we won the MN China LEAP Star Award[2].

I have created and am leading the Unit Testing Apprenticeship[3] project to ramp up unit testing competencies.

Since spring 2023 I am also **researcher** at Lab of Software Architectures and Information Systems[4] (lasaris) at Masaryk University[5] in Brno, Czech Republic.

I have given talk on DevConf.cz[6] developer conference, hope more I'll do in future.

I have triggered organization of RCP Software Developer's Conference 2022[7] and I have given there talk on Advanced Rust[8].

I have been working on hardware design, on low level software such as drivers, on OS code and services, on platform code and also on application code. I have implemented many tools for software developers and I have implemented number of CI pipelines and SCM scripts. Outside Nokia I have ported binutils and gcc for DSP56800 processor and I have been in automation and robotics before coming to Nokia. I know how most of our software is built and how it looks, **it is my hacker heart which urges me to explore what and how works, so I go and keep exploring and learning**. Both inside Nokia but also outside.

---

1 https://confluence.ext.net.nokia.com/display/RCP/Commit+Reviewer+Apprenticeship
2 https://nokia.sharepoint.com/sites/MN_China_LEAP_Star_Award/SitePages/Home.aspx
3 https://confluence.ext.net.nokia.com/display/RCP/Unit+Testing+Apprenticeship
4 https://lasaris.fi.muni.cz/
5 https://www.muni.cz/en
6 https://devconfcz2020a.sched.com/
7 https://confluence.ext.net.nokia.com/display/RCP/RCP+Software+Developer%27s+Conference+2022
8 https://confluence.ext.net.nokia.com/display/RCP/SW+Developer+Conference+sharing+material

2 Jan's testimony on code coverage targets

OK today is 09 Mar 2023 and once again the topic of reporting unit test coverage and setting limit for minimum value of test coverage pops up.

This time with COOP enforced DoD (DEP Tailed DoD criteria[9]).

I'm programming for 35 years and 24 years from that in Nokia, and yes I'm still writing code. **This particular topic pops up every about 5 years** when software quality stalls and there is need to provide data which would show how good our software is or when some manager needs to measure just because they don't have any measure of software quality. Well, **code coverage is very bad measure, and even worse target**. There are much better metrics for code quality to use.

By measuring test coverage number and seeing that it is let say 60% and setting target to 80%, the requestor of this requirement can show "See we have improved our software, our code coverage raised from 60% to 80%", means our software got better.

The dear **programmers in meanwhile were busy to reach that magic number, and sure they always will, but .... What an unfortunate complete mystification**. The software and product quality doesn't get better, it actually gets worse ... so well then probably 80% was not enough, lets set the target to 90%. No surprise the software development get suddenly much slower and the quality of software drops even more.

"How that is possible?" the requestor of requirement thinks. Well soon after the requirement is dropped and all returns to where it should be, whoever requested the threshold moves to other position all is forgotten. Until 5 year later, again someone new comes and start with "Hah lets fix our software quality issues I don't see we have threshold for code coverage, lets set it to 70%!" ... and this repeated, believe me, every 5 years or so. I have lived through this already 5 times. The sad thing is that **this is something what whole industry knows for ages, but we still fall each time into same trap and it cost us huge amount of effort and worse software**.

I have decided to write this **page to gather all the evidence of my software development experience and all the evidence which anyone can read** if they would try to search for it. My hope is to **finally stop this so that we don't waste effort and time on something which is fundamentally wrong**.

And don't take me wrong. **The code coverage is essential tool and very useful one, but it must be utilized in correct way.** So should we collect and measure code coverage? Yes definitely. Should we set target values and use them for gating? Definitely no! **Should this be used as metrics by managers for seeing software quality. By all means no!**

Read what I have collected bellow, click on the links and read the complete texts. See the authors all so much respected in software industry.

Hope that after reading all what I have gathered you will stop requesting any unit test coverage target numbers, and instead, you will request meaningful metrics and you will request and use the code coverage analysis along with the code reviews as tool to indeed make your code better. **BTW if you use TDD you will never ever encounter this topic.**

Just to make it clear. **Yes I have done the same mistake also**. I have tried to use code coverage as gate in past. And yes it turned all bad as described by everyone else in industry. I have bee also on projects some 12 years ago in IPA platform where we used coverage for gating. It was nightmare for test quality where you had **target set for 80% and with new code changes you have just fell to 79.9%**. What we did? Well we have invented a useless test ... or we moved to bar. Same happened some 5 years ago in RCP and for some crazy

9 https://confluence.ext.net.nokia.com/display/CloudDeployment/DEP+Tailed+DoD+criteria

reasons our pipelines still fail when CQG[10] limit is not hit. What we do when you fall by 0.1%, yes you guessed right, we drop the bar or again implement some useless test.

