ACRUMEN:

What IS Software Quality Anyway?!

by
Dave Aronson

CURRENT TIME: 6:00 EXACTLY, want 5-7, perfect, watch the pace

dup slide so i can flip to the next one to start the keynote timer

content is moved left ~100 pt so as not to be blocked by speaker-view in zoom

ACRUMEN:

What IS Software Quality Anyway?!

by
Dave Aronson

Mr. Contestmaster, fellow Toastmasters, and honored guests: I'm a software developer with almost 40 years of experience, and I'm here to tell you all about my definition of software quality!

Image: https://tenor.com/view/why-huh-but-why-gif-13199396

But *why*? After all, most of you are presumably not software developers!

However, we all *use* software. For instance, we're using Zoom right now. I think most of us are, to put it politely, frequently dissatisfied with the *quality* of a lot of the software we use. But again, *why?* Why is their quality so often so low? I think a large part of the problem is the lack of an agreed definition.

Several years ago, I was *looking* for a good . . .

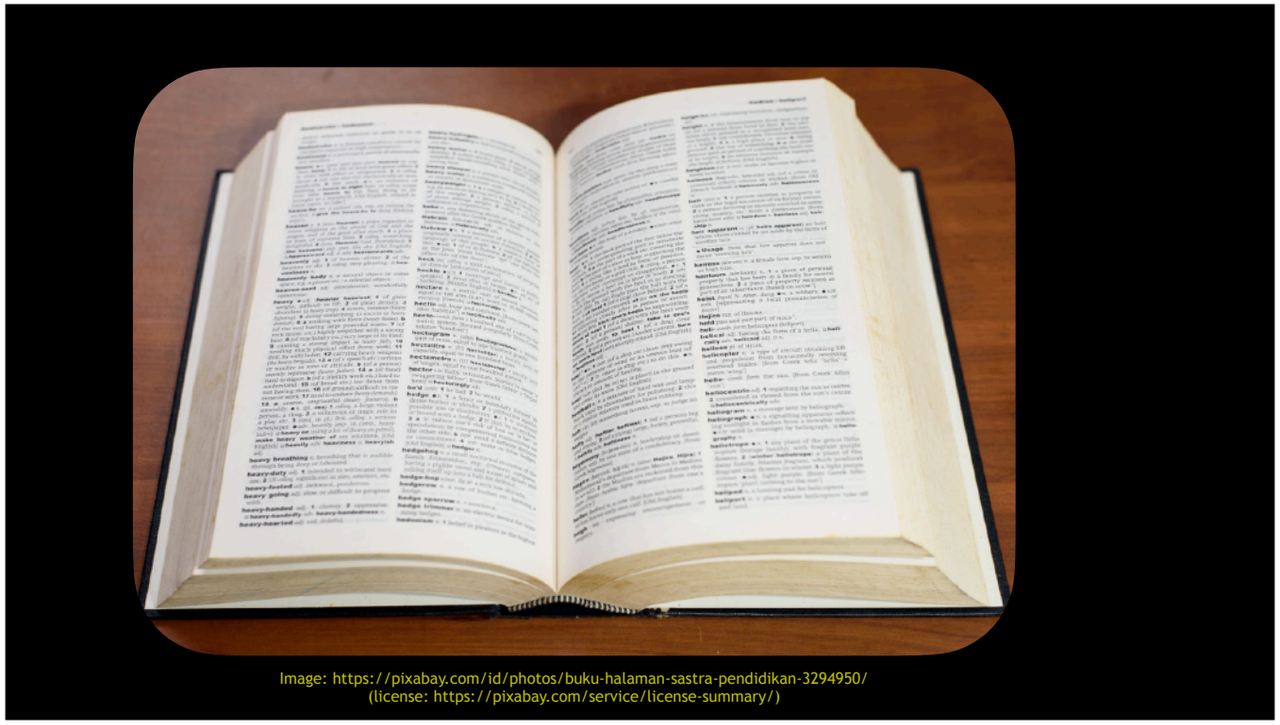


Image: https://pixabay.com/id/photos/buku-halaman-sastra-pendidikan-3294950/
(license: https://pixabay.com/service/license-summary/)

. . . definition, but the ones I found all had rather serious problems. Most were . . .

