33

What kind of special training do engineers working on mission-critical software receive? (stackoverflow.com)

submitted 2 months ago by StanislavPankevich

43 comments **share**

all 43 comments

sorted by: **best**

[–] **digitlworld** 37 points 2 months ago*

I happen to work for a company that develops software that runs on airplanes. My job function is to help create and maintain tools that assist with ensuring said software works correctly.

(This is a simplification) Per the FAA's FAR Part 21, you are required to follow rigorous steps and provide evidence that you followed those steps. To assist in this, an organization called the RTCA developed a number of documents, in coordination with experts in the industry that more specifically detail exactly how you have to do things.

The most important, industry standard, document for flight-worthy software that I'm aware of is DO-178 (which is currently in revision C). This document tells you what rigor you must follow in order to prove your software works as intended. The rigor is adjustable based on something called a Design Assurance Level, or DAL.

Basically, DALs are categories for software based on the impact failure of that software would have. For instance, "Failure may cause a crash. Error or loss of critical function required to safely fly and land aircraft." is considered Catastrophic and would require the highest level of rigor available in DO-178, Level A. Less critical software, "Failure has no impact on safety, aircraft operation, or crew workload.", receives Level E, and has the least rigor (but still has rigor). And there are levels B, C and D as well, covering ever increasing calamity if your software fails.

DO-178 covers verification and validation. To develop the product, there are a plethora of standards that you can employ. At least in the

search

this post was submitted on 16 Dec 2016

33 points (77% upvoted)

shortlink: https://redd.it/5iohue

username password

□ remember me reset password login

Submit a new link

programming

subscribe 734,072 readers

630 users here now

/r/programming is a reddit for discussion and news about computer programming

Guidelines

- Please try to keep submissions on topic and of high quality.
- Just because it has a computer in it doesn't make it programming.
- Memes and image macros are not acceptable forms of content.
- If there is no code in your link, it probably doesn't belong here.
- App demos should include code and/or architecture discussion.
- Please follow proper reddiquette.

Info

- Do you have a question? Check out /r/learnprogramming, /r/cscareerquestions, or Stack Overflow.

US, they primarily come from RTCA (the DOs), ARINC (ARINC 100-900 series documents), and MIL-STD documents (for military applications).

Some of these specifications dictate how your software must behave. For instance, my understanding (and I don't write flight-worthy software, so I might be wrong here) is that at certain DALs, you're not allowed to allocate new memory (malloc/new) at any point after an initial startup of the software. Once running, you have to have a static memory footprint.

All of this stuff is aggregated into each company's own policies and procedures. My company has their own proprietary processes for following meeting all of this. During the process, we're audited multiple times to ensure things are done correctly. It's an incredibly complex process.

I learned what I know on the job through various trainings that my company provides, through experience and through my own research. But all of that training really is intended to help you learn all of the standards and practices so that you can work day to day within them.

And this is just for flight-worthy software. There are other standards and practices for hardware, for data, for data formats, for mechanically constructed objects, etc. And that's *just* for aviation. Each major industry that has safety/mission-critical engineered products has their own set of standards and practices that you have to learn to follow.

Keep an eye on the automotive industry as autonomous cars are developed. That's probably the one safety critical industry that's in its infancy. I suspect much of the FAA stuff will be adopted/evaluated to apply there given the sheer safety of air travel, at least via the US/FAA (take a look here and see the last time a US originated airline suffered fatalities).

EDIT: Actually read the questions on SO:
Question 1: The difference, is in standards, practices, policies, procedures, and oversight that you MUST adhere to. There is plenty you can do to build good, reliable software that is not mission critical, you're just not required by regulation to do it.

- Do you have something funny to share with fellow programmers? Please take it to /r/ProgrammerHumor/.
- For posting job listings, please visit /r/forhire or /r/jobbit.
- Check out our faq. It could use some updating.
- If you're an all-star hacker (or even just beginning), why not join the discussion at /r/redditdev and steal our reddit code!

