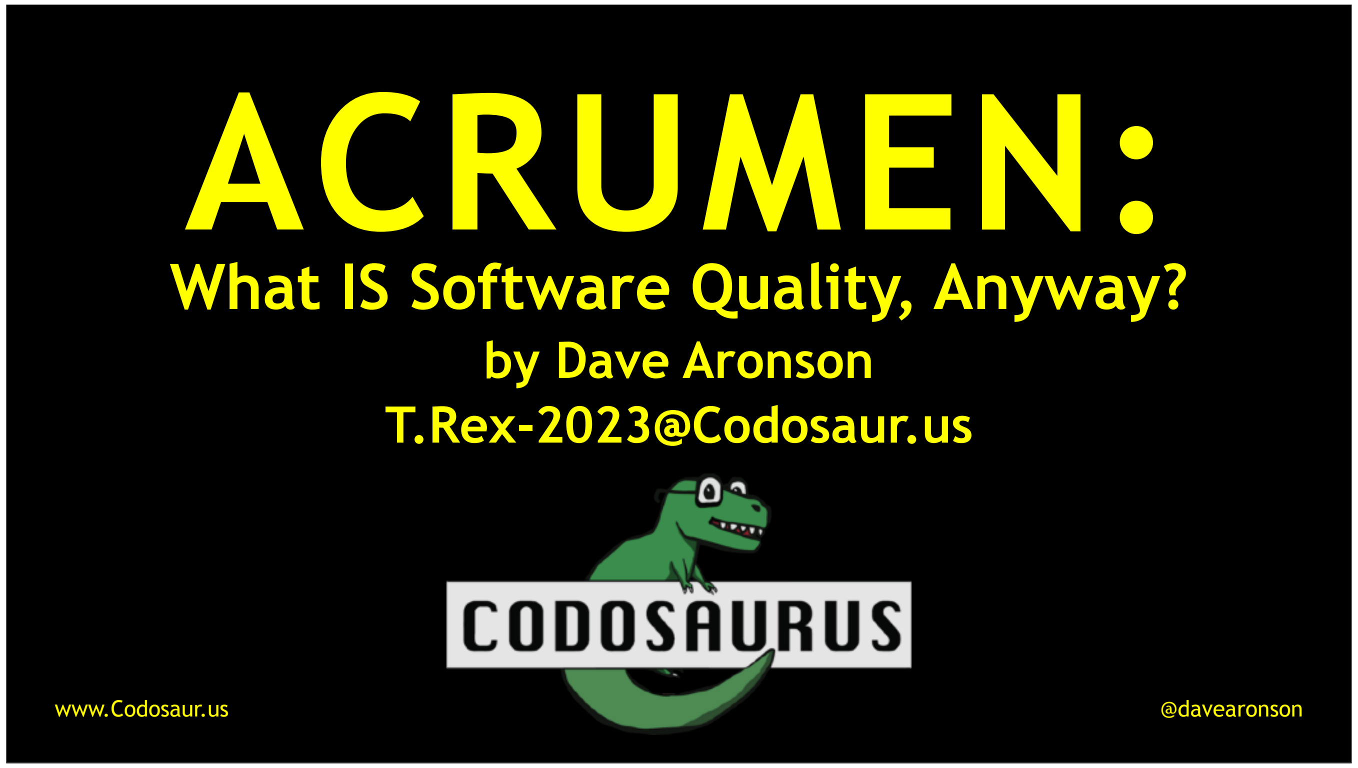


ACRUMEN:

What IS Software Quality, Anyway?

by Dave Aronson
T.Rex-2023@Codosaur.us

CODOSAURUS

www.Codosaur.us

@davearonson

(NOTE TO SELF: HAVE BUSINESS CARD READY! Maybe find a better picture for “how do we know”?)

Current time: ~18.5-19 mins, want 20 MAX, so WATCH THE AD-LIBS!

ACRUMEN:

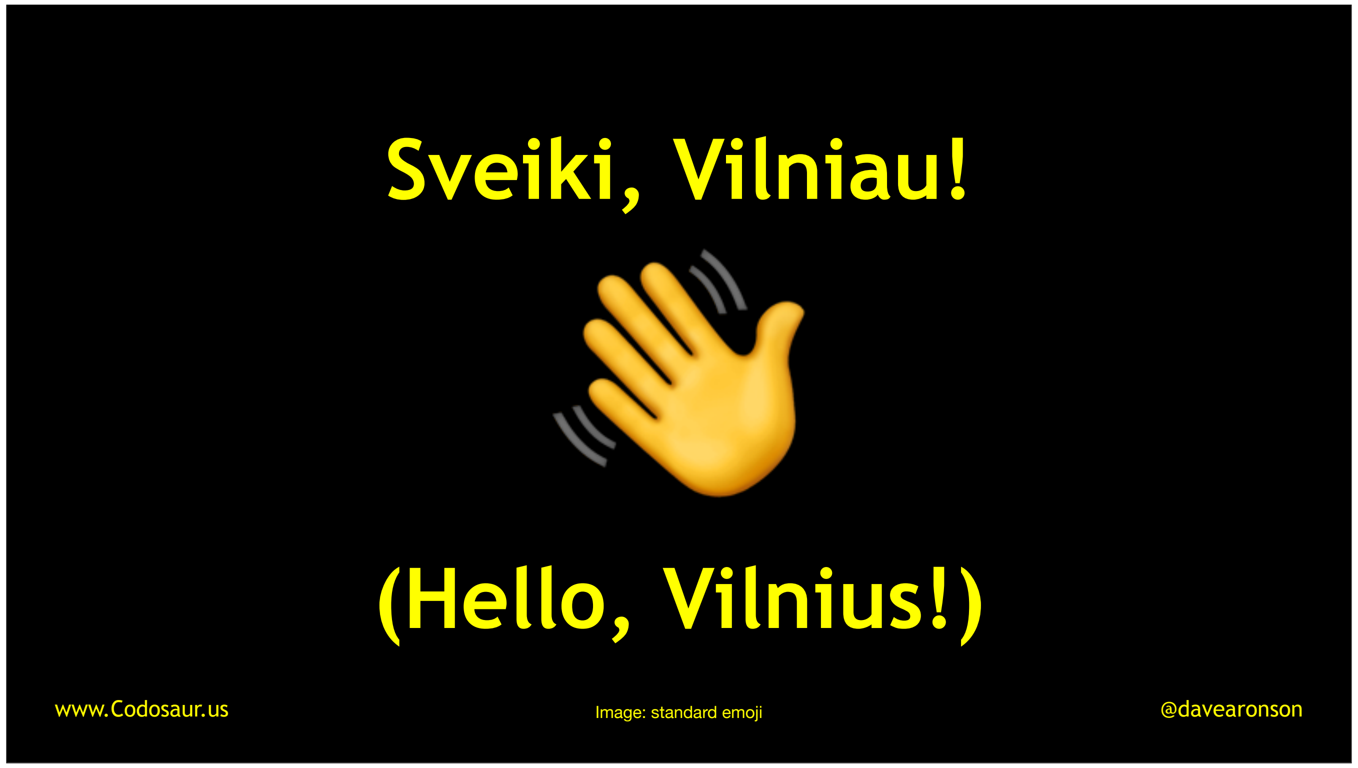
What IS Software Quality, Anyway?

by Dave Aronson
T.Rex-2023@Codosaur.us

CODOSAURUS

www.Codosaur.us

@davearonson



Sveiki, Vilniau!

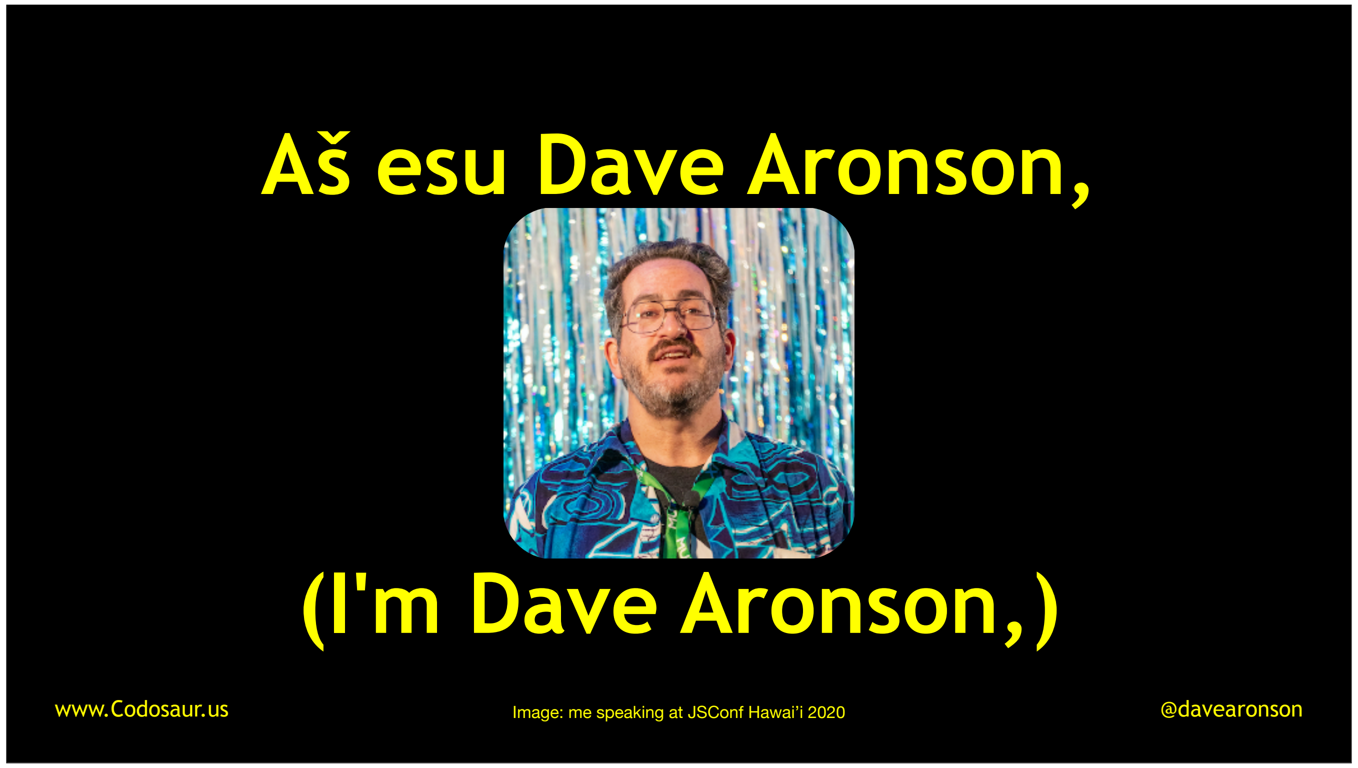
(Hello, Vilnius!)

www.Codosaur.us

Image: standard emoji

@davearonson

SveikEE VILnio!



Aš esu Dave Aronson,

(I'm Dave Aronson,)

www.Codosaur.us

Image: me speaking at JSConf Hawai'i 2020

@davearonson

AHSH EHsoo Dave Arrronson,

T. Reksas iš Codosaurus,

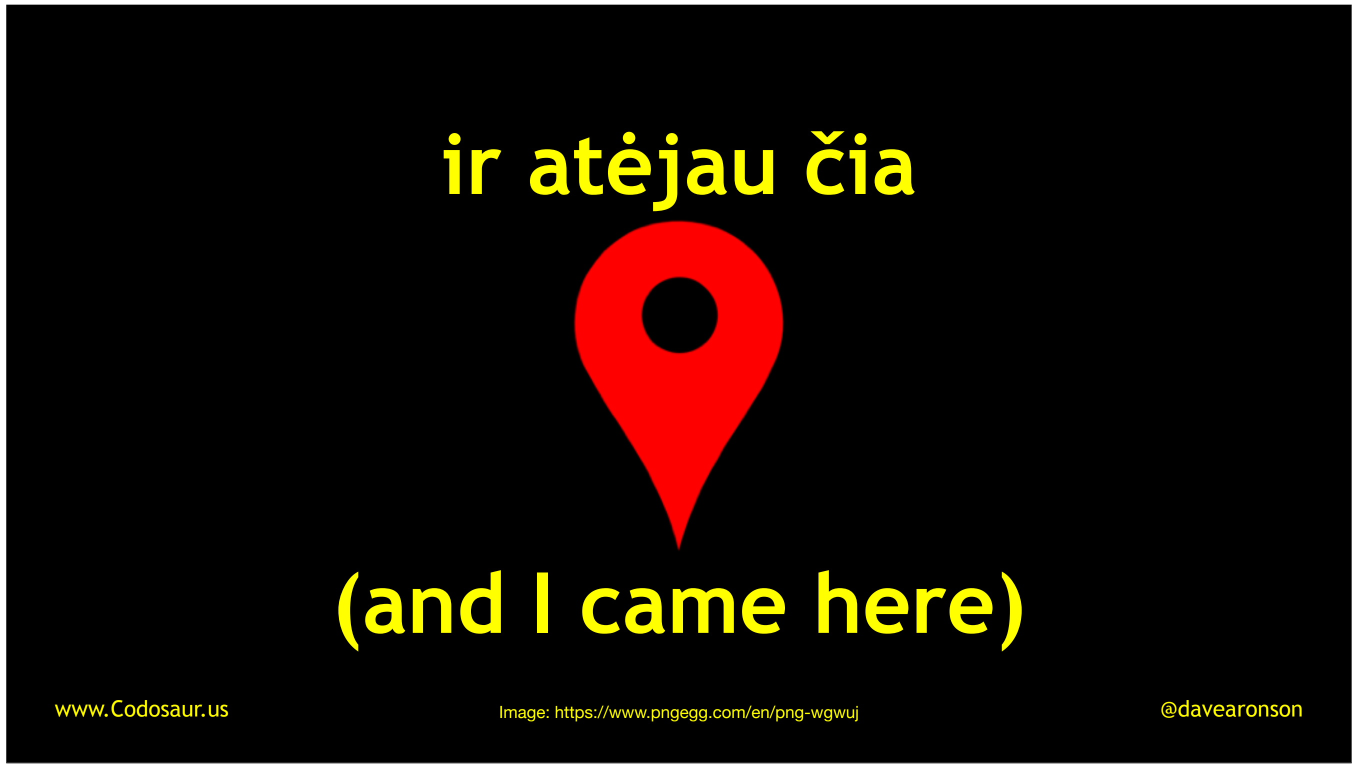
(the T. Rex of Codosaurus,)

www.Codosaur.us

Image: my company logo!

@davearonson

T. RRREKsas ish Codosowrus,



ir atėjau čia

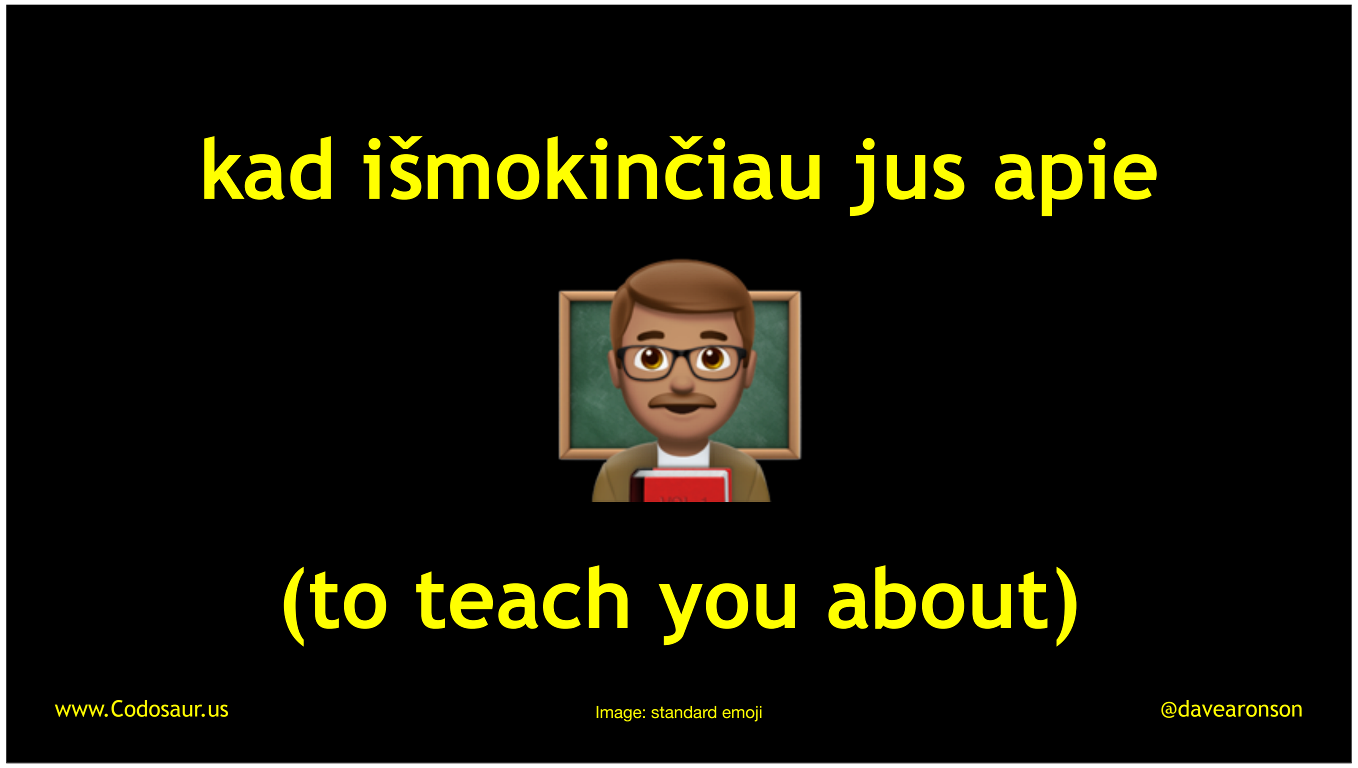
(and I came here)

www.Codosaur.us

Image: https://www.pngegg.com/en/png-wgwuj

@davearonson

irrr ahTAYyo cheh



kad išmokinčiau jus apie

(to teach you about)

www.Codosaur.us

Image: standard emoji

@davearonson

kad ishmoKIHNchyo yoos ah-PEE-eh

ACRUMEN

www.Codosaur.us

@davearonson

ACRUMEN!