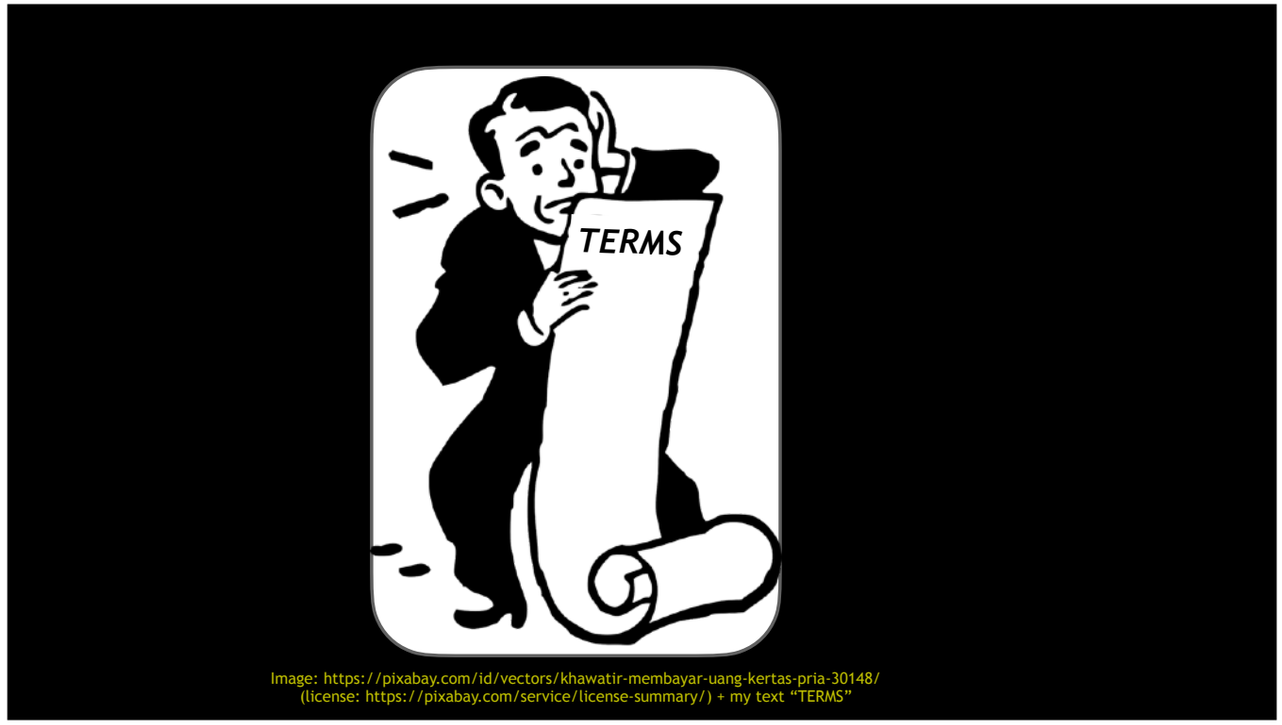


Image: https://pixabay.com/id/vectors/khawatir-membayar-uang-kertas-pria-30148/
(license: https://pixabay.com/service/license-summary/) + my text “TERMS”

. . . long lists of complex ideas, full of developer jargon and other technical terms, but I wanted something short and simple, so that non-techies would understand it too, so they could give us more precise feedback about *exactly how* our software . . . falls short, shall we say. Some definitions were . . .

Image: https://www.flickr.com/photos/59937401@N07/5857412037
(license: https://creativecommons.org/licenses/by/2.0/)

. . . proprietary, or applicable only to certain styles or technologies, sometimes also proprietary, or otherwise restricted. But I wanted something everybody could use, with all software, for free. Some definitions weren't even about the software at all, but all about the . . .

. . . process, or its byproducts, like meetings and documents. Some of these meetings and documents may be helpful, but to make them the definition, I felt completely missed the point. I wanted something more focused on the software itself, and *de*scriptive rather than *pre*scriptive. Long story short, I didn't see anything that I liked, nor that was commonly accepted. So, in the spirit of the popular techie comic strip . . .