Related reddits

- /r/technology
- /r/ProgrammerTIL
- /r/learnprogramming
- /r/askprogramming
- /r/coding
- /r/compsci
- /r/dailyprogrammer
- /r/netsec
- /r/webdev
- /r/web_design
- /r/gamedev
- /r/cscareerquestions
- /r/reverseengineering
- /r/startups
- /r/techsupport

Specific languages

created by spez a community for 11 years

MODERATORS message the moderators

ketralnis
spez
Poromenos
tryx
dons
masta
kylev
chromakode
a_redditor

about moderation team »

< > discussions in r/programming X

2124 · 383 comments
AWS released a summary of the S3 issue

Question 2: I think (from my experience) that they receive that training as introductory software quality assurance classes in a software engineering program in college and then followed primarily by on the job training. Also, there are plenty of companies out there that specialize in training people to the standards in their industry. I just find that those companies are usually brought in by the company you work for, not something you go get yourself.

Question 3: This is tricky. It's industry specific, for one, but it's also pricey. Every document I mentioned above costs money. Each individual RTCA DO document costs money (for instance, RTCA charges $250 for a soft copy of DO-178C on their website). The ARINC documents will be similar. But there are also books out there that cover this specialty (for instance this). And I haven't really looked, but I'm sure there are colleges out there that have degree programs specifically for safety critical software development.

from earlier this week

permalink **embed**

[–] **McChubby007** 5 points 2 months ago

Good answer - I used to work on Avionics too. We once had to write a DO-178B Level A compliant digital clock (in Ada of course) because of its attachment to the ARINC bus - most likely the world's most expensive alarm clock.

permalink **embed** **parent**

[–] **Money_on_the_table** 2 points 2 months ago

> receives Level E, and has the least rigor (but still has rigor).

I'm actually surprised at how little rigor DAL E has. One of our supporting units is comprised of DAL C and DAL E and the DAL E part could be thrown on with very little oversight. Naturally if it failed there's no consequence to safety, but still seems strange considering how much effort goes into everything else.

permalink **embed** **parent**

[–] **digitlworld** 2 points 2 months ago

Yeah, I actually looked at the Wikipedia page after this and ate my words on that. Level E has no objectives according to the table on that page. I'd double check, but I don't have my own copy of that document.

permalink **embed** **parent**

[–] **Euphoricus** 2 points 2 months ago

How are the standards "enforced"? Are there some 3rd party or government bodies that can be employed to verify if you follow the standard? Are there any mandated checks if the standards is being properly followed or is following the standard self-imposed and only problem is that you get fined if something goes wrong? If you were to violate the standard, what is level of punishment? Would it be possible for a company to loose it's business if the violation was bad enough?

permalink **embed** **parent**

[–] **digitlworld** 1 point 2 months ago

Not sure on most of that, but the FAA is the primary oversight. They are part of the audits. Usually the punishment (or at least i think so) is grounding your product (not allowing it to fly) or delaying the release of said product. Both are incredibly expensive for your company's coffers and reputation. The other trick is that most of the avionics companies are suppliers to the plane manufacturers (e.g. Boeing). If your company screws up the process and is delayed, that also costs your customer, such as Airbus, money. The next time a contract comes around, are they going to want to do business with you again?

permalink **embed** **parent**

[–] **StanislavPankevich** [S] 1 point 2 months ago

Great answer! Thank you very much!

permalink **embed** **parent**

[–] **p1-o2** 1 point 2 months ago

Thank you so much for this detailed answer.

permalink **embed** **parent**

[–] **Money_on_the_table** 19 points 2 months ago

OOOh, something I can actually talk about.

I'm in my third year of working at a large company that creates control software for jet engines (The EEC). I've never worked in an app development business, but am learning in my spare time.