During our Unit Testing Apprenticeship[11] project we were discussing also this topic, interesting when you go to source code and check the coverage you will almost always see that software with high number have completely purely written unit tests. **The unit tests are very much useless and even worse they become maintenance nightmare**. You may wish to visit our Common review UT issues[12] for some examples.

10 http://cqg.dynamic.nsn-net.net/ci3/dashboard/
11 https://confluence.ext.net.nokia.com/display/RCP/Unit+Testing+Apprenticeship
12 https://rcp.gitlabe2-pages.ext.net.nokia.com/rcp-docs/development/common_issue.html?highlight=coverage#ut-issues

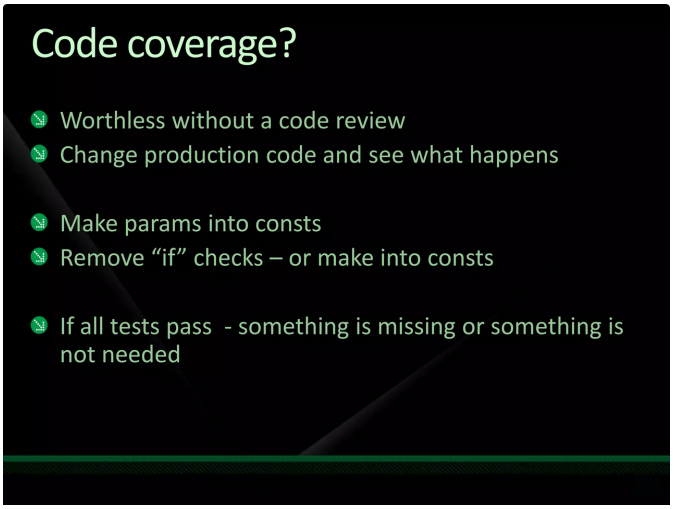
3 Roy Osherove, "The Art of Unit Testing. Third Edition[13]"

> *Sometimes people actively write tests to achieve some imagined "test coverage" goal set by management.* ***Those tests usually serve no real value except to get management off people's backs*** *so they can do real work. Show your manager the following text:*
>
> ***Managers:*** *Code coverage shouldn't ever be a goal on its own. It doesn't mean "code quality". In fact, it often* ***causes your developers to write meaningless tests that will cost even more time to maintain****. Instead, measure "Escaped Bugs" ,"Time to Fix" and well as other metrics that I provide in chapter 11.*

I also suggest reading the The Art of Unit Testing, Second Edition[14] where is even more discussion and experience with setting the code coverage targets.

13 https://www.manning.com/books/the-art-of-unit-testing-third-edition
14 https://www.manning.com/books/the-art-of-unit-testing-second-edition

4 Roy Osherove on Unit Testing Good Practices and Horrible Mistakes[15]



15 https://www.slideshare.net/royosherove/roy-osherove-on-unit-testing-good-practices-and-horrible-mistakes/

5 Unit testing best practices with .NET Core and .NET Standard[16]

> ***Code coverage***
>
> *A high code coverage percentage is often associated with a higher quality of code. However, the measurement itself can't determine the quality of code.* ***Setting an overly ambitious code coverage percentage goal can be counterproductive****. Imagine a complex project with thousands of conditional branches, and imagine that you set a goal of 95% code coverage. Currently the project maintains 90% code coverage. The amount of time it takes to account for all of the edge cases in the remaining 5% could be a massive undertaking, and the value proposition quickly diminishes.*
>
A high code coverage percentage isn't an indicator of success, nor does it imply high code quality*. It just represents the amount of code that is covered by unit tests. For more information, see* unit testing code coverage[17].

16 https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices
17 https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-code-coverage

6 Ben Morris, "Don’t use test coverage as a target[18]"

> *Measuring test code coverage can be a useful technique for finding where the gaps are in your automated tests.* ***The problem is that coverage is often used as a target, where it can be misleading, encourage the wrong behaviours and even distort development***.