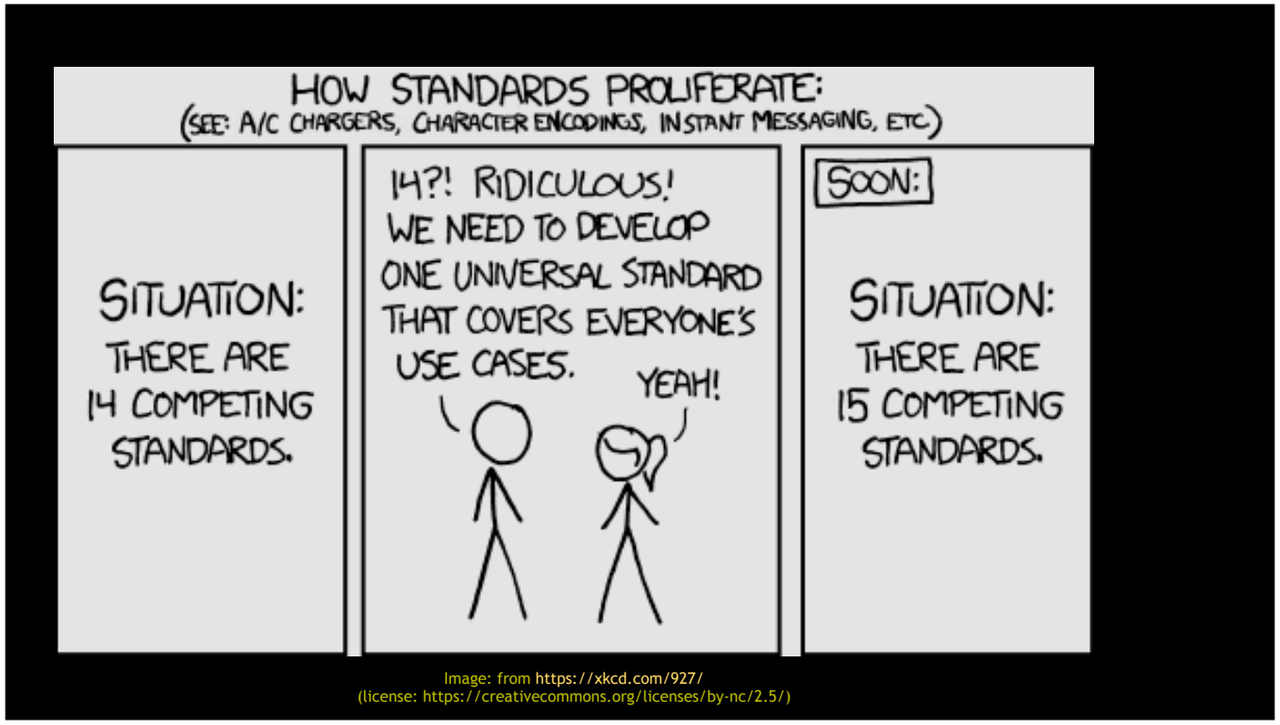


Image: from https://xkcd.com/927/
(license: https://creativecommons.org/licenses/by-nc/2.5/)

. . . XKCD (PAUSE!), I decided to make my own. However, rather than the usual approach of taking the best parts of all the existing approaches, I decided to whittle it down to just the bare essentials. Eventually, I came up with something I call . . .

ACRUMEN

. . . ACRUMEN. This is a Latin word, meaning “sour fruit”, which is why you’ll see so much lemon yellow in these slides. But what is it in this context? The acronym ACRUMEN (try saying *that* ten times fast!), stands for the idea that . . .

ACRUMEN means software should be:

. . . software should be (INHALE!) . . .

ACRUMEN means software should be:

Appropriate

Correct

Robust

Usable

Maintainable

Efficient

. . . Appropriate, Correct, Robust, Usable, Maintainable, and Efficient, usually in that order. But what does all *that* mean? First and foremost, actually more important than all the rest put together, it needs to . . .

ACRUMEN means software should be:

Appropriate : doing what the stakeholders need

Correct

Robust

Usable

Maintainable

Efficient

. . . *do what the stakeholders need* it to do, in other words, do the right job. And please notice that I say “stakeholders”, which is a lot more people than just the users or the customers. Next it needs to . . .

ACRUMEN means software should be:

Appropriate : doing what the stakeholders need

Correct : doing its job(s) without errors

Robust

Usable

Maintainable

Efficient

. . . do that job, or more likely those jobs, correctly, without bugs or other errors, or in other words, do the job right. It's pretty much exactly what it sounds like. Next, it should be . . .

ACRUMEN means software should be:

Appropriate : doing what the stakeholders need

Correct : doing its job(s) without errors

Robust : hard to make it crash/malfunction/etc.

Usable

Maintainable

Efficient

. . . hard for anyone to make the software crash or malfunction, or even *seem* to, whether deliberately *or accidentally*. In particular, note that malfunctioning includes revealing or altering information when it's not supposed to, and other security concepts. On the other hand, it should be . . .