Visgi . . .

☝

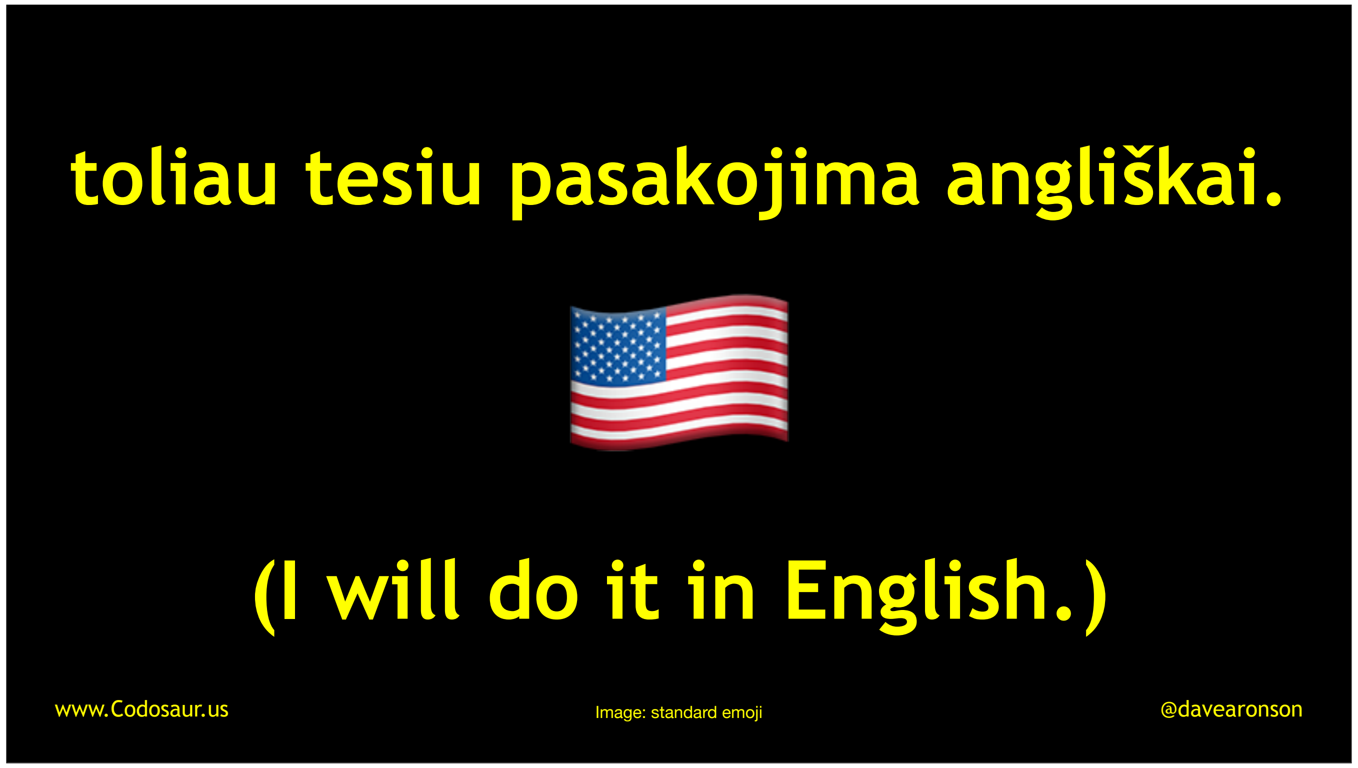
(But . . .)

www.Codosaur.us

Image: standard emoji

@davearonson

VIZgih . . .



toliau tesiu pasakojima angliškai.

(I will do it in English.)

www.Codosaur.us

Image: standard emoji

@davearonson

tohlYO TESSu PassaKoimah AHNglishkay. (PAUSE!)

Mainly because you've just heard almost all the Lithuanian I speak! (PAUSE!)

This talk will be in sharp contrast to the others. You've heard about features specific to .NET, TypeScript, and C#. But this talk will be so technology-agnostic, that there is actually . . .

www.Codosaur.us
Image: code of my github.com/CodosaurusLLC/genaver, colored by Reveal.JS, plus standard nope-scope
@davearonson

. . . no code in it at all! Not for any lack of coding expertise, as I have over 38 years of experience in a wide variety of programming languages. Rather, it's mainly to emphasize that, despite the way many of us *treat* it so, it's ***not** all about the code!* In fact, I deliberately avoid saying "*code* quality", since that is just *part* of my overall topic of *software* quality.

So I'm here to teach you my definition of software quality, but . . .

www.Codosaur.us Image: https://tenor.com/view/why-huh-but-why-gif-13199396 @davearonson

. . . why? We all agree we need more quality, but without a *definition*, it's very hard to achieve, or to get someone else to acknowledge that we have achieved it. So, I'm trying to get everybody on the same page. (Yeah, good luck with that!)

Several years ago, I was *looking* for a good definition, but all the ones I found had serious problems. Most were . . .

www.Codosaur.us Image: https://pixabay.com/id/vectors/khawatir-membayar-uang-kertas-pria-30148/ + my text @davearonson

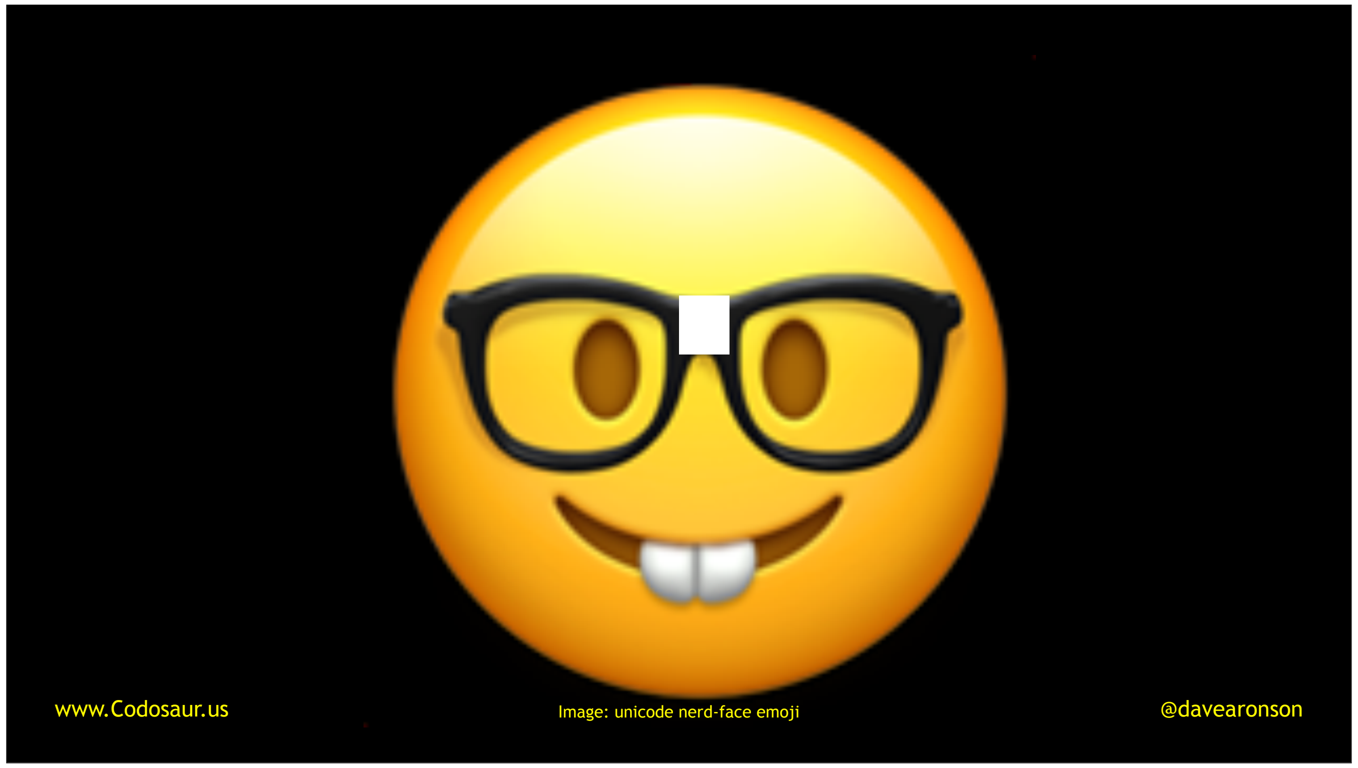
. . . long lists of complicated terms, full of developer jargon. That's fine for talking amongst *ourselves*, but I wanted a definition that *even non-technical* people would understand, so they could give us more precise *feedback* about *exactly how* our software sucks.

Some definitions were . . .

www.Codosaur.us Image: https://www.flickr.com/photos/59937401@N07/5857412037 @davearonson

. . . proprietary, making us buy expensive software or documents. Some were only applicable within the context of certain styles or technologies, often also proprietary. I felt that all of that was just plain wrong. I wanted something we could *all* use, for *free*.

Some definitions focused exclusively on issues of interest to . . .

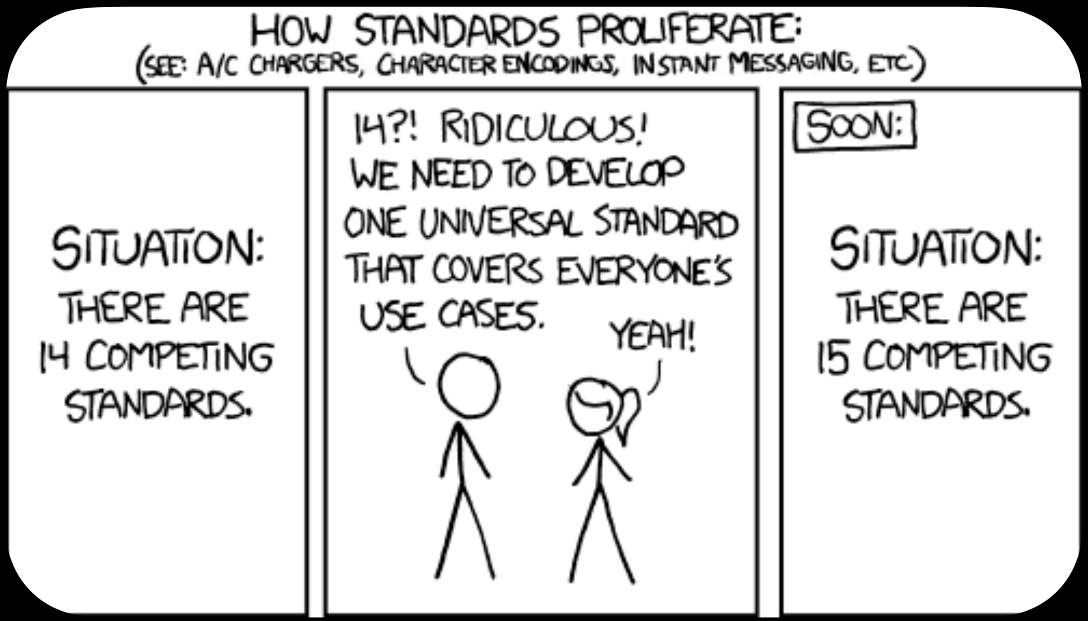


. . . us developers, ignoring the needs of the users and other stakeholders. Some weren't even about the software at all, but all about the . . .

www.Codosaur.us
Images: https://pixabay.com/illustrations/meeting-conference-personal-3321175/
and many copies of https://pixabay.com/vectors/papers-stack-heap-documents-576385/
@davearonson

. . . process, or the byproducts, like meetings or documents. These may be helpful, but to make them the *definition*, I felt totally missed the point.

I didn't see anything that I liked, nor that was commonly accepted, so in the spirit of . . .


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

. . . XKCD (PAUSE!), I decided to make my own.

To keep it simple, I (step back) zoomed out from . . .

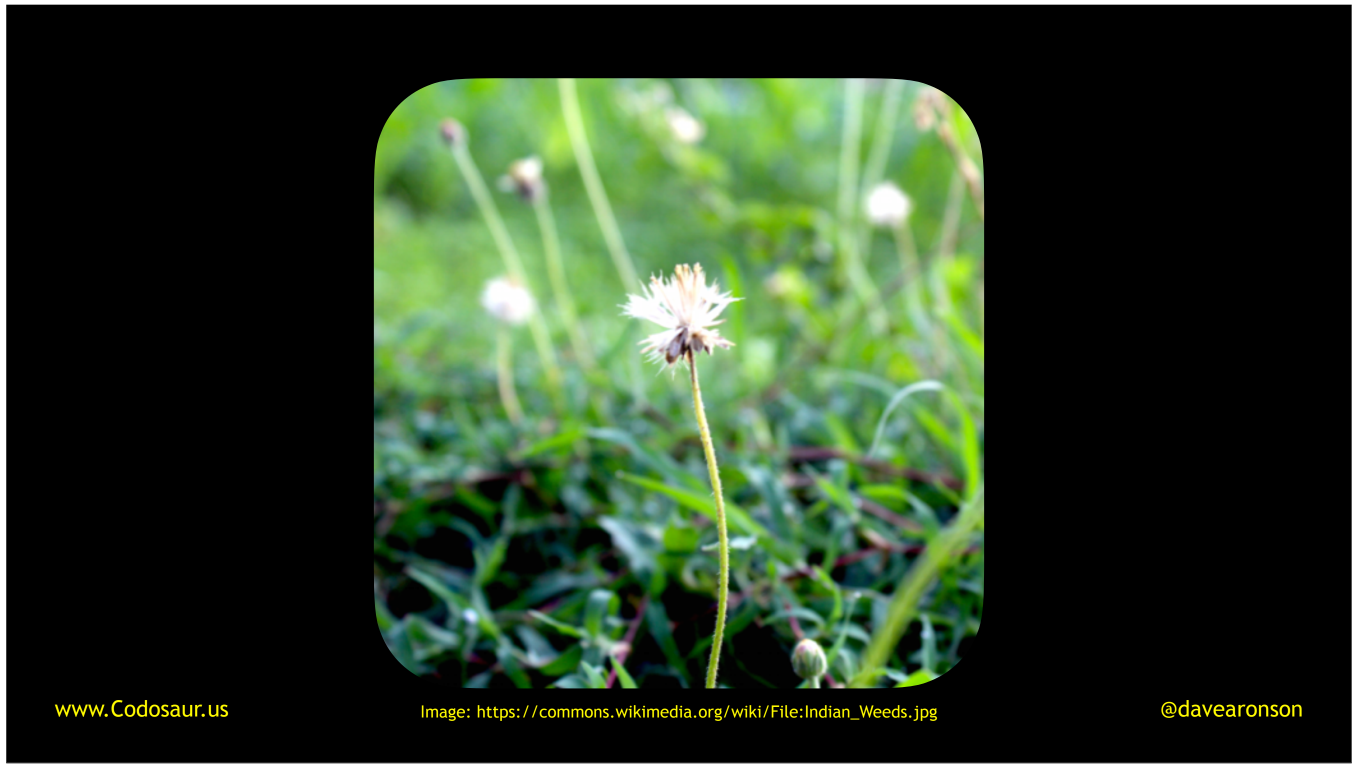

www.Codosaur.us

Image: https://commons.wikimedia.org/wiki/File:Indian_Weeds.jpg

@davearonson

. . . down in the weeds, where we developers tend to live, up to about . . .

