How to be an effective reviewer

Gilles Peskine

Original: Jan 2019; revised Sep 2020

First, get into the right mindset!

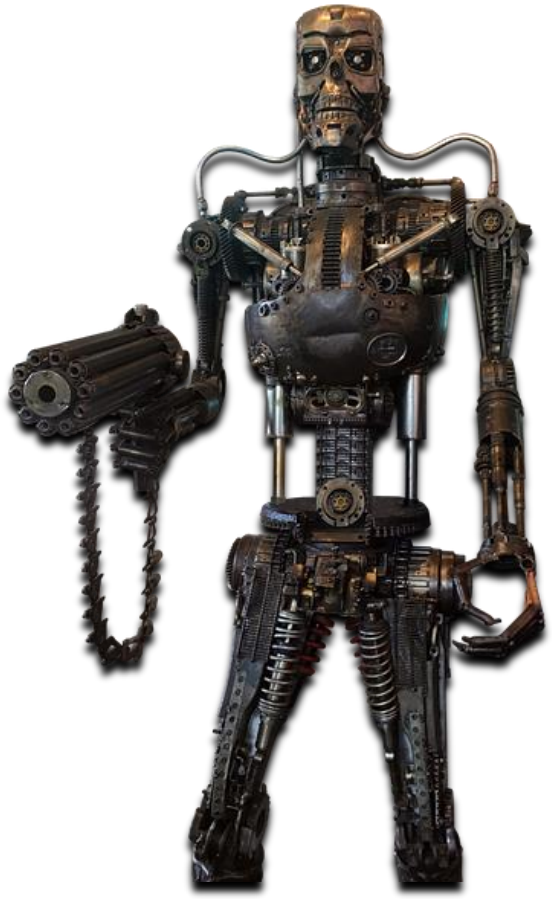
In the future, someone else is going to use and maintain this code.

They'll hold me responsible for any bug.

That person is a psychopath.

They know my home address.

They have a time machine.



arm

Channel your inner **gatekeeper**

Do we want this? Why?

- Bug fix? User request? Internally desired feature? Maintenance improvement?
- If I don't think we should do it, enquire internally

Look at the general shape

- Does it change what I think it should change?

Does it solve the problem?

- And is this the (or at least a) right way to solve the problem?
- Check against any applicable requirements or architecture document
  - If there isn't one, should there be?
- Am I aware of other ongoing work that it would conflict with?

Add JavaScript interpreter #666

Open totallyinnocent wants to merge 1 commit into ARMmbed:develo

arm

Channel your inner **user**

Is the PR useful?

- For its original objective
- For something slightly different

Do I understand the documentation?

- Do I understand which function(s) to call and how?
  - Do I understand how to build/configure? Should this be in the default build?
- Is it sufficiently clear and detailed? Is it well-written and formatted?
  - Especially: preconditions, error cases
- Does it need a changelog entry? a sample application? a knowledge base article?

arm

Channel your inner **conservative developper**

Looking at the code only, without the context of the PR,

Do I understand why the code is correct
and does what it says on the tin?

  - If not, it needs either a bug fix or a comment
- Robustness: what later changes might break it?
  - Will the compiler catch it? The tests? Static analysis?
- Are the changes justified? (if it ain't broke, don't fix it)
- Does the code conform to the documentation?
- Portability (does it work on the DeathStation 9000? do we care?)
- Anything else I can think of, anything I've broken/seen break in the past, …

Channel your inner **progressive developper**

Does the resulting code look right?

- Does the PR go far enough?
  - Is there further refactoring to do?
  - Should more functions be made public? Fewer?

and your inner **competitor**

I could do so much better!

- Can it be made more obviously correct? easier to maintain?
- Performance: could it use less code, use less memory, be faster, …?

Make improvements now? File issues for later?

arm

Channel your inner **attacker**

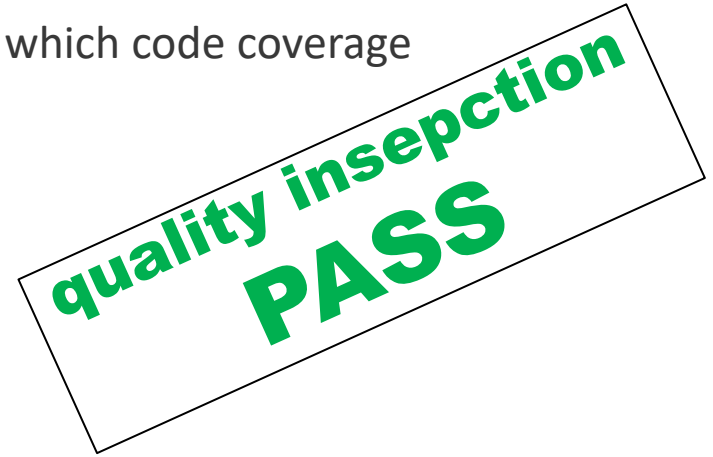
How do I break it?

- Input validation
- Buffer overflows and other pointer arithmetic
- Memory management (use of uninitialized memory, use after free, memory leak…)
- Are the documented preconditions sufficient to ensure the code is correct?
  - Should there be fewer documented preconditions and more checks in the code?
- Any other security concerns (e.g. side channels)

Channel your inner **quality assurancer**

Is the code well-tested?

- Bug fix: non-regression test if practical
- New feature: unit tests, integration/system tests if applicable
  - Tests for special cases (not just what the code *does* but what it *should do*, which code coverage measurements won't tell you)
- If any test is removed or modified, is this justified?
  - If a test needed to be changed, isn't this a compatibility break?



Does this conform to any applicable standard?

- Is the standard referenced in a comment?

Does this conform to our house rules? (Style, documentation habits, …)

Channel your inner **maintainer**

Backward compatibility

- What behaviors does this change? Does it break the API? the ABI? To what extent?
- User-facing documentation: is it clear what is guaranteed and what can change in future versions?

Suitability

- Usually this works for the author's use case. What other uses cases are there? Corner cases?
- If there's an API extension, is this what we'll still want in five years?

Maintainability

- Looking at the changes and the git commit messages only, do I understand what each step does?
  - Ok to need the context of the PR to understand the overall goal, but not to understand individual commits

Finally, channel your inner **everything**

Mindset: what's missing?

- Handling of special cases
- Documentation
- Tests
- Updates to build/test scripts
- Behavior in non-default configurations
- Things that I've (seen) forgotten in the past

Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!
감사합니다
धन्यवाद

Image credits

[2] Terminator Robot Futuristic Machine by jean52Photosstock. Pixabay
[3] Workaround (extract) by Randall Monroe. CC BY-NC 2.5
[4] Untitled, unknown artist. CC0 (public domain).
[5] Stubborn Donkey by Asaf Braverman. CC BY-NC-SA 2.0
[6] Ivory okimono of an elephant trampling a tiger, unknown artist, Japan. Photo by Galleries of Wolverhampton. CC BY-NC-SA 2.0
[6] Bear climbing a tree by Nicolas Vollmer (extract). CC BY 2.0
[6] Wolf Howling by skeeze. Pixabay
[7] Anonymous hacker behind pc by elconomeno. CC0 (public domain).
[8] Transpalatial arch expansion (extract) by Giorgio Fiorelli. CC BY 3.0
[10] The Oklahoma by Vegan Feast Catering. CC BY 2.0

arm