I'll try and tackle the questions. Feel free to ask further:

*1) What actually makes the difference between and iOS programmer and a programmer working on mission-critical software? *

None. We are all programmers, but the process that we follow is very different I expect. As I say, I've never worked in App development, but I expect the biggest difference is requirements, reviews and testing. To write flight-software it must adhere to DO-178, Design Assurance Level A. This dictates a lot of things and so our processes is set up to meet the standards it has set.

Our requirements are reviewed many times over from a systems, design, code and verification point of view. The language that the document uses is set in a standard, which must be complied with so that there are no ambiguities. The code is against rigid standards, no dynamic memory allocations, no pointers. The code and low-level design document are reviewed too. We write our actual code (in Ada. Some C and Assembler where needed) and also use SPARK to confirm that contracts of the specification are held to. Verification exercises a test for each requirement. We do check all the corner cases.

When we test our code, it's tested in an emulator, on the target hardware and then once on the EEC against a rig that simulates actual flight conditions (the EEC thinks it is on the engine) and induce failures to check they are handled correctly. Everything is recorded and done to standards. Everything is documented and it all traces up to show that we have met all the requirements. All tools used are qualified for use, compilation is carried out with a known compiler and switches, so we know that the code we write is what we get. The Worst execution time is calculated to make sure that the engine will be responsive to changes.

Here's the longer stage of how flight control software is written from a project perspective (i.e. new engine from customer, to new engine flying):

1. Customer Requirements are received and reviewed, clarifications made.
2. System Requirements are created, detailing what the control unit must do. This is reviewed.
3. Control Unit broken down into many software functions, each specific and where possible, simple.
4. Each specific function has a system requirement document. This is reviewed from a systems point of view, a design (code) point of view and a verification point of view.
5. The system document is then made into high-level and low-level requirements. Low-lever basically states the actual "set this value to this, loop over all of this" etc. High-level requirements abstract some of that away. This is reviewed from a systems point of view, a design (code) point of view and a verification point of view.
6. Code is implemented against requirements. Each code item is traced to a requirement, so that it's confirmed all requirements are covered. These trace up all the way to the system document.
7. Code is reviewed, to confirm the trace-up is correct and is against coding standards.
8. Verification write tests against the requirements (they do not see the code). If tests pass, great, if not a problem report is raised. This may be the requirements are ambiguous, or could be wrong code.
9. Problem report solved? Re-testing of the whole component occurs, not just the bit that was broken.
10. Full build tested on an EEC connected to a rig that simulates aircraft conditions
11. Once passed, EEC tested on an engine in a fixed facility.
12. Once passed, first-flight test, one new engine, 3 known certified engines.
13. Once passed, flight testing of software.

2) Where do the mission-critical software programmers receive their training and what do they study?

I studied at a standard University, no specialist training. Once in the company, I was taught some things in seminars, some on the job training, a lot of oversight of the work I did to confirm it was to standards. And everything is reviewed, so you learn all the time.

I had to learn Ada, SPARK, Misra C, the coding standards, the requirements standards. But not all at the same time. It depends on the job I was doing. When I was writing system requirements, I learnt those standards. If I was writing actual flight software, I'd learn the Ada and SPARK. There are various tools that we run that can check the code and point out errors too.

3) Any advice on where/how an application programmer (iOS, web) could learn more about the best programming practices used in mission-critical software?

Are you wanting to learn to apply those principals to iOS and web? Or to move into doing mission-critical? If it's the former, have clear requirements, error-checking on everything and keep code in short, easily testable components. If it is the latter, that will vary depending on where you work, what their standards are and what they use. But doing the error checking and requirements stuff will be good on a portfolio.

permalink **embed**

[–] **Wolfspaw** 10 points 2 months ago

What are your thoughs on Rust? It seems to be another language that takes safety very seriously.

permalink embed parent

[–] Money_on_the_table 7 points 2 months ago

Im afraid I know nothing about it.

permalink embed parent

[–] Wolfspaw 5 points 2 months ago

Thanks for your insights on safety-critical development!