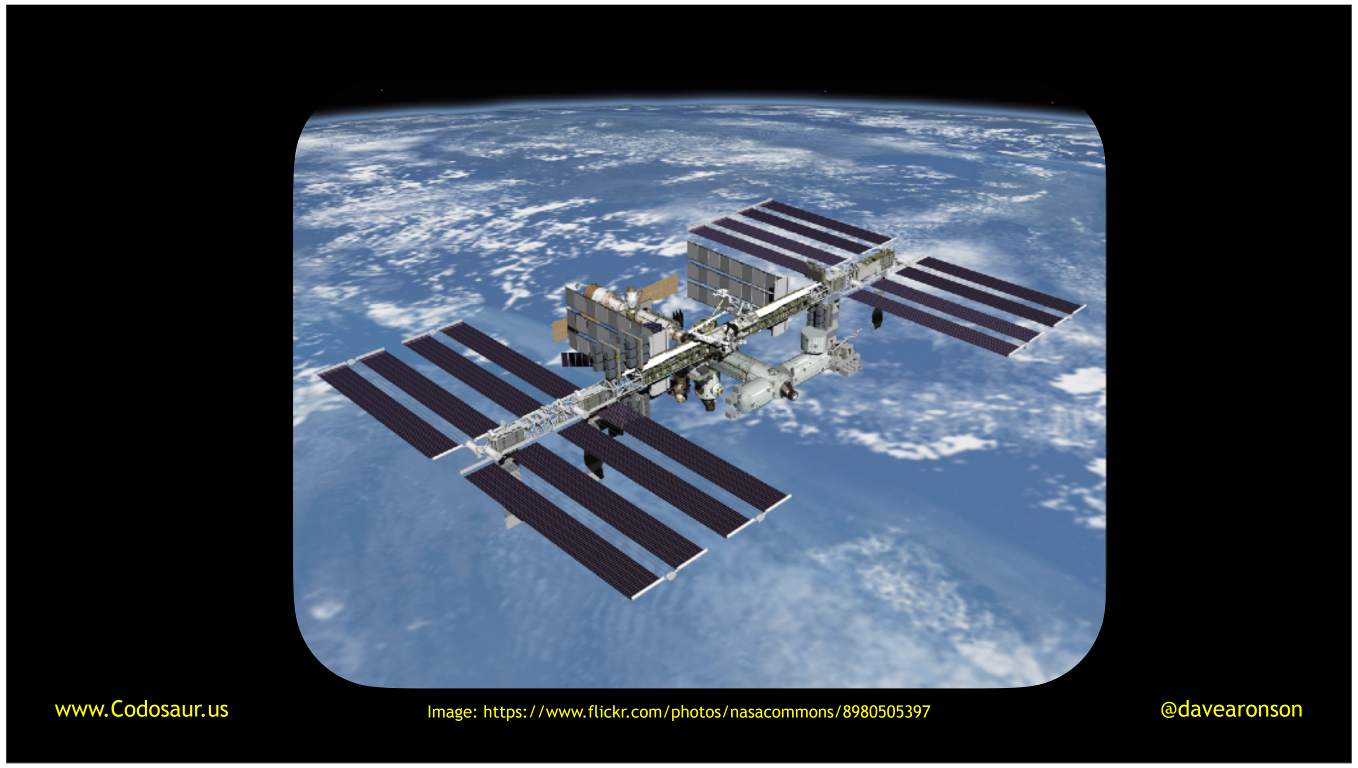
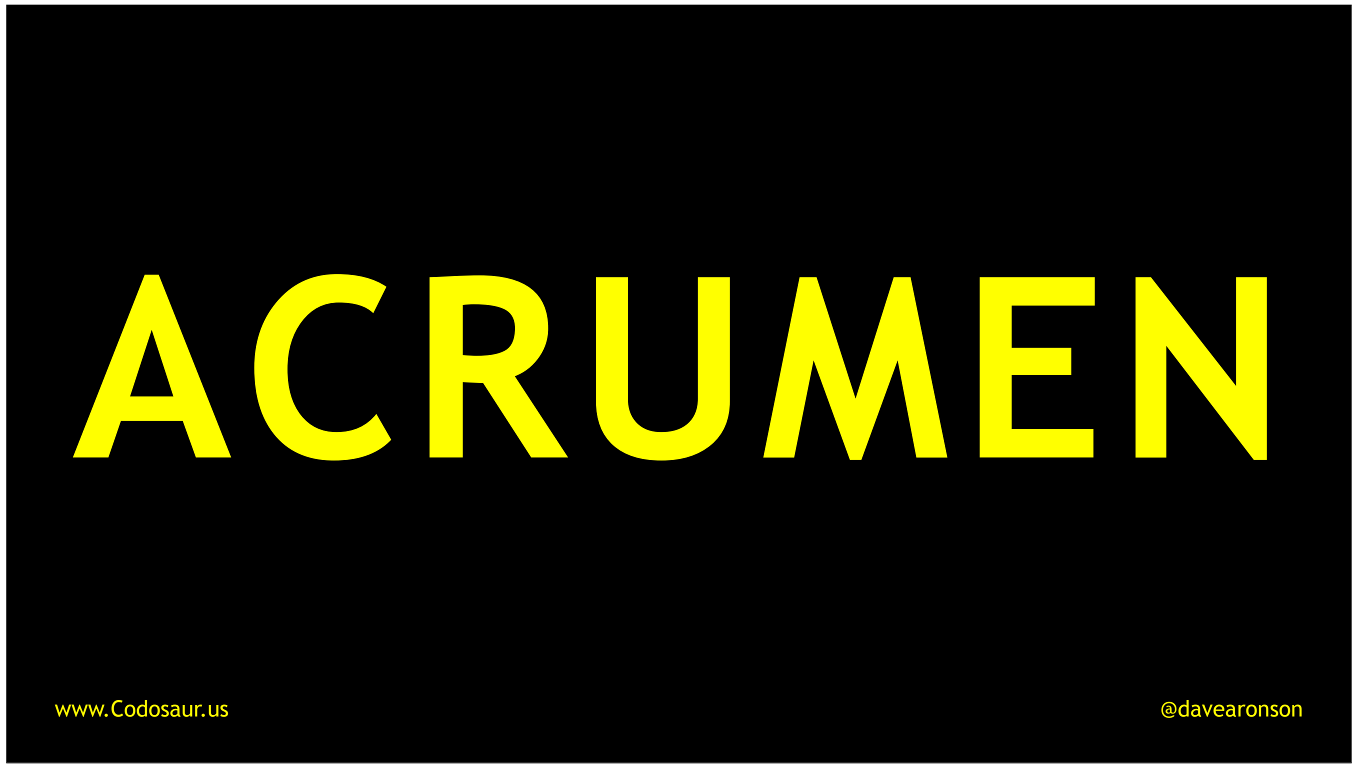

www.Codosaur.us

Image: https://www.flickr.com/photos/nasacommons/8980505397

@davearonson

. . . low earth orbit, so I could look at continents, not pebbles. That let me trim it down to just six aspects, with simple names and *relatively* simple explanations. The overall explanation literally fits on the back of a business card, and (HOLD UP BIZ CARD!) here's mine to prove it. See me afterward if you want one as a cheat-sheet.

I call this list of aspects . . .



ACRUMEN

www.Codosaur.us @davearonson

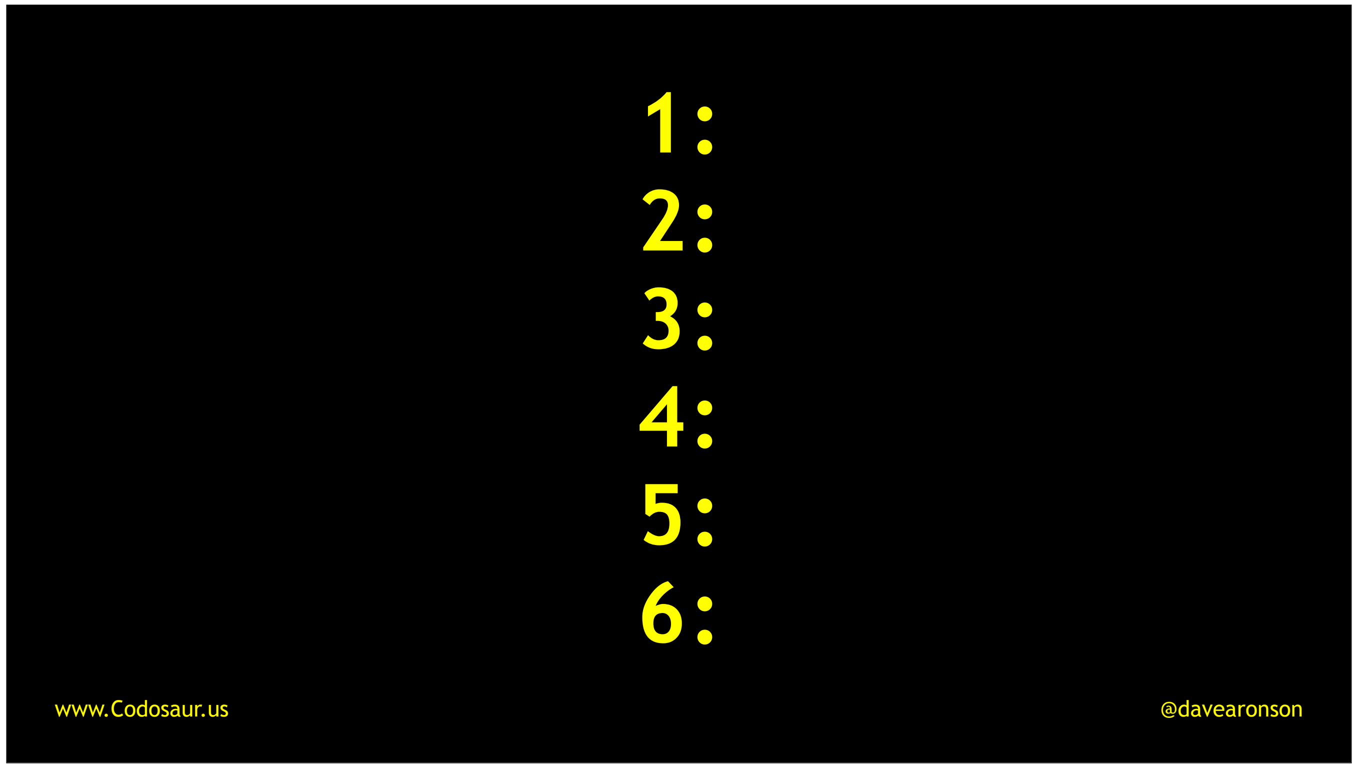
. . . ACRUMEN, which is a Latin word for sour fruit, such as . . .

www.Codosaur.us
Image: https://pixabay.com/en/lemonade-lemons-glass-beverage-1447521/
@davearonson

. . . lemons. That's why lemon yellow is the Official Color of ACRUMEN.

But what is ACRUMEN in *this* context?

The *acronym* ACRUMEN (try saying that ten times fast!), just takes those six aspects, and . . .

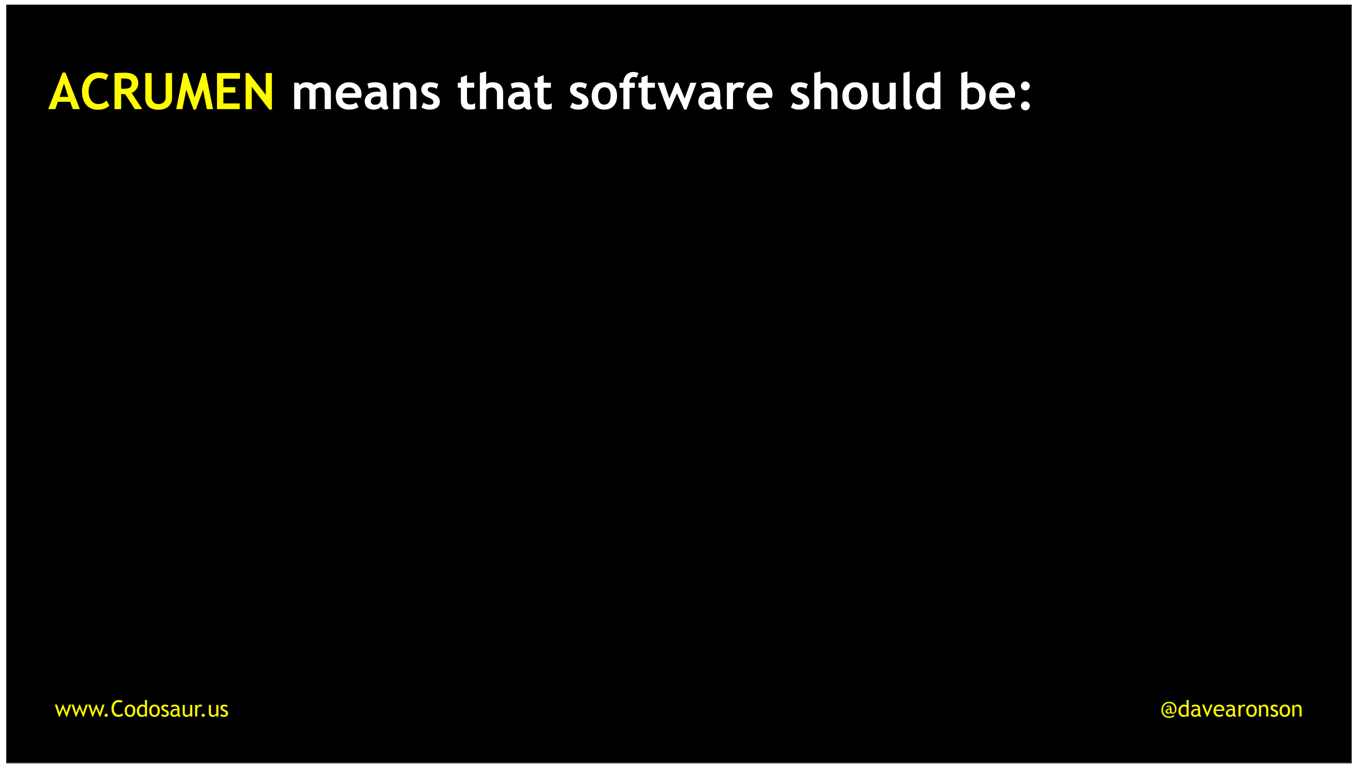

1:
2:
3:
4:
5:
6:

www.Codosaur.us

@davearonson

. . . puts them in priority order.

By now you're probably wondering, WHAT ARE THE ASPECTS ALREADY?! They are that . . .

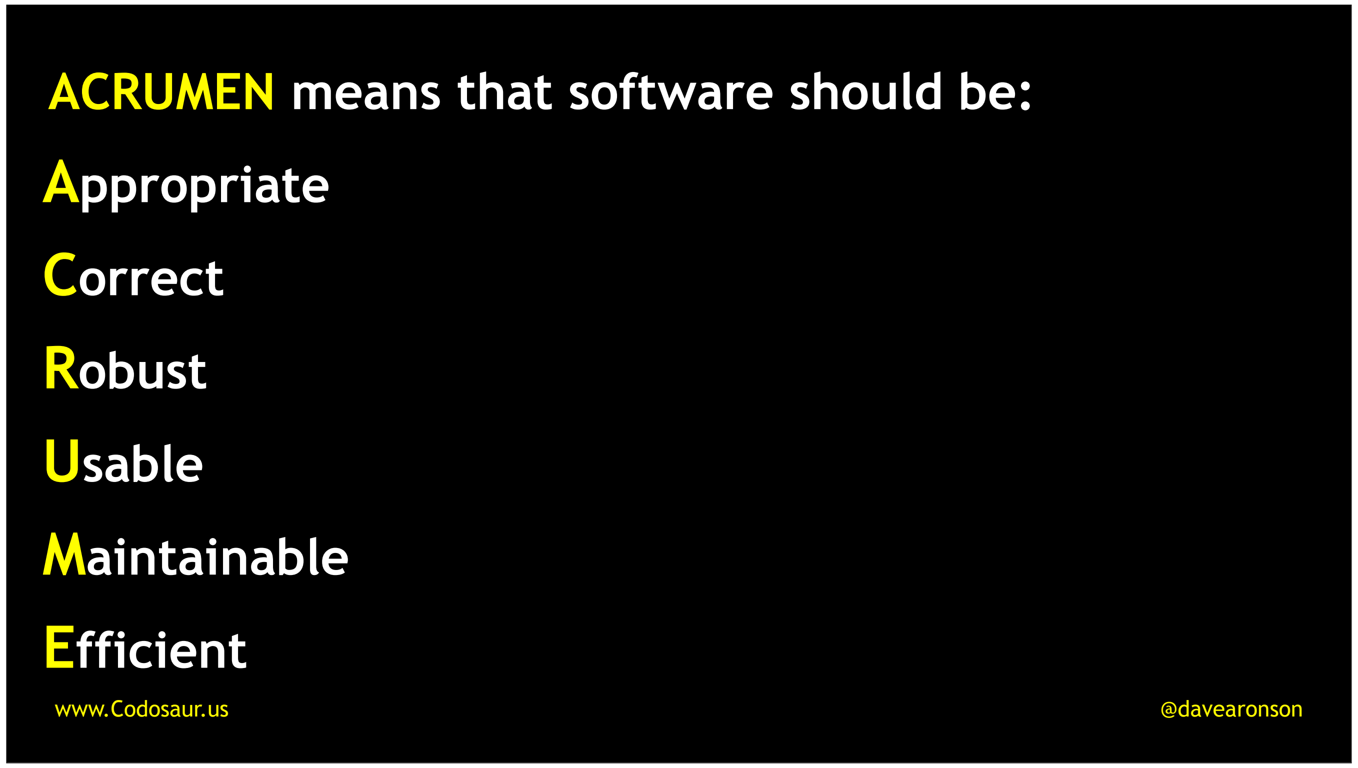


ACRUMEN means that software should be:

www.Codosaur.us

@davearonson

. . . software should be: (INHALE) . . .