ACRUMEN means software should be:

Appropriate : doing what the stakeholders need

Correct : doing its job(s) without errors

Robust : hard to make it crash/malfunction/etc.

Usable : easy for the users to use

Maintainable

Efficient

. . . easy for the users to use. Many people say “usable” and really mean “accessible”, but that’s just a subset. In my view, software should be *easy for everyone* to use, not just *equally difficult!* It should also be . . .

ACRUMEN means software should be:

Appropriate : doing what the stakeholders need

Correct : doing its job(s) without errors

Robust : hard to make it crash/malfunction/etc.

Usable : easy for the users to use

Maintainable: easy for the developers to change

Efficient

. . . easy for the developers to change, *with* low chance of error, *and* low *fear* of error. Last, dead last despite how we developers tended to worship this just a few short decades ago, it should . . .

ACRUMEN means software should be:

Appropriate : doing what the stakeholders need

Correct : doing its job(s) without errors

Robust : hard to make it crash/malfunction/etc.

Usable : easy for the users to use

Maintainable: easy for the developers to change

Efficient : easy on resources

. . . go easy on resources, not only the technical ones we usually think of like CPU and memory, but also “soft” resources like the user’s patience and brainpower, and the company’s money.

Now, I’ve said there are six aspects, and there are six listed up there, but ACRUMEN has seven letters! So what does the N stand for? Nnnnn . . .

nothing, I just tacked it on to make a real word, even if an obsolete one.

So how can we *apply* this definition? Mainly, we can keep it in mind as a mental . . .

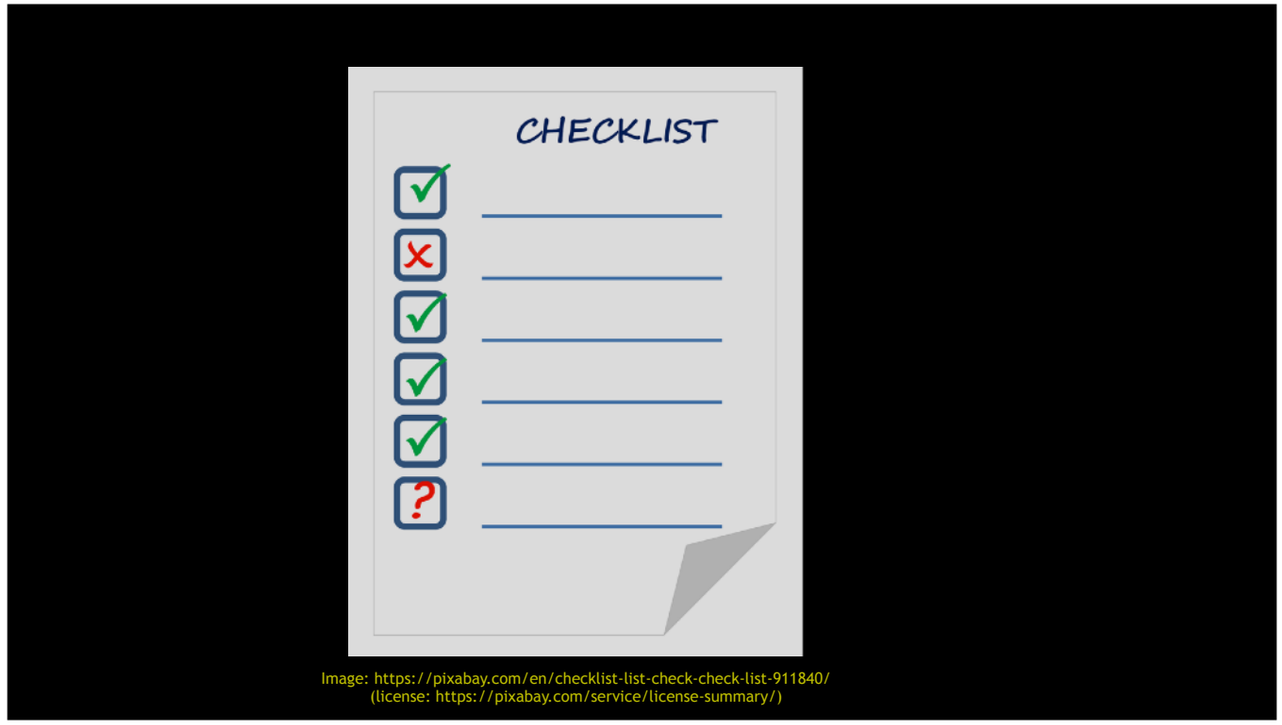


Image: https://pixabay.com/en/checklist-list-check-check-list-911840/
(license: https://pixabay.com/service/license-summary/)