I asked about Rust because it's one of those new languages that prioritizes safety.

permalink embed parent

[–] Money_on_the_table 7 points 2 months ago

I think new languages and safety critical don't really go hand in hand.

Even with Ada 95, we had a full study on the compiler, that confirmed how we were using it, and what options to use. If any of those flags were to be changed, it'd be a huge new study we would need to fund.

I think Rust would need to prove itself for a while before it could be used. For now, any C code has to be MISRA compliant, or justified in cases where we can't/won't for whatever reason.

permalink embed parent

[–] myrrlyn 3 points 2 months ago

I'm working on getting Rust's toe in the door, as it were, in aerospace, but I don't see any significant progress happening for semicolon years. Fortunately I'm hoping to be in it for the long haul.

permalink embed parent

[–] Wolfspaw 2 points 2 months ago

true, good points!

permalink embed parent

[–] ellicottvilleny 2 points 2 months ago

Maybe it does. But in aviation/avionics and military applications it will be many years before anything but C, and ADA is acceptable.

permalink embed parent

[–] mmstick 4 points 2 months ago*

Basically, the safety guarantees of Ada also apply to Rust, but Rust goes further, most notably with an ownership model that ensures and prevents a wide range of additional issues.

First, there's the compile-time borrow checking. You may only mutably borrow a variable once at a time. You may immutably borrow a variable any number of times. You may not mutably borrow while it is already immutably borrowed, and vice versa. You may not mutably borrow more than once a time.

Then there's the borrowing and ownership mechanic. A variable that is borrowed by `self`, and does not implement the `Copy` trait, will transfer ownership to the function/method that it is passed into, and will thus be dropped at the end of that function/method call. The compiler will warn you when you try to re-use a variable that had it's ownership transferred because that variable no longer exists. In contrast, borrowing a variable by `&self` or `&mut self` will not transfer ownership, so the variable can be re-used again. Types that implement `Copy` are basically small data structures and primitives that are cheaper to copy than to reference: integers, floats, and booleans for example.

Rust also does not use null pointers but favors an `Option` and `Result` enum, and it's considered idiomatic to always return a `Result` when there's a possibility for an error to occur, or an `Option` when there's a possibility for nothing to return.

Error handling is pretty convenient now in Rust. Say you want to perform an action and that action might cause an error, and if an error occurs you want to immediately return that error. You can use the `?` operator to tell the compiler that an error should be returned if the result is `Result::Err()`, otherwise unwrap the value if the result is `Result::Ok()`. It looks something like this:

```
let mut file = File::open(path)?;
```

You may mutate the error types with convenient methods provided by `Option` and `Result`.

```
let mut file = File::open(path).map_err(CustomErr::UnableToOpenFile)?;
let next_value = iterator.next().ok_or(CustomErr::NoValueProvided)?;
```

There's also some convenient libraries for chaining errors, which is a bit too complicated to explain here, but basically you can generate very detailed error messages by chaining error messages as they propagate back. It looks something like:

```
do_something().chain_err(|| "something went wrong")?;
```

Couple with the fact that Rust supports and promotes test-driven development and benchmarking tests within the `cargo` build tool, and you have a pretty strong language to offer for critical software scenarios. The above happens to prevent a large number of critical bugs in both single-threaded and concurrent software designs. It does a pretty good job of shaping up an inexperienced programmer to write correct code all the time. Makes Rust an ideal replacement for C in everything from embedded software to kernels to desktop applications and now web applications via WebAssembly.