ACRUMEN means that software should be:

Appropriate

Correct

Robust

Usable

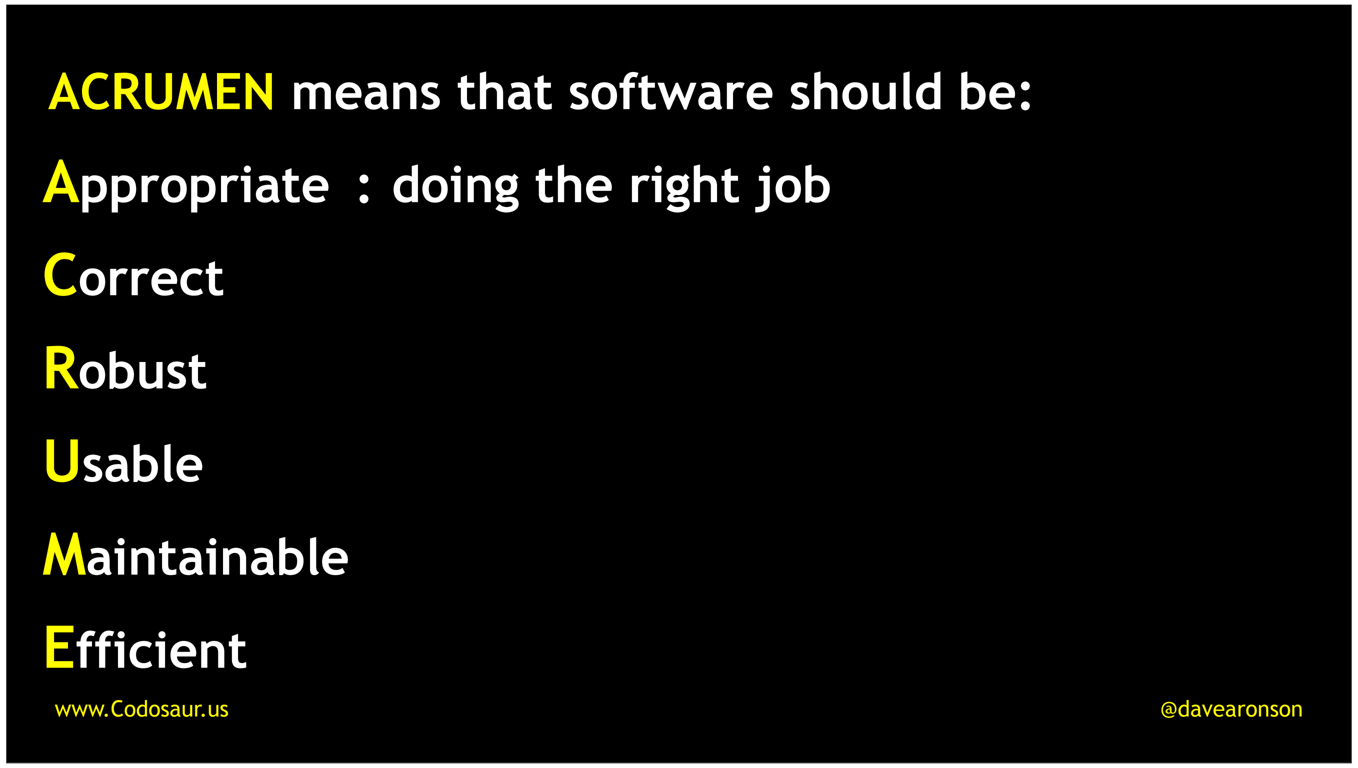
Maintainable

Efficient

www.Codosaur.us

@davearonson

. . . Appropriate, Correct, Robust, Usable, Maintainable, and Efficient. But what does all *that* mean? First, it needs to be . . .



ACRUMEN means that software should be:

Appropriate : doing the right job

Correct

Robust

Usable

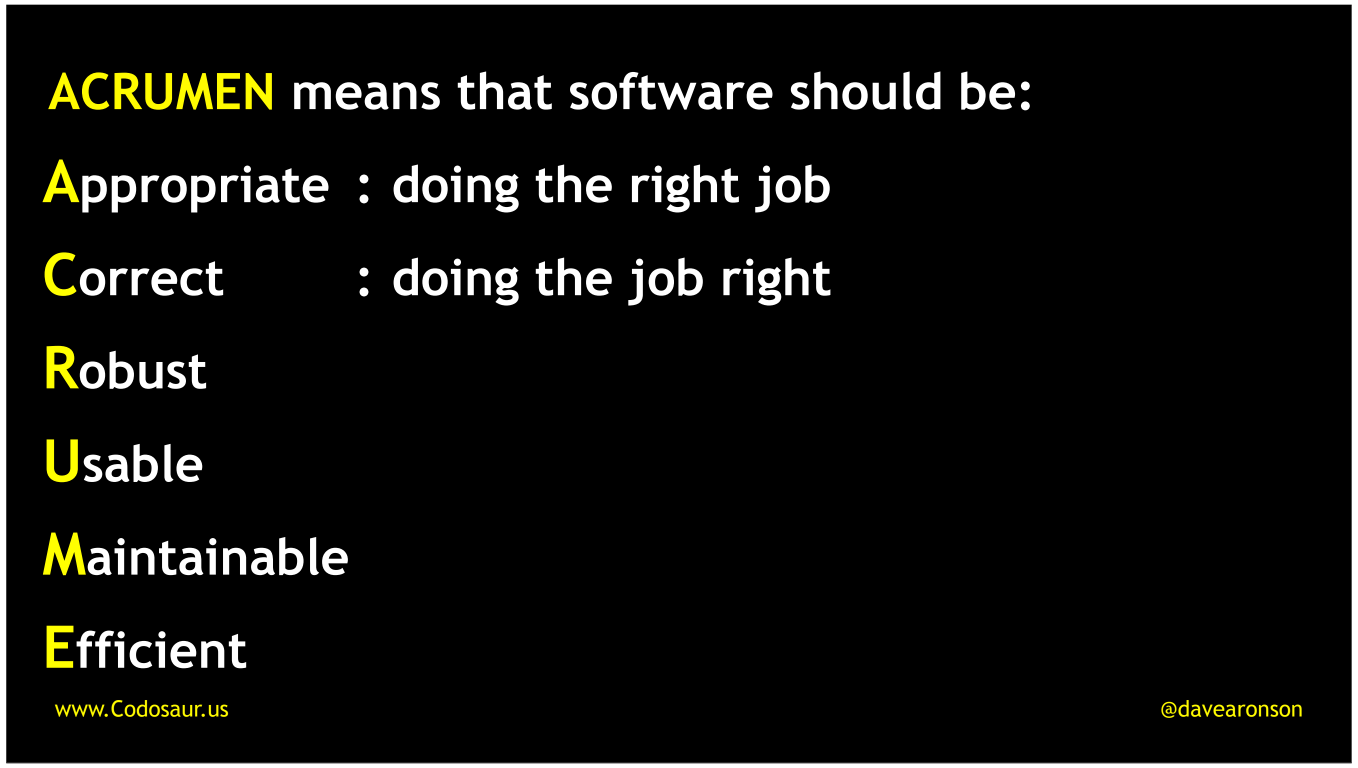
Maintainable

Efficient

www.Codosaur.us

@davearonson

. . . *doing what the stakeholders need* it to do, in other words, doing the *right job*. Then it needs to be . . .



ACRUMEN means that software should be:

Appropriate : doing the right job

Correct : doing the job right

Robust

Usable

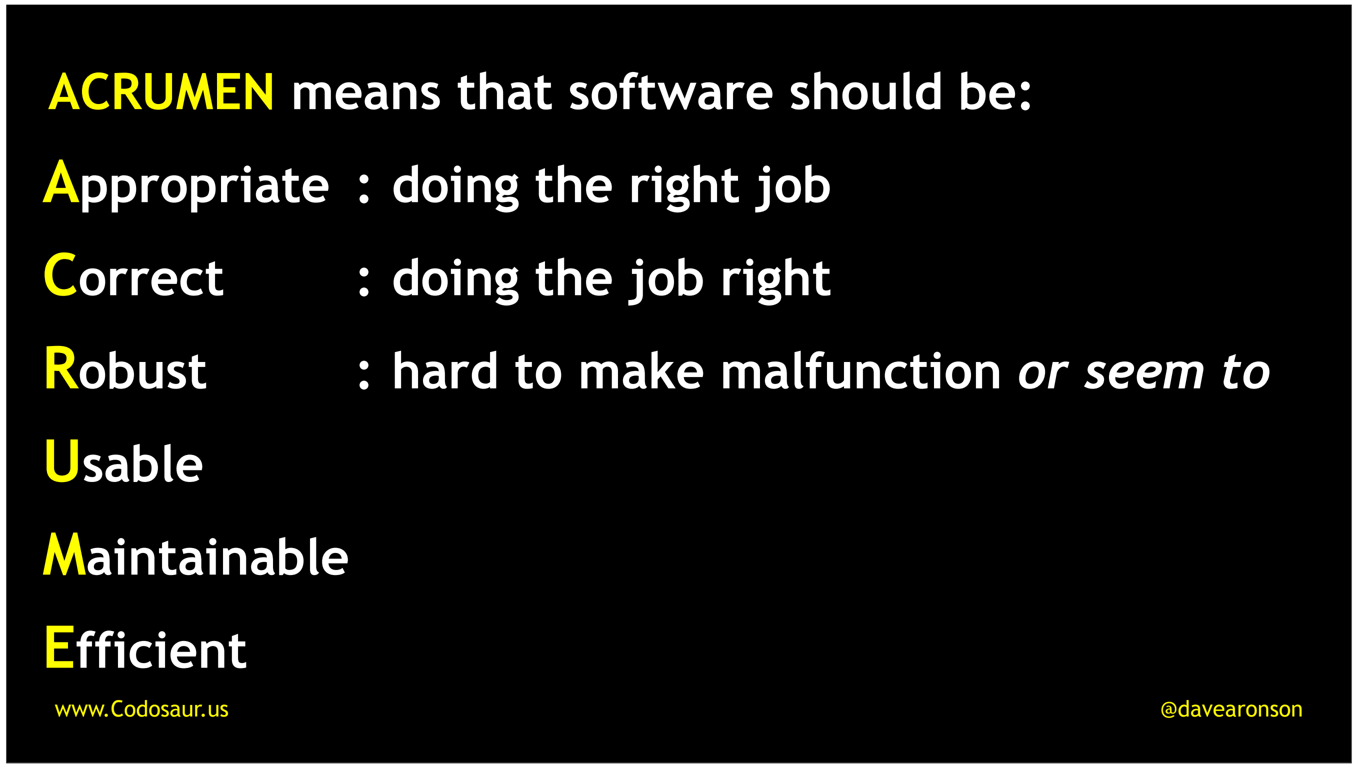
Maintainable

Efficient

www.Codosaur.us

@davearonson

. . . *doing* that job *correctly*, or in other words, doing the *job right*. It should be . . .

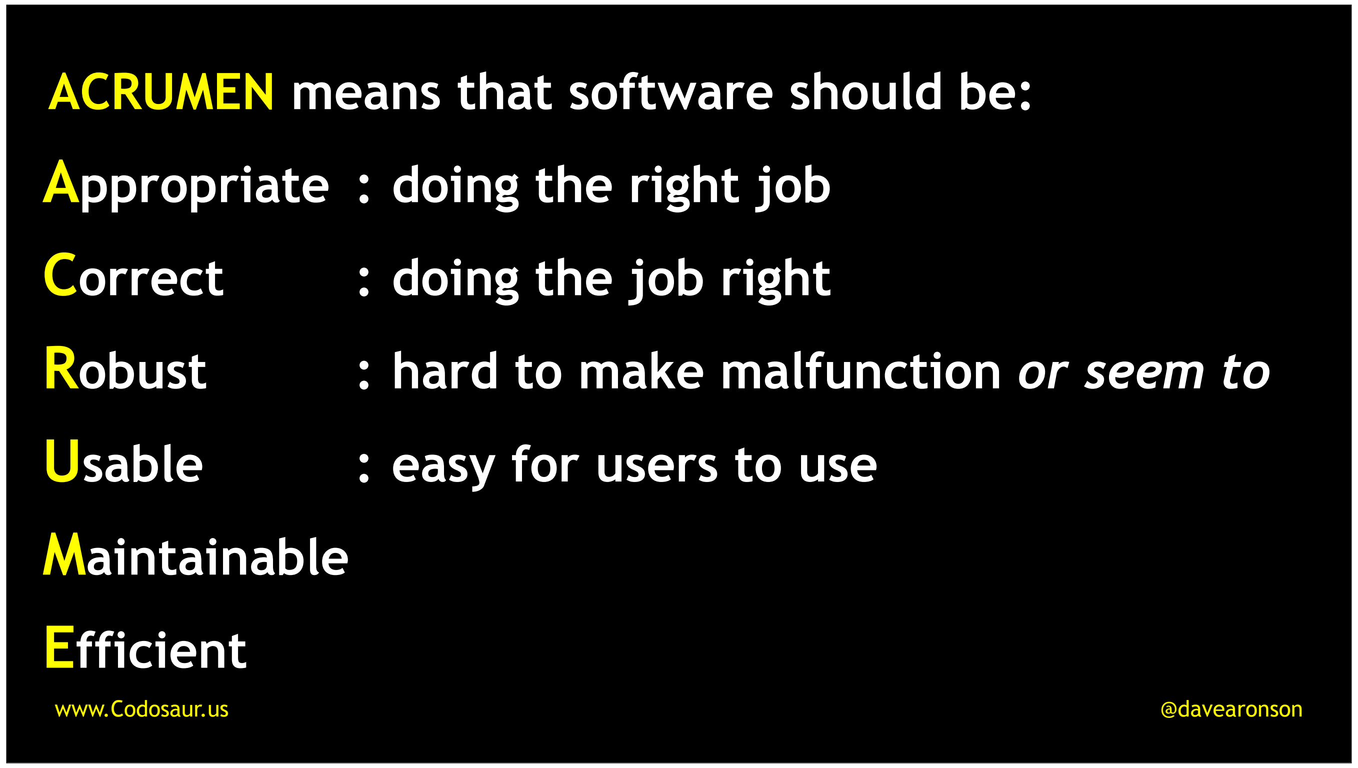


ACRUMEN means that software should be:

Appropriate	: doing the right job
Correct	: doing the job right
Robust	: hard to make malfunction *or seem to*
Usable	
Maintainable	
Efficient	

www.Codosaur.us @davearonson

. . . hard for anyone to make it malfunction, *or even seem to*, but it should be . . .

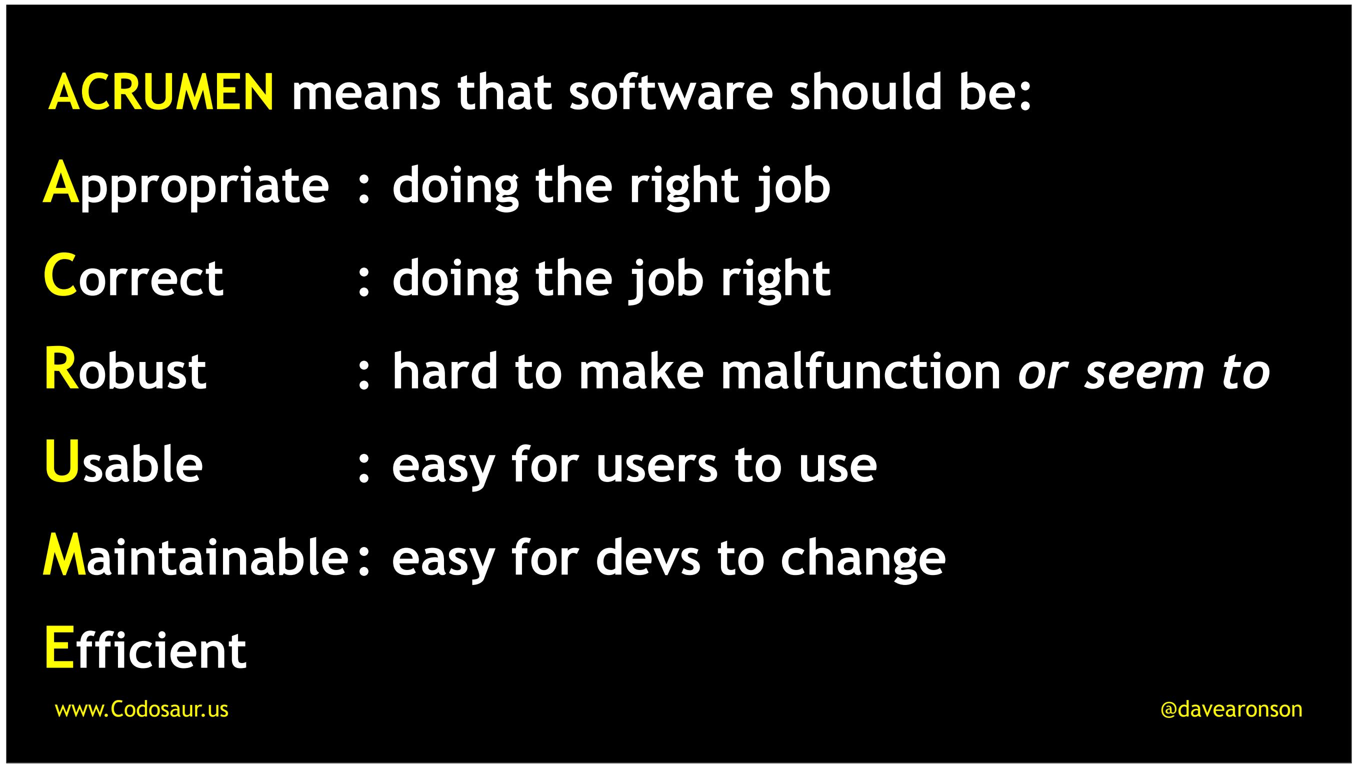


ACRUMEN means that software should be:

Appropriate	:	doing the right job
Correct	:	doing the job right
Robust	:	hard to make malfunction *or seem to*
Usable	:	easy for users to use
Maintainable		
Efficient		

www.Codosaur.us @davearonson

. . . easy for the users to use and . . .