. . . *checklist*, when writing or evaluating software. We can ask, *is* it Appropriate, *is* it Correct, and so on, or *how* good is it in each aspect, whether that be on some scale, or by simple triage, or is it *good enough* for *our needs?* And if the answer is ever that it's not good enough, we can ask, what can be done to *. . .*

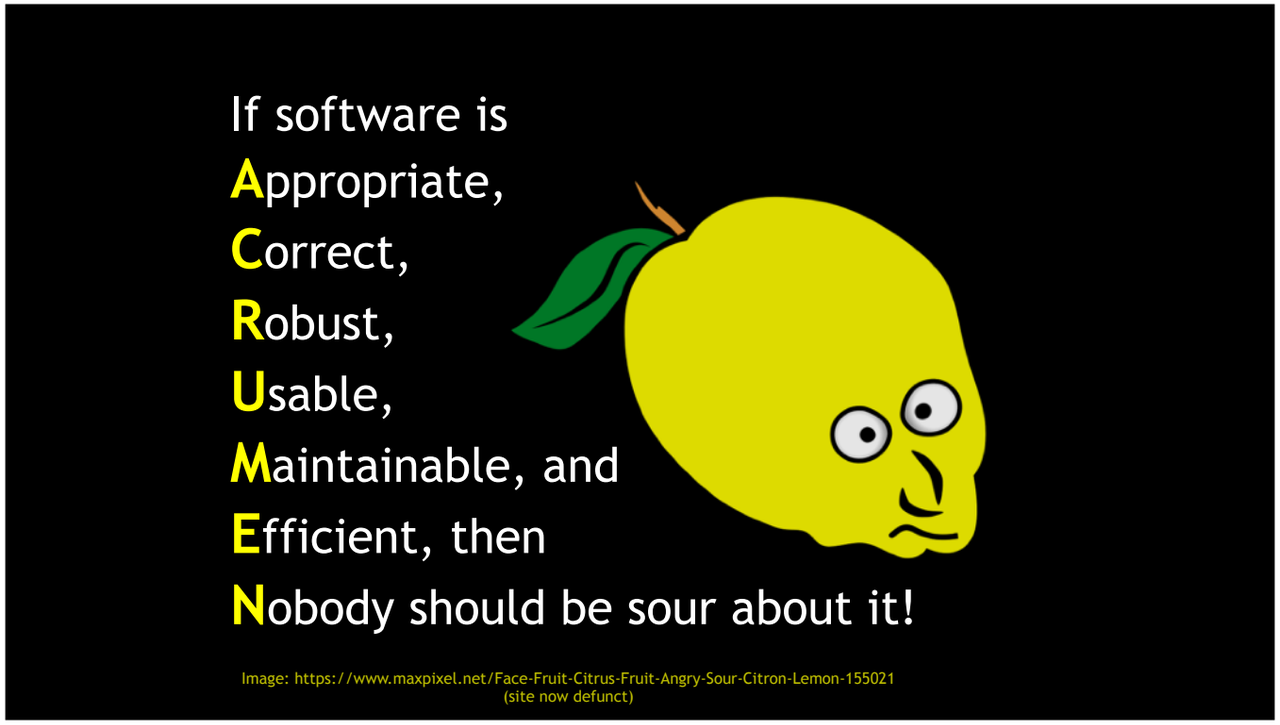
Image: https://www.flickr.com/photos/syume/4908943709/
(license: https://creativecommons.org/licenses/by-nc-nd/2.0/)

. . . *make* it so? Software developers, and their management, can also set . . .

Image: https://www.flickr.com/photos/bensutherland/205606714
(license: https://creativecommons.org/licenses/by/2.0/)

. . . targets, for how good they *need* their software to be in each aspect. Looking at software through the lens of this list of aspects can help prioritize any further work on it.

In conclusion . . .



. . . if we software developers remember to make sure that our software is appropriate, correct, robust, usable, maintainable, and efficient, then nobody should have any cause to be sour about the FRUITS OF OUR LABORS.

Mr. Contestmaster . . . .