18 https://www.ben-morris.com/dont-use-test-coverage-as-a-target/

7 Anthony Sciamanna, "Code Coverage Complications[19]"

> ***Coverage is a Result, Not a Goal***
>
> *The crux of the problem is that high code coverage results from quality-first development practices. Naturally, organizations with low code coverage are missing these practices. Instead of dedicating time and effort to improving development practices, a* ***focus on increasing code coverage misconstrues the result as the goal****.*
>
> ***Goodhart's Law***
>
> *Coverage goals are often a textbook example of Goodhart's Law. The law states that "****when a measure becomes a target, it ceases to be a good measure****." Absent an outcome-based goal, pursuing higher code coverage will only increase suffering in the codebase.*
>
> 

19 https://www.industriallogic.com/blog/code-coverage-complications/

8 Mark Seemann, "Code coverage is a useless target measure[20]"

> *Let's make it clear, then: don't set goals for code coverage.*
>
> *You may think that it could make your code base better, but* ***asking developers to reach a certain code coverage goal will only make your code worse****.*
>
> *I'll show you some examples...*

20 https://blog.ploeh.dk/2015/11/16/code-coverage-is-a-useless-target-measure/

9 Chris Cooney, "Is Test Coverage a Good Metric for Test or Code Quality?[21]"

> ***Code Coverage is a Dumb Metric***
>
> *It's measurements are reliable when you're tracking how much of your code is ran by your tests but it tells you absolutely nothing of the value of those tests. Visualising it on its own is useless because it has no reliable, predictive relationship with the quality of the code or the tests.* ***Quality gates that prevent releases for minor decreases in code coverage are invites for crazy shit ...***

21 https://hackernoon.com/is-test-coverage-a-good-metric-for-test-or-code-quality-92fef332c871

10 Henri Benoit, "Code Coverage: A misleading metric[22]"

> *In a new project without tons of legacy see code I do expect to have a high code coverage. The problem with code coverage comes from the fact that your management may not only expect high coverage but actually require a certain level of code coverage. And* ***whenever a certain level of coverage becomes a target, developers try to reach it****. The thing is that it* ***encourages them to bend over backwards by writing poor tests just to meet the goal****. It leads to writing tests for their own sake.*

22 https://benohead.com/blog/2014/08/19/code-coverage-a-misleading-metric/

11 Umer Mansoon, "The Problem With Code Coverage Metrics[23]"

> *Some organizations and managers make high level of code coverage mandatory for their teams. "**90% code coverage and no less!**", says the manager. It becomes another metric in management's arsenal to assess code quality and (god forbid) team's performance. It's a **big mistake** to interpret and use code coverage in this way...*
>
> *Let me get this straight again: **code coverage is a valuable metric. But when management turns it into a goal and becomes fixated on it to measure quality and performance, things rapidly disintegrate**....*
>
> ***Managers should expect their team to have high coverage but they must not turn it into a target.** Metrics do not make good code. The ultimate goal is to have fewer bugs that escape into the production.*

23 https://codeahoy.com/2016/04/16/do-not-misuse-code-coverage/

12 Carlos Arguelles, Marko Ivanković, and Adam Bender, "Code Coverage Best Practices[24]"

> *More important than the percentage of lines covered is human judgment over the actual lines of code (and behaviors) that aren't being covered.*

24 https://testing.googleblog.com/2020/08/code-coverage-best-practices.html

13 Vard Antinyan, Jesper Derehag, Anna Sandberg, Miroslaw Staron, "Mythical Unit Test Coverage[25]", Ericsson conclusions

> *It is a continuous struggle to understand how much a product should be tested before its delivery to the market. Ericsson, as a global software development company, decided to evaluate the adequacy of the unit-test-coverage criterion that it had employed for years as a guide for sufficiency of testing. Naturally, one can think that if increasing coverage decreases the number of defects significantly, then coverage can be considered a criterion for test sufficiency. To test this hypothesis in practice, we investigated the relationship of* ***unit-test-coverage measures and post-unit-test defects in a large commercial product of Ericsson. The results indicate that high unit-test coverage did not seem to be any tangible help in producing defect-free software***.