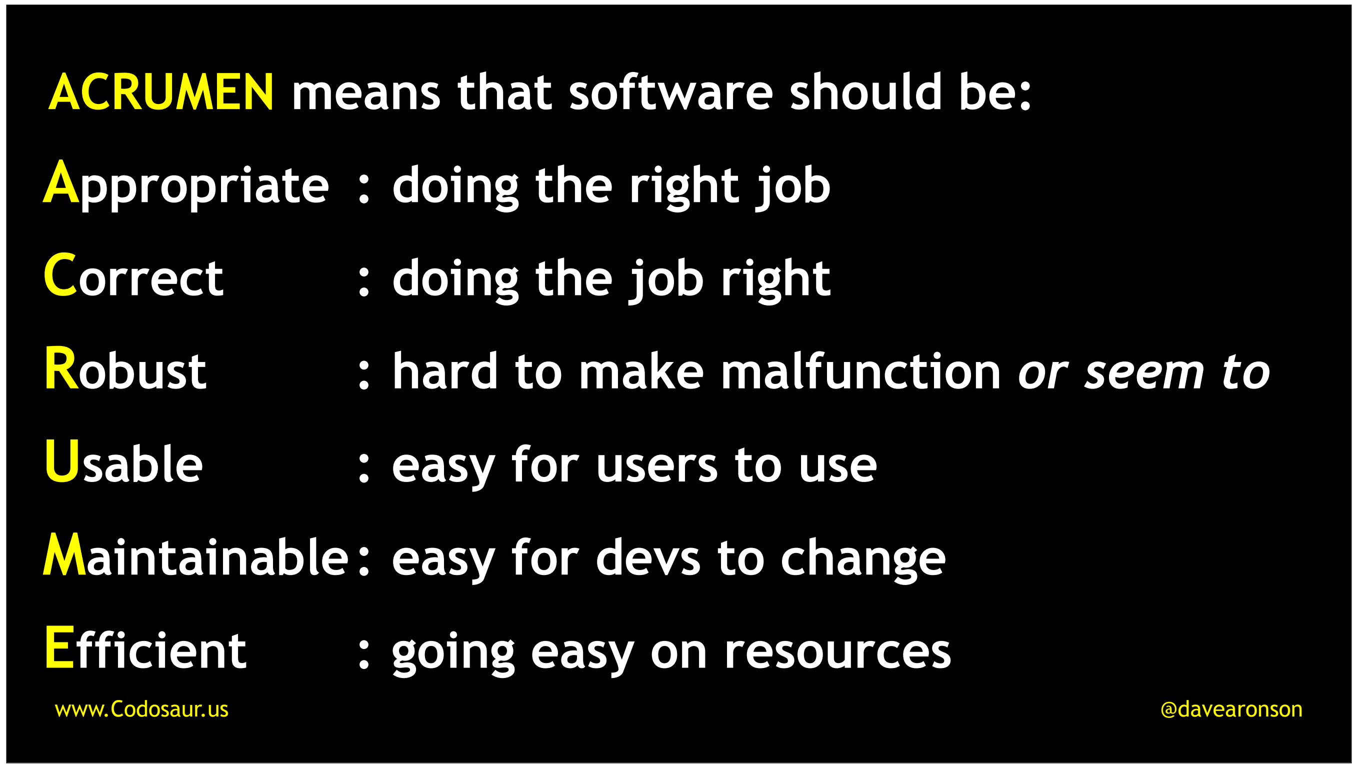


ACRUMEN means that software should be:

Appropriate	:	doing the right job
Correct	:	doing the job right
Robust	:	hard to make malfunction *or seem to*
Usable	:	easy for users to use
Maintainable	:	easy for devs to change
Efficient		

www.Codosaur.us @davearonson

. . . the developers to change. *Dead last* despite how we developers tend to worship this, it should be . . .



ACRUMEN means that software should be:

Appropriate	:	doing the right job
Correct	:	doing the job right
Robust	:	hard to make malfunction *or seem to*
Usable	:	easy for users to use
Maintainable	:	easy for devs to change
Efficient	:	going easy on resources

www.Codosaur.us @davearonson

. . . easy on resources.

So, what's the N for? Nnnnn . . .

www.Codosaur.us

@davearonson

. . . nothing! I just tacked it on to make a real word.

While all that’s fresh in our minds, I'll address one . . .

. . . frequently asked question:

aside from going into detail on the tips, how do we actually *use* ACRUMEN itself, the list of aspects?

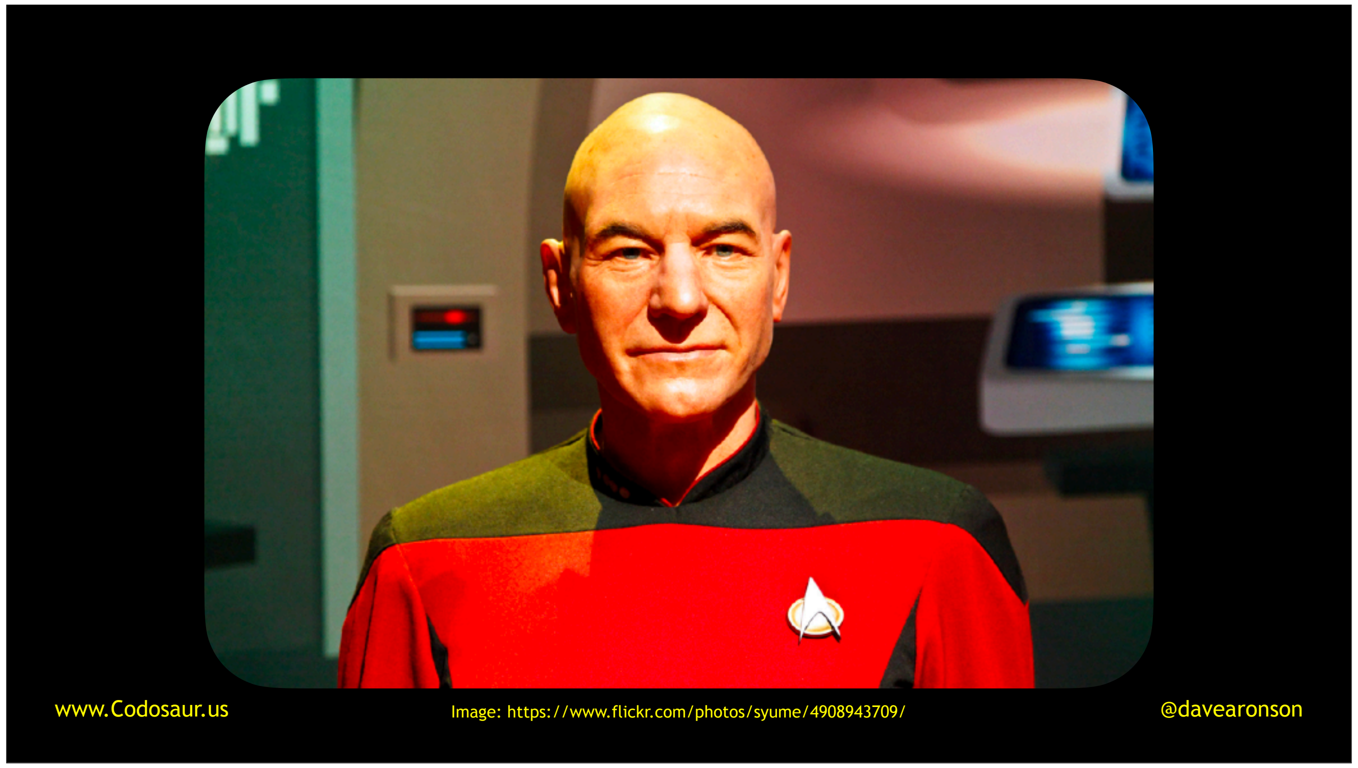
Mainly, we can keep it in mind as a . . .

www.Codosaur.us

Image: https://pixabay.com/en/checklist-list-check-check-list-911840/

@davearonson

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



. . . *make* it so?

In the short term, we can ensure that our *projects* are likely to *meet* these criteria. In the long term, we can ensure that our *processes support* these criteria, by including helpful activities, maybe even an *explicit evaluation against* these criteria. We can also set . . .

www.Codosaur.us

Image: https://www.flickr.com/photos/bensutherland/205606714

@davearonson

. . . targets, for how good we need the system to be in each aspect.

Now we'll look closer at each aspect, starting with . . .

THE FOLLOWING PREVIEW HAS BEEN APPROVED FOR
APPROPRIATE AUDIENCES
BY THE MOTION PICTURE ASSOCIATION OF AMERICA, INC.

THE FILM ADVERTISED HAS BEEN RATED
G GENERAL AUDIENCES
All Ages

www.filmratings.com www.mpaa.org

www.Codosaur.us @davearonson

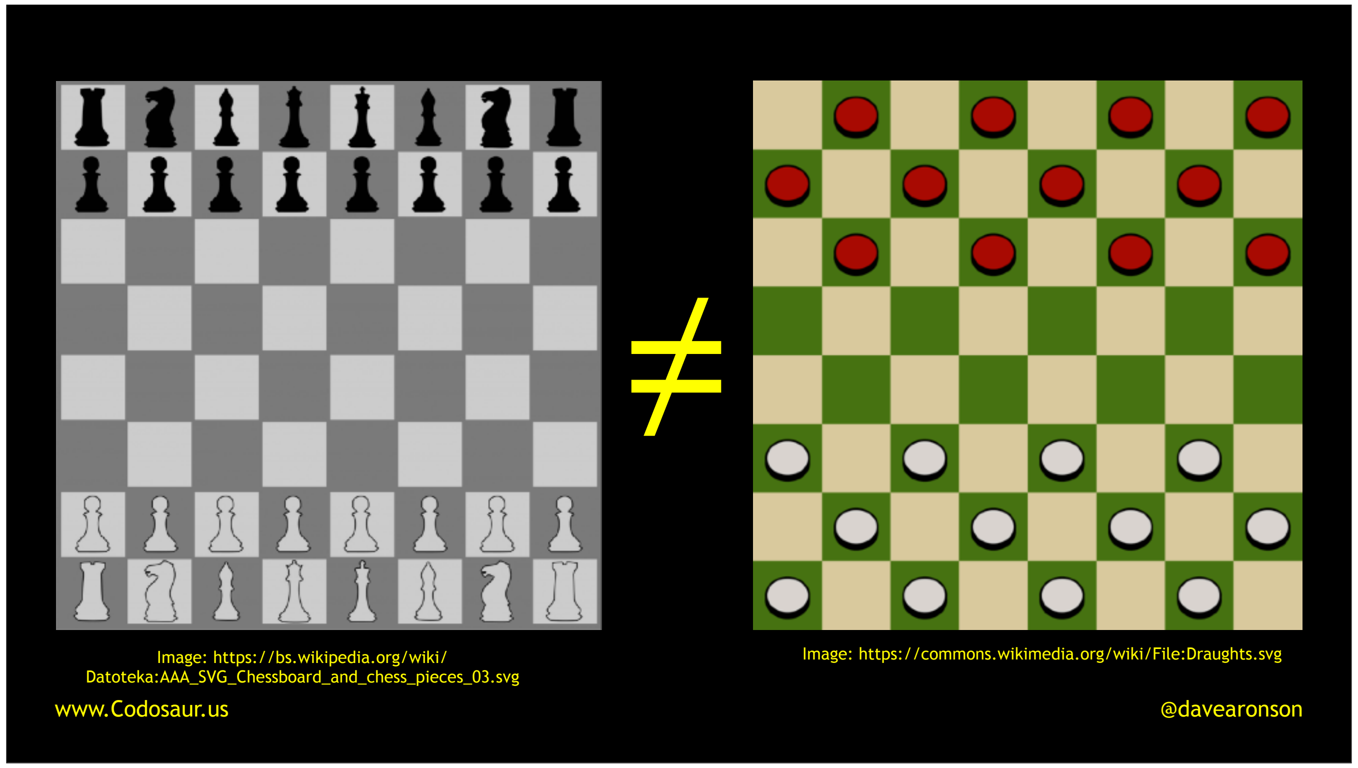
. . . Appropriateness.

If our software doesn't have this, then Nothing. Else. Matters. (PAUSE!) If our software is doing the *wrong job*, it *doesn't matter* how *well* it's doing the wrong job. So, appropriateness is not only more important than any other aspect, it's even more important than . . .

A
CRUME
www.Codosaur.us
Image: https://pixabay.com/en/balance-scale-justice-law-judge-154516/ + my text & lines
@davearonson

. . . *all* the others *put together*. And yet, we developers are generally not taught that this is even a thing, let alone one that *we* need to think about.

To *prove* this point, let's try a little thought experiment. Suppose you want a program to play . . .


≠
Image: https://bs.wikipedia.org/wiki/Datoteka:AAA_SVG_Chessboard_and_chess_pieces_03.svg
Image: https://commons.wikimedia.org/wiki/File:Draughts.svg
www.Codosaur.us
@davearonson

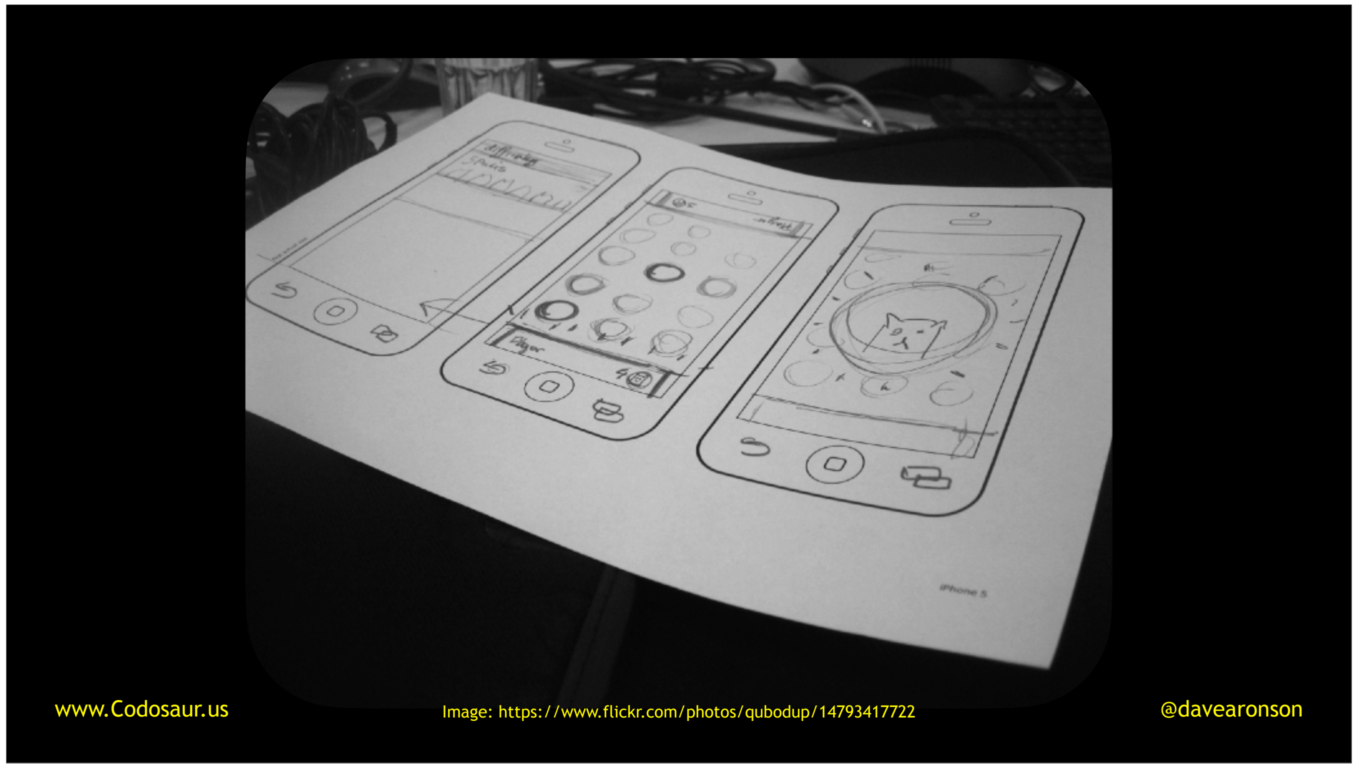
. . . *checkers*, and I write for you the world's greatest *chess* playing program. It's as correct, robust, usable, maintainable, and efficient as anyone could ever want. But you probably won't be happy with it, because . . . it's not checkers. It's not what you asked for. It's not what you *need*. In ACRUMEN terms, it's not *appropriate*.

So how do we achieve appropriateness? In an ideal world, we would have . . .

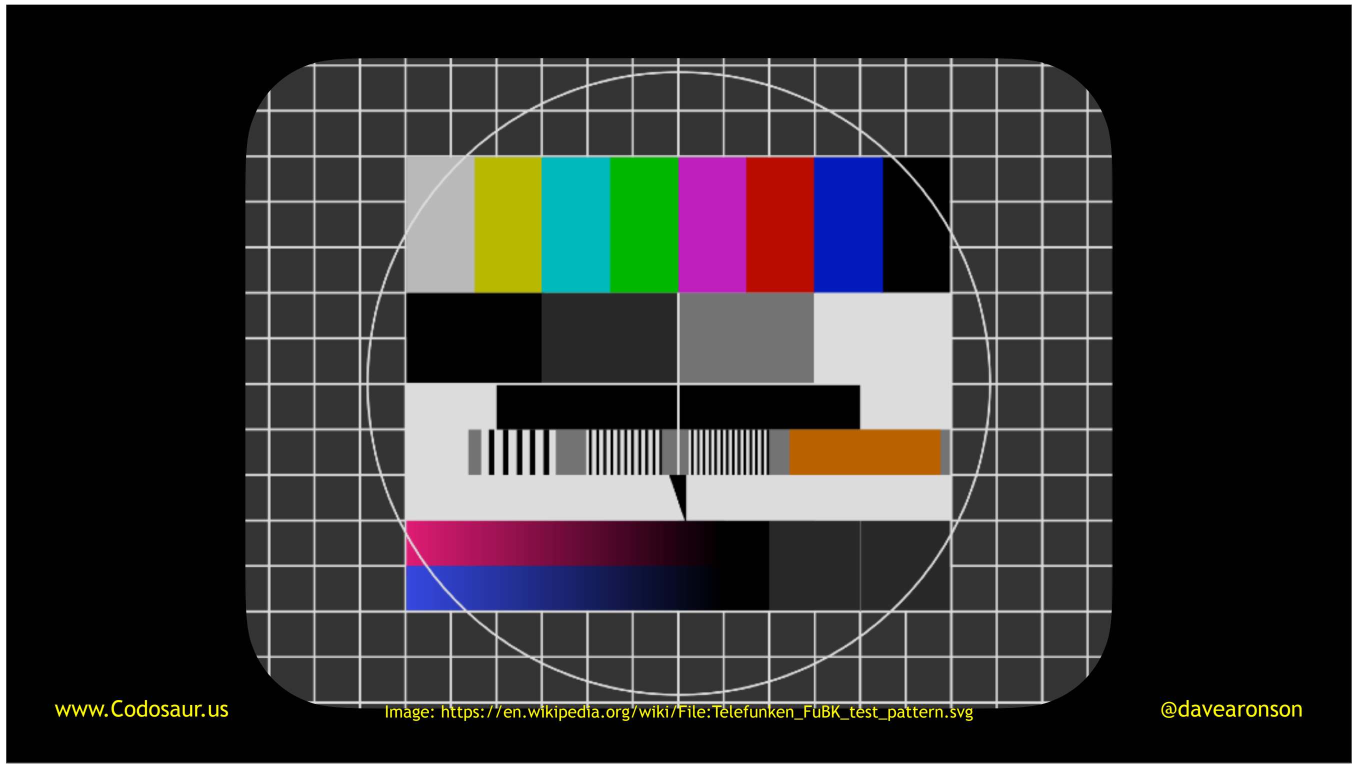
www.Codosaur.us
Image: https://www.flickr.com/photos/wocintechchat/22543243101
@davearonson

. . . frequent direct contact with the stakeholders, to ask them questions and get their feedback. Unfortunately, we don't usually get that opportunity, and often not even requirements analysts, or business analysts. So we usually have to settle for occasional remote indirect contact with a representative of some stakeholders. It doesn't work quite as well, but having *some* communication with *someone* with a clue, is *vital*.

Once we think we have a good grasp of the stakeholders' needs, we can show them, or their representatives, . . .



. . . mockups and prototypes of what we *intend* to do, and demos of what we *have* done. This gives them a chance to *correct our wrong ideas* of their needs, before we go too far down the wrong rabbit-hole. There's another thing, though, that I'll be returning to over and over in this talk. We can propose . . .


www.Codosaur.us
Image: https://en.wikipedia.org/wiki/File:Telefunken_FuBK_test_pattern.svg
@davearonson

. . . *tests!* In particular, I recommend the Given/When/Then pattern: given these preconditions, when this happens, then this is the result. This makes a great link between the worlds of business and tech, because the business people can understand it, and we can turn it into a runnable test.

Our next aspect is . . .

www.Codosaur.us

Image: https://commons.wikimedia.org/wiki/File:Correct.svg

@davearonson

. . . correctness. Nothing can actually *stop* us from *writing* code that isn't correct, at least with the tools we have today. So, the big question is: . . .

www.Codosaur.us
Image: https://zh.wikipedia.org/wiki/File:Thermoskanne(hoch,_silber).JPG
@davearonson

. . . how do we *know*? (PAUSE!)

As you probably know . . .

www.Codosaur.us Image: https://commons.wikimedia.org/wiki/File:Test_type_to_test_eyesight,_England,_1920-1960_Wellcome_L0058205.jpg @davearonson

. . . *tests* let us know whether or not our code is correct . . . *assuming* of course that the tests *themselves* are correct, but that's another story.

I'll skip over a lot of common advice about how many of what *kinds* of tests to write and how and when, but I'l point out that the usual types of tests, like unit, integration, feature, system, and so on, can only prove the correctness of cases *we thought* to test. But there are some advanced techniques that can help find cases we didn't think of, such as . . .

www.Codosaur.us

Image: https://www.flickr.com/photos/athomeinscottsdale/3279949186

@davearonson

. . . property-based testing and . . .

www.Codosaur.us

Image: https://pixabay.com/en/genetic-modification-mutant-mutation-549889/

@davearonson

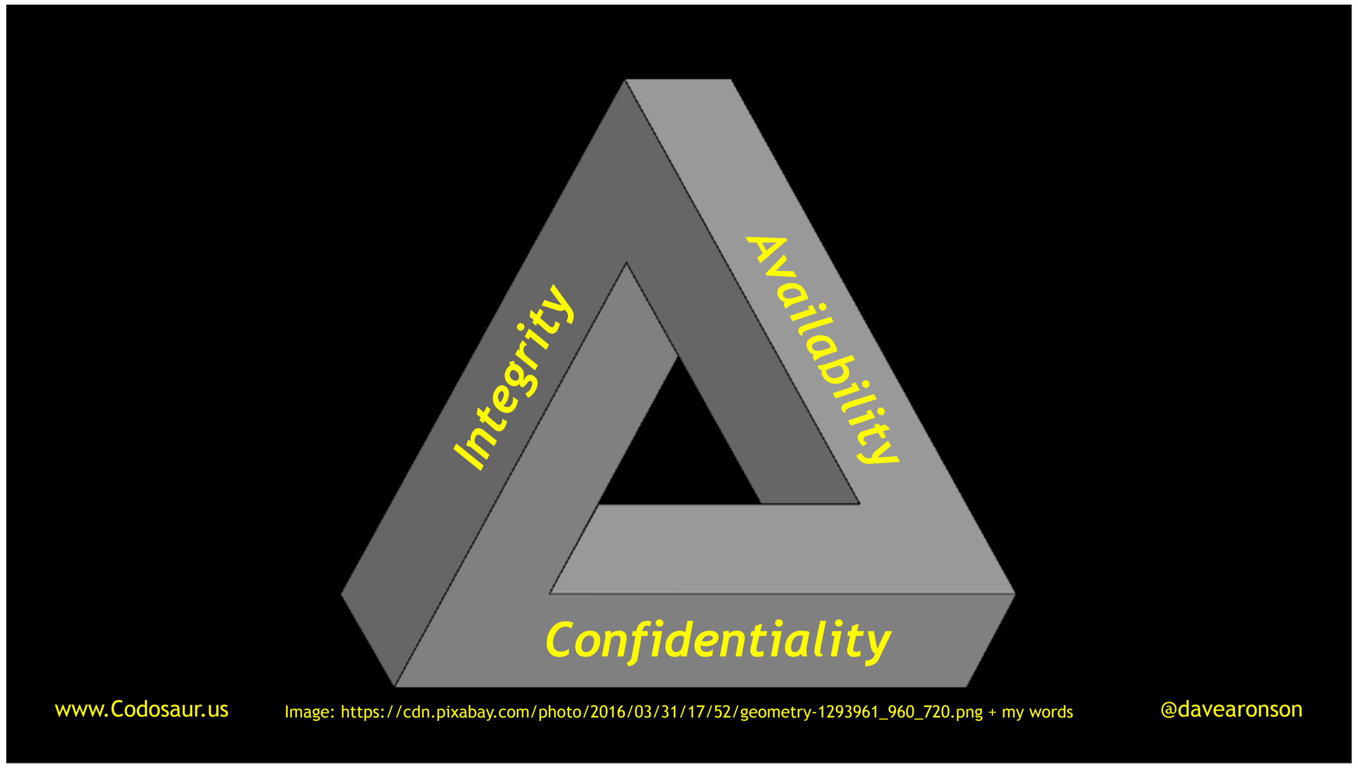
. . . mutation testing — which I'll be presenting about on Friday.

We should have enough test coverage, of assorted kinds and levels, and verified to be actually meaningful, to have strong confidence in the correctness of our code.

Next up we have . . .

ROBUST
901A
HANOMAG
www.Codosaur.us
Image: https://commons.wikimedia.org/wiki/File:Hanomag_Robust_901A_Cloppenburg.jpg
@davearonson

. . . robustness. The short explanation is that it's hard to make the software malfunction, or even *seem* to, but what does *that* mean?! There are a few other things, but most of what I mean is covered by a core concept of information security: . . .


Integrity
Availability
Confidentiality
www.Codosaur.us
Image: https://cdn.pixabay.com/photo/2016/03/31/17/52/geometry-1293961_960_720.png + my words
@davearonson

. . . the CIA Triad. No, it's nothing to do with spies and gangsters, it's this triangle up here, of Confidentiality, Integrity, and Availability. So, robust software does *NOT:*

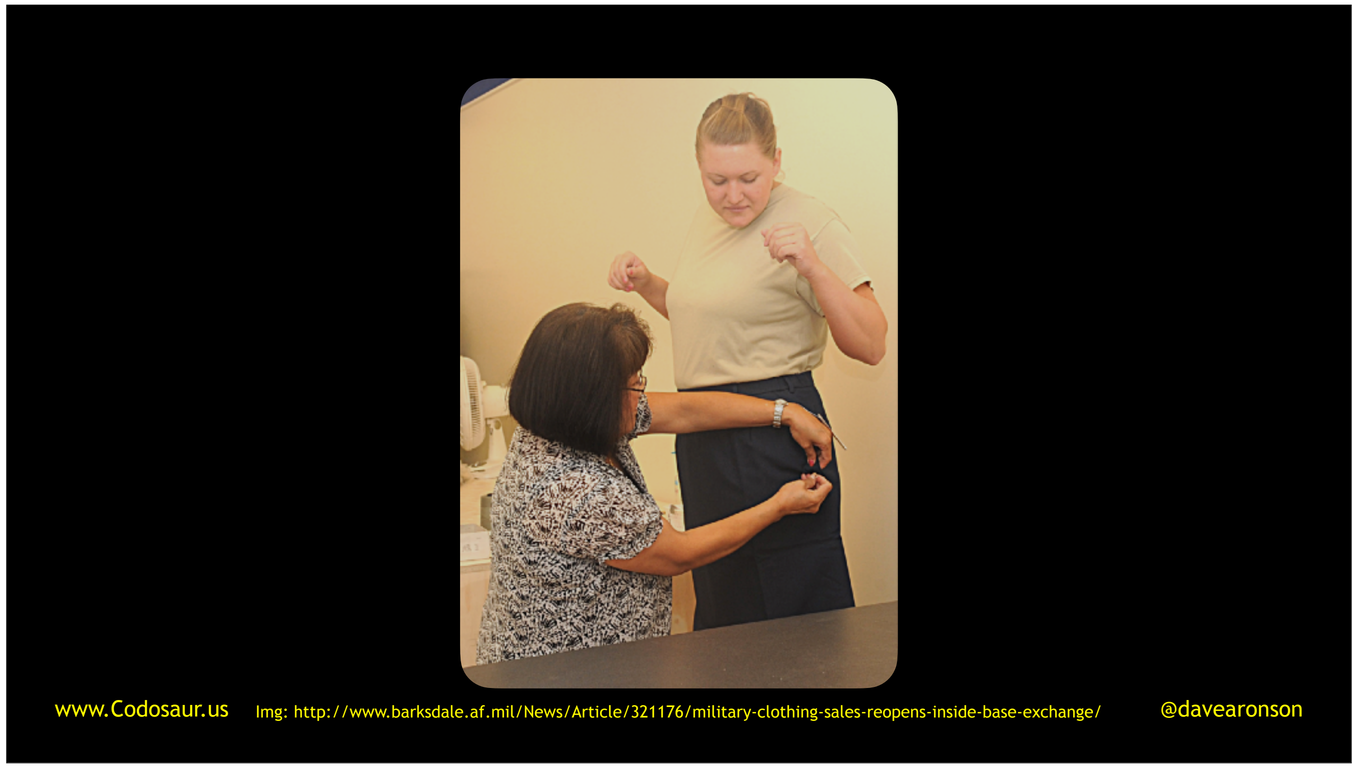
SOMEBODY BLABBED
...don't talk about ship movements !
...don't talk about war production !
BUTTON YOUR LIP !

www.Codosaur.us

Image: https://commons.wikimedia.org/wiki/File:Somebody_blabbed.jpg

@davearonson

. . . reveal data when it’s not supposed to, . . .



. . . alter data when it’s not supposed to, . . .

The page cannot be displa
The page you are looking for is curren
unavailable.
The Web site might be experien
difficulties, or you may need
your browser settings.
Cannot find server
Try again

www.Codosaur.us Image: https://pixabay.com/en/error-www-internet-calculator-101408/ @davearonson

. . . or become unavailable when it’s not supposed to, even if an attacker is trying to . . .

www.Codosaur.us
Image: https://pixabay.com/en/star-wars-darth-wader-villain-2463926/
@davearonson

. . . *force* it to do these things.

So how do we achieve all *that?*

Again, we could bring in the experts, which would be . . .