25 https://www.researchgate.net/publication/324959836_Mythical_Unit_Test_Coverage

14 Robert "Uncle Bob" Martin

> *Audience question to Uncle Bob: Which parts of the code do you unit test? **What about code coverage?** Uncle Bob: Well (chuckling).. **You test the parts of the code that you want to work**. – link*[26]
>
> *The only test coverage goal that makes any sense is 100%. It's an asymptotic goal. You'll likely never get there. But you should never stop trying. – link*[27]

26 https://devrant.com/rants/2311696/audience-question-to-uncle-bob-which-parts-of-the-code-do-you-unit-test-what-abo
27 https://twitter.com/unclebobmartin/status/1205922909350293505

15 Dan Lebrero, "The tragedy of 100% code coverage[28]"

> *The tragedy is that once a "good practice" becomes mainstream we seem to forget how it came to be, what its benefits are, and most importantly, what the cost of using it is.*
>
> *Instead, we just mechanically apply it without too much thought, which usually means that we end up with at best mediocre results,* ***losing most of the benefits but paying all (or even more) of the cost****. In my experience writing* good unit tests is hard work[29]*.*

28 https://dev.to/danlebrero/the-tragedy-of-100-code-coverage
29 http://danlebrero.com/2016/11/06/good-test-vs-bad-tests/

16 "Dave Thomas on Testing Past, Present, and Future[30]"

> ***Use experience and judgement to choose when not to write tests***
>
> *...I would guess my test coverage is maybe 10%. And I am very happy with that. Because I am changing this language daily. I'm ripping out how I do things, replacing it with something else. And if I was to have tests for that, I would not get any work done. I would be spending all of my time refactoring all of my tests...*

30 https://semaphoreci.com/blog/dave-thomas-testing-past-present-future

17 Martin Fowler, "TestCoverage[31]"

> *From time to time I hear people asking what value of test coverage (also called code coverage) they should aim for, or stating their coverage levels with pride. Such statements miss the point.* ***Test coverage is a useful tool for finding untested parts of a codebase. Test coverage is of little use as a numeric statement of how good your tests are.***

31 https://martinfowler.com/bliki/TestCoverage.html

18 Brian Marick, "How to Misuse Code Coverage[32]"

> ***I expect a high level of coverage. Sometimes managers require one. There's a subtle difference.***
>
> *Suppose a manager requires some level of coverage, perhaps 85%, as a "shipping gate". The product is not done - and you can't ship - until you have 85% coverage.*
>
> *The problem with this approach is that people optimize their performance according to how they're measured. You can get 85% coverage by looking at the coverage conditions,*
> *picking the ones that seem easiest to satisfy, writing quick tests for them, and iterating until done. That's faster than thinking of coverage conditions as clues pointing to*
> *weaknesses in the test design. It's especially faster because thinking about test design might lead to "redundant" tests that don't increase coverage at all. They only find bugs.*
>
> *So which are people going to do? Spend a larger amount of time writing better tests that rank the same as tests tailored for high coverage, or write the quicker tests?*
>
> ***If a part of your test suite is weak in a way that coverage can detect, it's likely also weak in a way coverage can't detect.***

32 http://www.exampler.com/testing-com/writings/coverage.pdf

19 References

- https://www.slideshare.net/royosherove/roy-osherove-on-unit-testing-good-practices-and-horrible-mistakes/8-Code_coverage_Worthless_without_a
- https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices
- https://www.ben-morris.com/dont-use-test-coverage-as-a-target/
- https://www.industriallogic.com/blog/code-coverage-complications/
- https://blog.ploeh.dk/2015/11/16/code-coverage-is-a-useless-target-measure/
- https://www.diffblue.com/Testing/webinars/coverage-when-is-enough-enough-stop-setting-unrealistic-unit-test-coverage-goals/
- http://ithare.com/too-much-unit-testing-is-detrimental-for-code-quality/
- https://hackernoon.com/is-test-coverage-a-good-metric-for-test-or-code-quality-92fef332c871
- https://benohead.com/blog/2014/08/19/code-coverage-a-misleading-metric/
- https://codeahoy.com/2016/04/16/do-not-misuse-code-coverage/
- https://testing.googleblog.com/2020/08/code-coverage-best-practices.html
- https://www.researchgate.net/publication/324959836_Mythical_Unit_Test_Coverage
- https://www.informit.com/articles/article.aspx?p=1621865&seqNum=5
- https://methodpoet.com/100-code-coverage/
-
  Sorry, the widget is not supported in this export.
  But you can reach it using the following URL:
  https://twitter.com/unclebobmartin/status/1205922909350293505?lang=cs
- https://devrant.com/rants/2311696/audience-question-to-uncle-bob-which-parts-of-the-code-do-you-unit-test-what-abo
- https://dev.to/danlebrero/the-tragedy-of-100-code-coverage
- https://www.slideshare.net/Kevlin/structure-and-interpretation-of-test-cases
- https://semaphoreci.com/blog/dave-thomas-testing-past-present-future