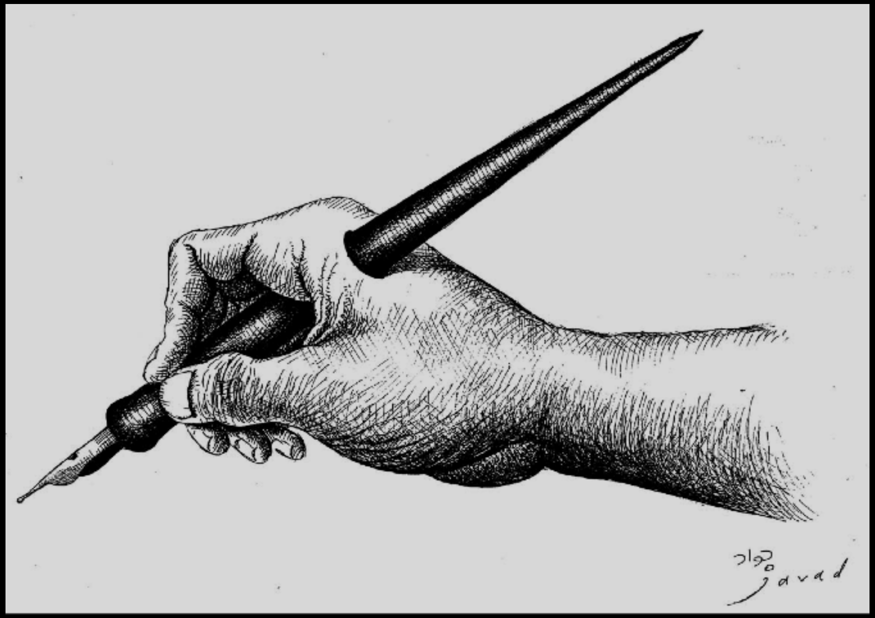


Image: https://commons.wikimedia.org/wiki/File:Javad_alizadeh_-penetrating-pen.jpg

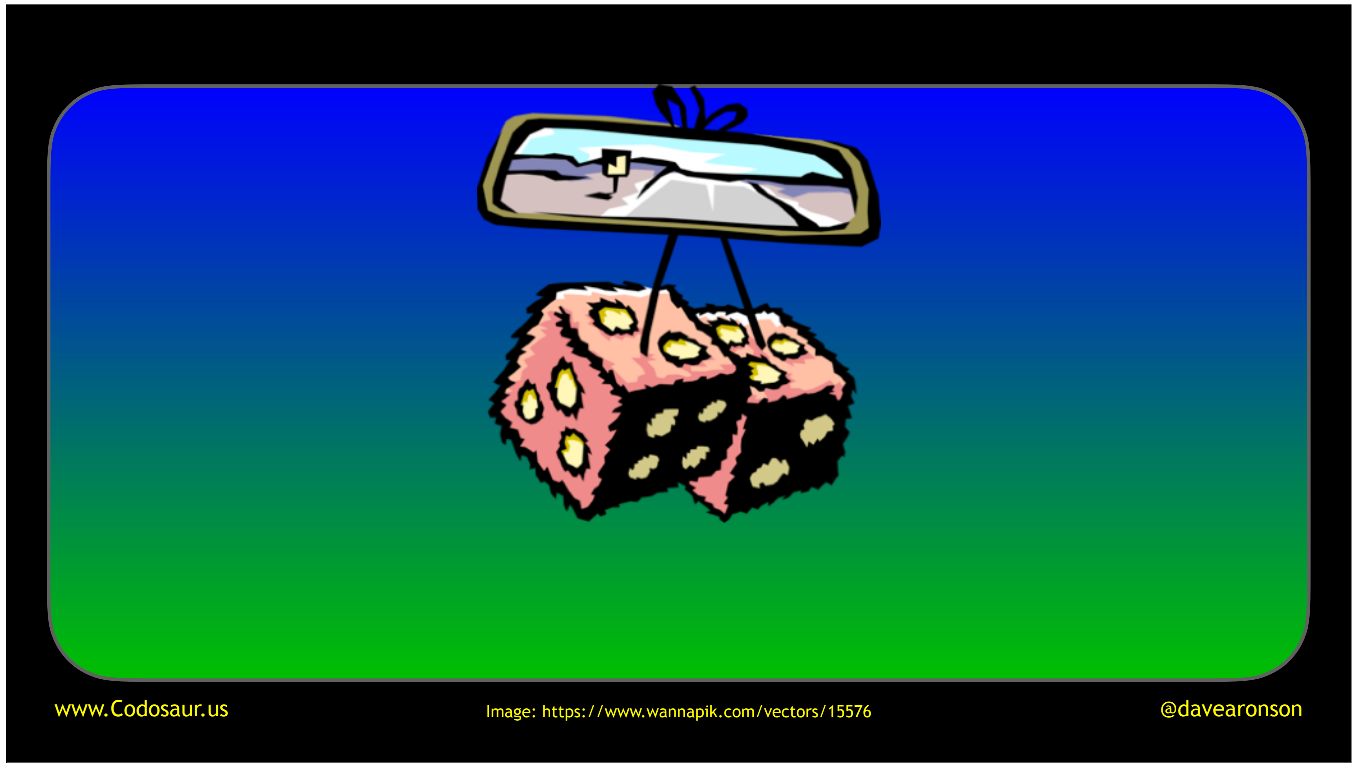
. . . penetration testers, or for short, pen testers. But, they're expensive, and disruptive, because they *need* to test the *production* system. So, again, we'll usually have to do without them. But, we can use some of their *tools*, such as software like . . .

www.Codosaur.us

Image: https://pixabay.com/en/girl-child-trampoline-blonde-212022 (CC0)

@davearonson

. . . static analyzers, . . .



www.Codosaur.us

Image: https://www.wannapik.com/vectors/15576

@davearonson

. . . fuzzers, and . . .

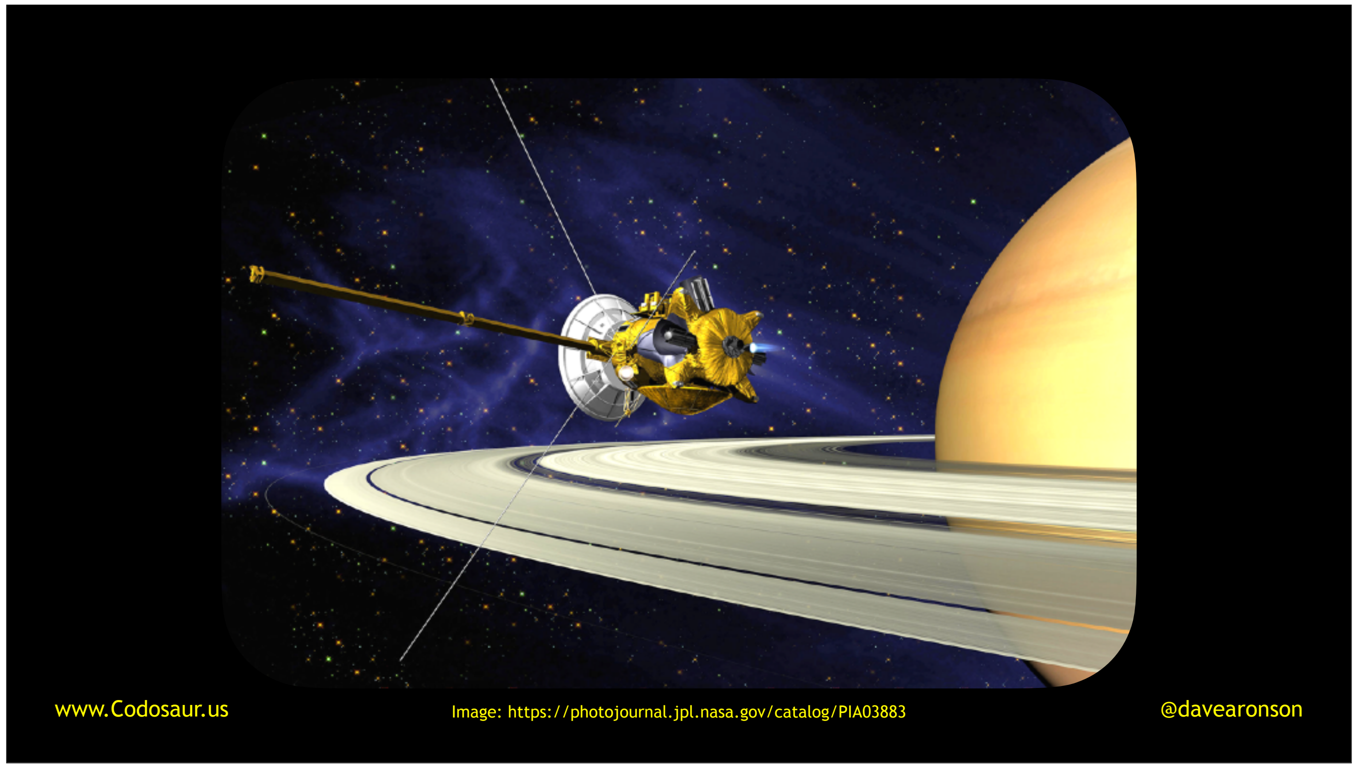
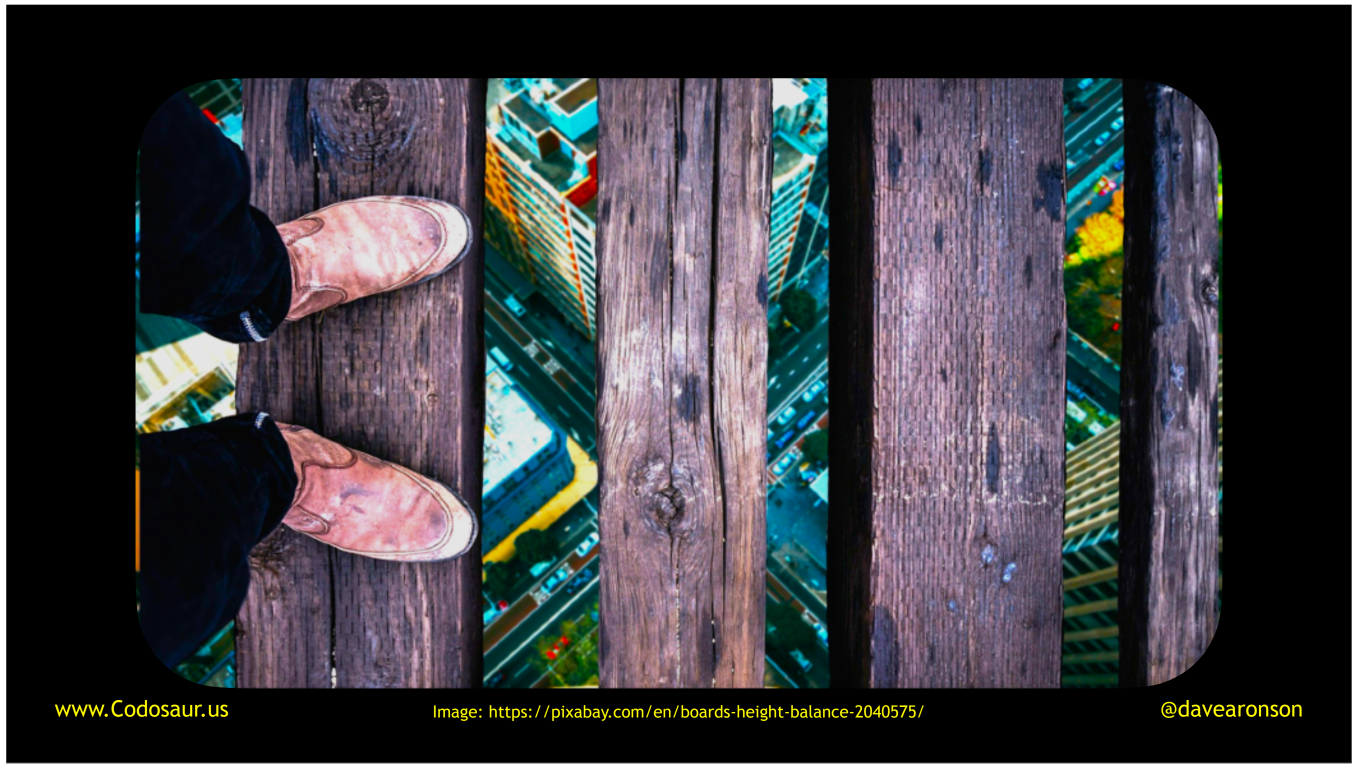

www.Codosaur.us

Image: https://photojournal.jpl.nasa.gov/catalog/PIA03883

@davearonson

. . . probes. But that's only covering *actual* fragility. What about *seeming* fragile? For that, we must ask ourselves . . .


www.Codosaur.us
Image: https://pixabay.com/en/boards-height-balance-2040575/
@davearonson

. . . what could go wrong. Our software should handle *all* reasonably foreseeable problems, from simple user error, to system problems like a full disk, and even external problems like losing a network connection, as gracefully as possible, while giving as little information as possible to potential *attackers*.

Our next aspect is one often seen as a tradeoff with security: . . .

www.Codosaur.us

Image: https://www.flickr.com/photos/slgc/24803618672

@davearonson

. . . usability. Hard-to-use software can cause headaches, or even lead the user to do the wrong thing. Remember what happened in Hawai'i in January 2018, due to software that was hard to use? They had a false alarm about an incoming nuclear missile! Just think what could happen if that were the launch system, not just an alarm!

Unfortunately, if we Google software usability, we find mostly things about ensuring that users with various challenges can use our software about as well as the rest of us. In other words, accessibility. That's a good goal in itself, but I'm adding on that it should be . . .

EASY Cheese
American
Cheddar
www.Codosaur.us
Image: https://www.flickr.com/photos/jeepersmedia/13943817487
@davearonson

. . . *easy for everyone* to use, not just *equally difficult*. Granted, the user may be facing various *challenges*. We can *start* with the ones that *accessibility* usually addresses, like low vision, color vision, hearing, or fine motor control. But there are other whole *types* of challenges we should be aware of, like lack of literacy, cultural knowledge, and even *intelligence*. Yes, we may joke about stupid users, but statistically, about half of them *will* be below average.

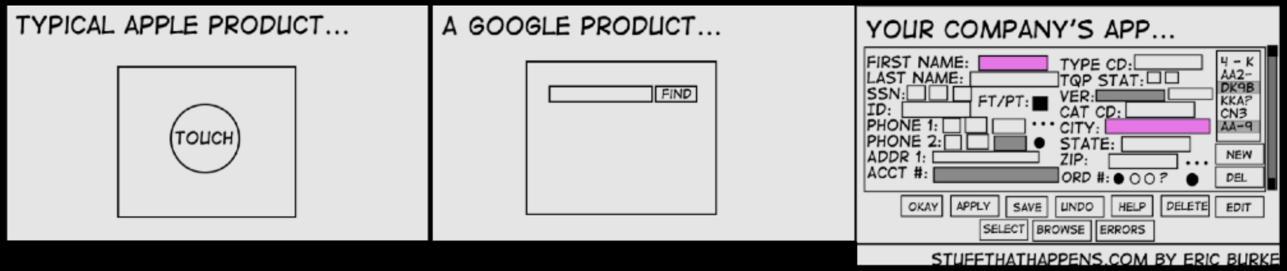
So how do we achieve all this? Once again, ideally we can bring in . . .

www.Codosaur.us

Image: https://commons.wikimedia.org/wiki/File:The_Experts_MET_DT209332.jpg

@davearonson

. . . the experts, like ideally a User Experience expert, maybe a User Interface expert, or at the very least a designer, even an old-fashioned *print* graphic designer. Again, we'll usually have to do without their help, but we can go a long way by applying their principles. For instance . . .



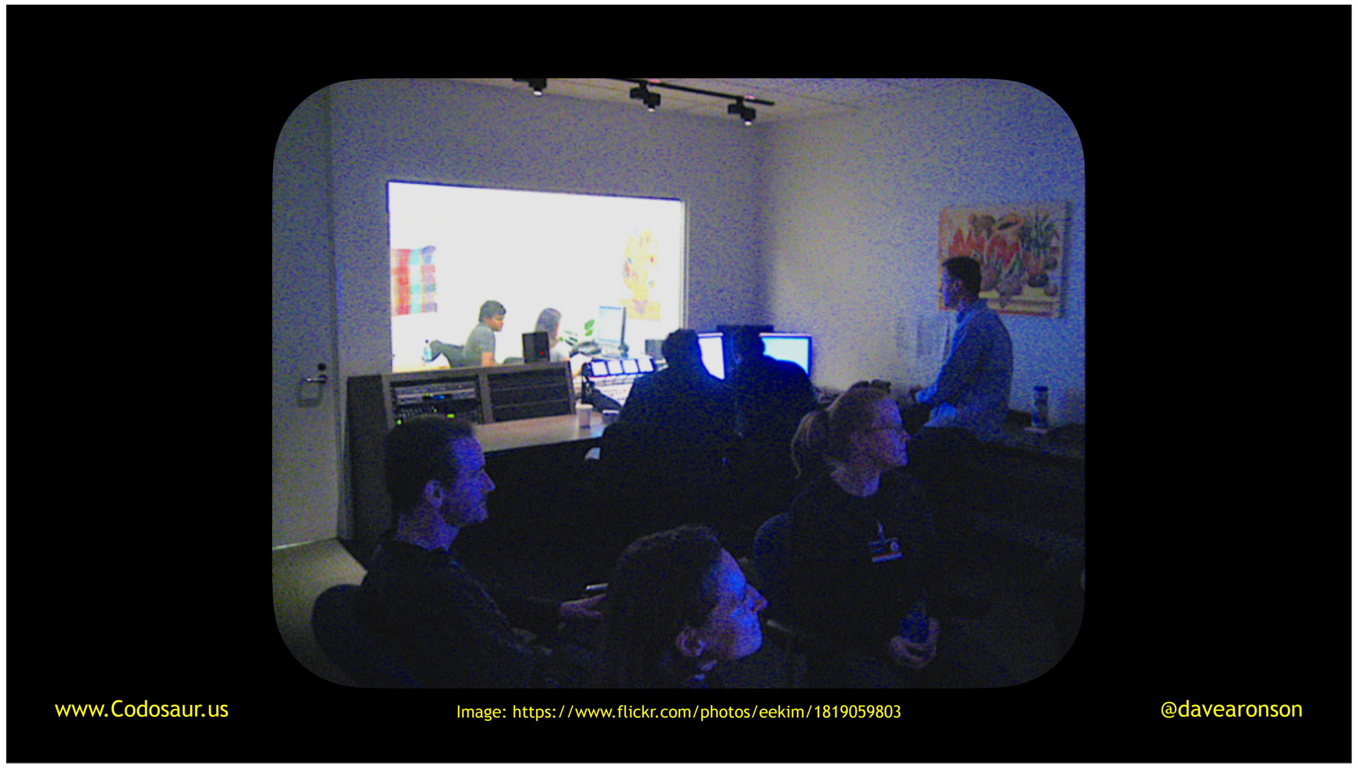
www.Codosaur.us

Image: http://stuffthathappens.com/blog/simplicity/
(awaiting permission, but Fair Use anyway due to educational purpose)

@davearonson

. . . here we see an illustration of the KISS Principle, meaning "Keep It Simple, Stupid!" I think many of us would recognize some of our own work in that cluttered mess at the end.

Also, it may not be as definable and quantifiable as correctness, but a user interface can still be . . .



. . . tested! We can watch some of our typical users use it (which is what's going on in this photo), and fix their pain points.

The next aspect is the one we usually think of most: . . .

www.Codosaur.us
Image: https://nara.getarchive.net/media/japanese-maintenance-personnel-observe-as-sergeant-robert-morris-of-the-67th-3024eb
@davearonson

. . . maintainability. We'd probably all agree that the basic concept is that "maintainable" software is easy to change. But I add that it's easy to change, *with* . . .

www.Codosaur.us

Image: https://pxhere.com/en/photo/615255

@davearonson

. . . low *chance* of error (we don't want a *dicey* situation), and . . .

www.Codosaur.us
Image: https://pixabay.com/en/potatoes-fear-horror-pot-cook-3119211/
@davearonson

. . . low *fear* of error, even for . . .

www.Codosaur.us Image: https://pxhere.com/en/photo/796734 @davearonson

. . . a novice programmer, who is . . .

www.Codosaur.us

Image: https://commons.wikimedia.org/wiki/File:New_Guy_(5895483627).jpg

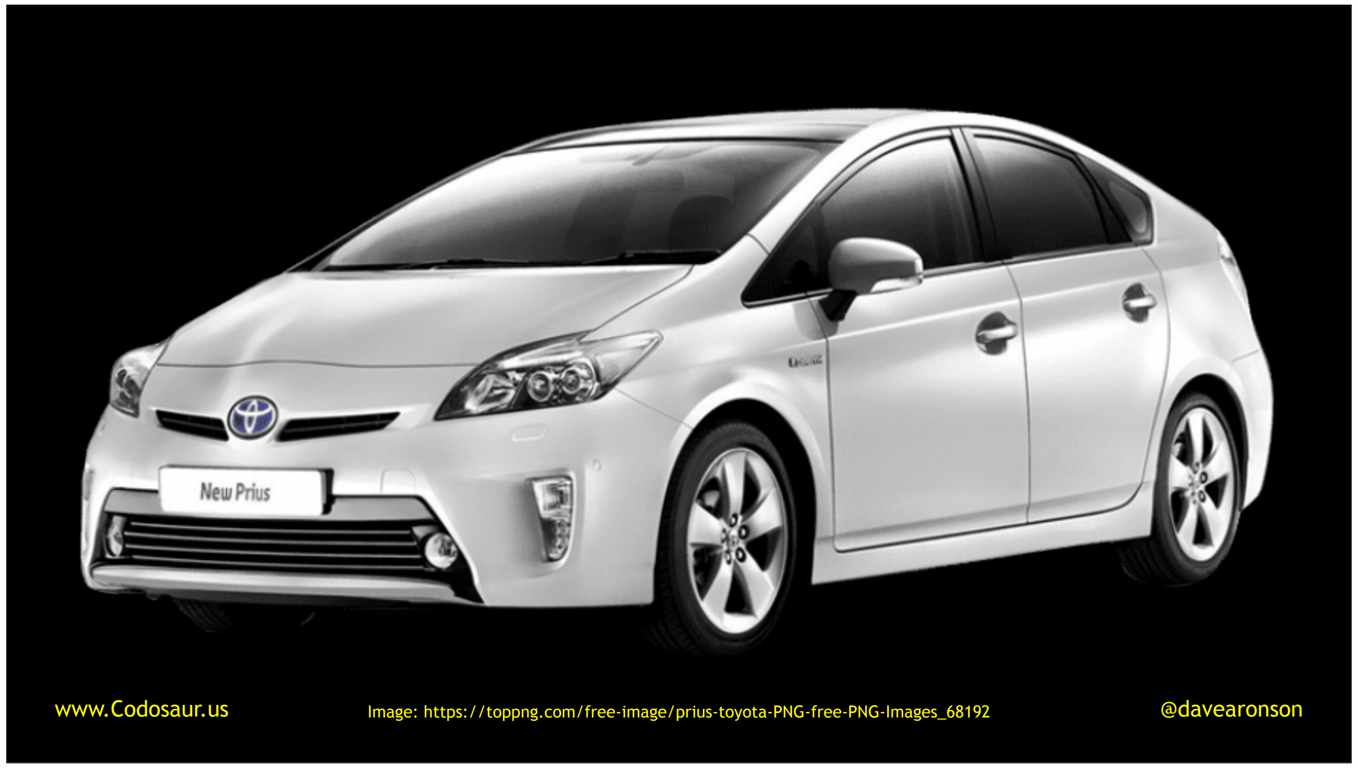
@davearonson

. . . new to our project.

Now how do we achieve all this? For better or worse, the vast majority of software engineering advice is aimed squarely at this. So, rather than expound on lots of generic principles like YAGNI and SOLID, and so on, I'm going to stick to my theme and tell you how . . .

. . . testing can help with maintainability. The *old* tests from any *previous* feature additions, bug fixes, and so on, form a *regression* test suite, to catch anything we break, that used to work. Just *knowing* that that is *there*, as a *safety net*, will reduce our *fear* of error. And *that* will allow us to progress at a quick pace with a clear and focused mind, rather than creeping along slowly and erratically because we're terrified of breaking something accidentally and not discovering it until users complain. And *that* speedup is why I mentioned fear at all.

For the final aspect, software should be . . .


New Prius
www.Codosaur.us
Image: https://toppng.com/free-image/prius-toyota-PNG-free-PNG-Images_68192
@davearonson

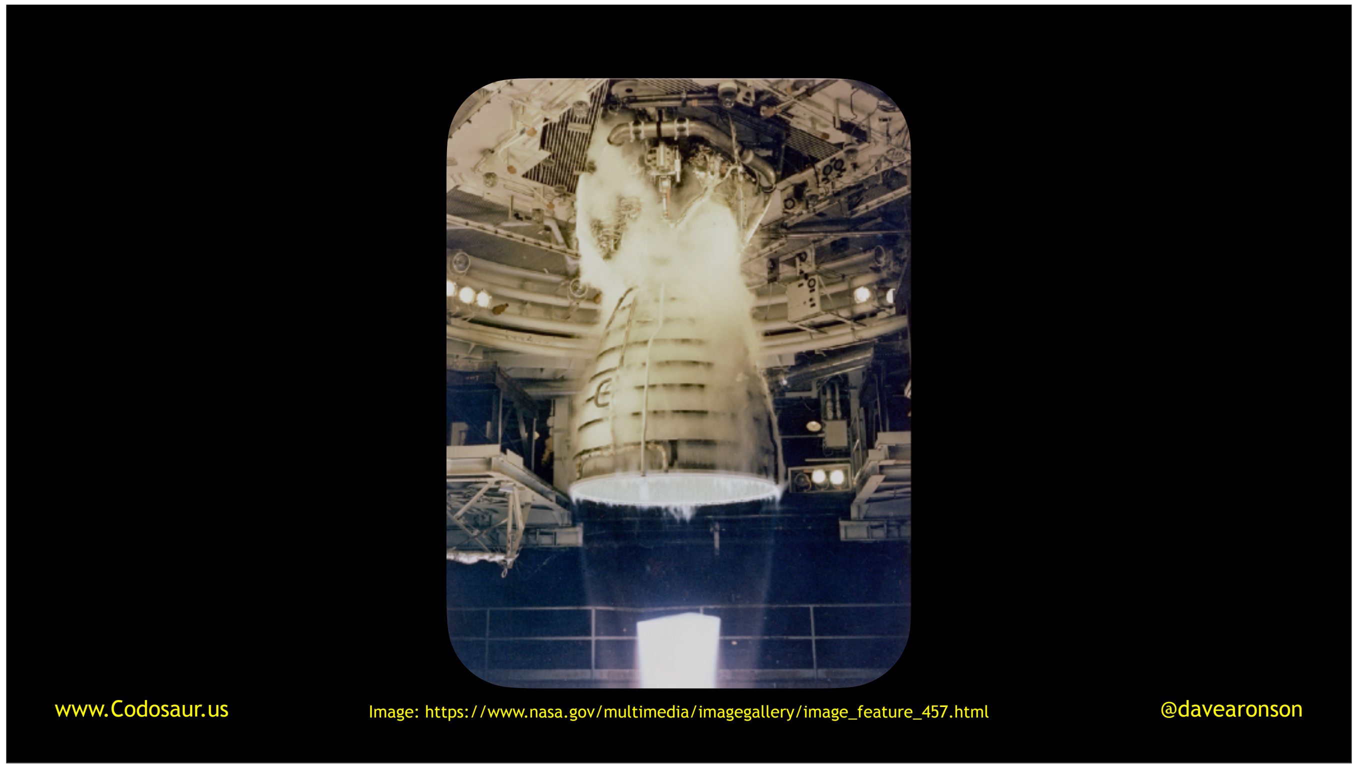
. . . efficient, in other words, go easy on resources. Mainly we know about techical resources, like CPU, RAM, bandwidth, and screen space, but there are other kinds, such as the user's *patience* and *brainpower*, and the company's *money!*

So how do we achieve efficiency? There are many kinds of resources, and many ways each can be abused, so there are many many different kinds of inefficiency, but for now I'm going to focus on fixing the most obvious and common kind: slowness.

I'm sure we've all had a program run slowly, then we stare at the code, spot where we think it's inefficient, spend a long time optimizing that little piece, run the program again, and . . . it's still slow! Right? Don't do that!

www.Codosaur.us Image: https://pixabay.com/en/diet-calorie-counter-weight-loss-695723/ @davearonson

Measure it instead! Humans aren't really very good at spotting the inefficiencies, but there are *profilers* and *traffic sniffers* and such, that will tell us *exactly* where, or at least when, we're using too much CPU, RAM, bandwidth, etc. Then we can track down the root cause, fix it, and slap a . . .



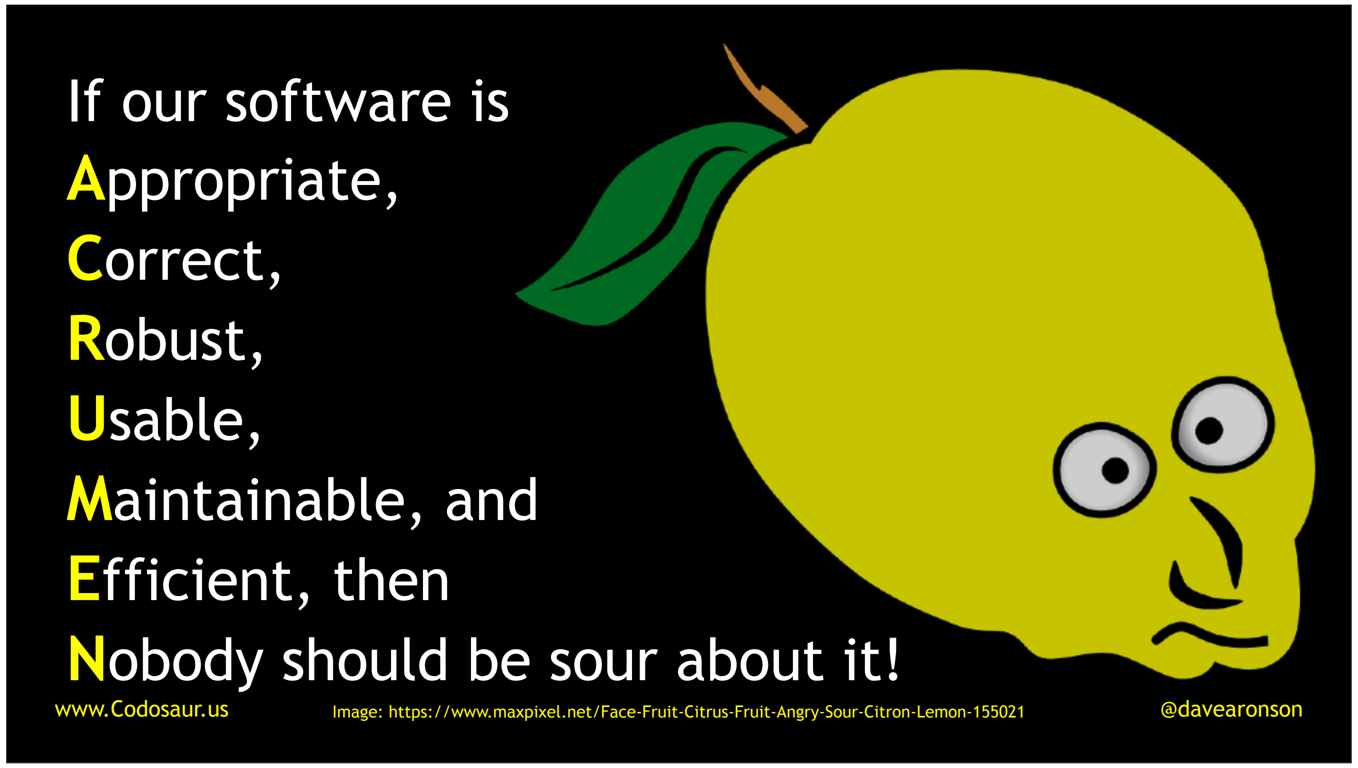
www.Codosaur.us

Image: https://www.nasa.gov/multimedia/imagegallery/image_feature_457.html

@davearonson

. . . *performance test* around it (you knew I had to mention testing eventually), to prevent that kind of regression.

In conclusion . . .

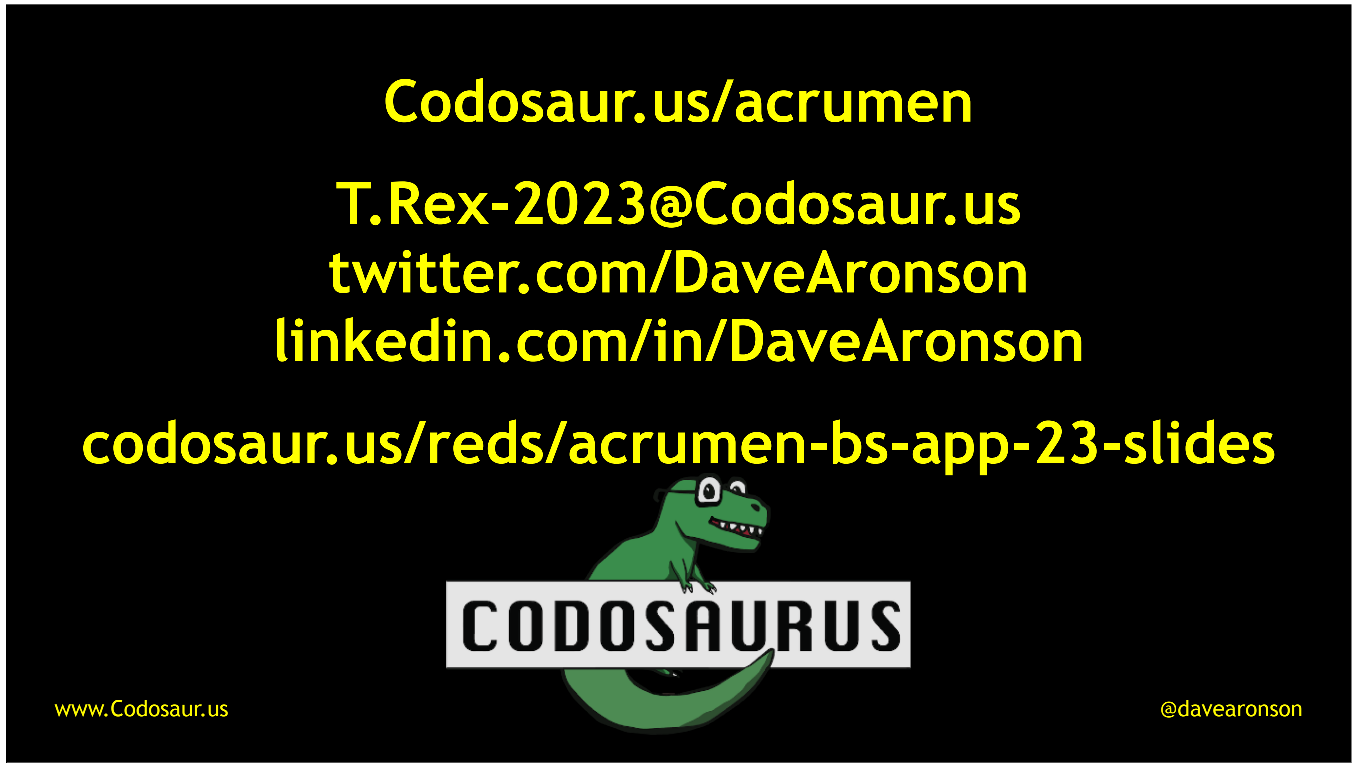


If our software is
Appropriate,
Correct,
Robust,
Usable,
Maintainable, and
Efficient, then
Nobody should be sour about it!

www.Codosaur.us Image: https://www.maxpixel.net/Face-Fruit-Citrus-Fruit-Angry-Sour-Citron-Lemon-155021 @davearonson

. . . if we 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.

And now . . .



Codosaur.us/acrumen

T.Rex-2023@Codosaur.us
twitter.com/DaveAronson
linkedin.com/in/DaveAronson

codosaur.us/reds/acrumen-bs-app-23-slides

CODOSAURUS

www.Codosaur.us

@davearonson

. . . we don't have much time for Q&A, but there's a link for more info, my assorted contact info, and the URL where the slides will be eventually, complete with a script. Now go out there and write better software, now that we can all agree what that even means.