As a final note, RedoxOS's ralloc memory allocator, written in Rust, would also help quite a bit for security and mission-critical scenarios. It allows for more flexibility than current allocators, logging, higher security via zeroing, and checking if an allocation failed.

permalink **embed** **parent**

[–] **_mean_** 3 points 2 months ago

Now we just need to strengthen the LLVM infrastructure to add embedded targets.

permalink **embed** **parent**

[–] **myrrlyn** 2 points 2 months ago

There's a PR on my behalf for Rust to target SPARC, and I'm excited to see it land.

permalink **embed** **parent**

[–] **Wolfspaw** 1 point 2 months ago

Good summary of Rust strengths for safety!

permalink **embed** **parent**

[–] **ykechan** 48 points 2 months ago

> Would you fly on a plane if you knew that it is software is 100% Javascript?

No.

permalink **embed**

[–] **XANi_** 13 points 2 months ago

It would be a race, which crashes first, software or the plane

permalink **embed** **parent**

[–] **HakunaMatado** 15 points 2 months ago

The answer will be jquery.

permalink **embed** **parent**

[–] **Kissaki0** 2 points 2 months ago

But JavaScript does not crash! I just becomes unresponsive!

permalink **embed** **parent**

[–] **XANi_** 2 points 2 months ago

Just like programs do not run out memory, you just have too little of it

permalink **embed** **parent**

[–] **popcorp** 12 points 2 months ago

Take a look at the JPL coding standard and MISRA. These are foundations for a safe automotive/aerospace industry code. What is funny, if you follow 'good practices' for C/C++ you are already covering most of the requirements there.

I'd say the difference is not a training, but diligence and a conscious choice of not making compromises when designing the application.

I've been coding software for the air traffic control for some years and I do apply these principles with every code I write now. It makes the app better in all aspects, and it's not a hassle anymore.

permalink **embed**

[–] **1101_debian** 14 points 2 months ago

Pity that such an interesting question is being closed on SO.

permalink **embed**

[–] **Miserable_Fuck** 15 points 2 months ago

> ~~Pity~~ Typical that such an interesting question is being closed on SO.

Ftfy

permalink **embed** **parent**

[–] **DysFunctionalProgram** 12 points 2 months ago

I know it is fun to hate them but really if they were not so strict they would be flooded. We, the end users, would be the ones suffering because we would have to wade through millions of useless posts to get to what we want. This question is not fit for stack overflow, which is focused on programming and the act of writing software. This should be asked in the "software engineering" stack exchange instead as it is more about general career advice and not about actually writing software.

permalink **embed** **parent**

[–] **Dickferret** 1 point 2 months ago

We are already treated to selected questions and stack overflow is already a desert of stupid questions.

permalink **embed** **parent**

[–] **StanislavPankevich** [S] 9 points 2 months ago

The topic has been closed on SO, so this is an attempt to get some answer here.

permalink **embed**

[–] **DysFunctionalProgram** 7 points 2 months ago

It's not really a good fit for SO anyways. Ask it in one of the sister stack exchange sites such as "software engineering"

permalink **embed** **parent**

[–] **ellicottvilleny** 1 point 2 months ago

Nope. Not even over there do they want your giant three-sub-question opinion questions.

permalink **embed** **parent**

[–] **BoxOfNotGoodery** 6 points 2 months ago

Nasa: http://sdtimes.com/nasas-10-rules-developing-safety-critical-code/ and http://pixelscommander.com/wp-content/uploads/2014/12/P10.pdf

I've also been involved in a number of projects that I can't describe too much, however, what I experienced was much more about proof that proper logical and physical systems were in place.

Proof was largely determined by audits and ensuring certain process and procedures were in place.

My experience is all from a contracted point of view, I've never actually worked on a government project, while being directly employed by the government.

From that perspective, there was no "special training".

permalink **embed**

[–] **WallStProg** 6 points 2 months ago

Here's one: http://www.stroustrup.com/JSF-AV-rules.pdf

This is the spec for F-35 software, written by someone you may have heard of.

permalink **embed**

[–] **s-expression** 5 points 2 months ago

The JSF software crashes, a lot, and has been the longest schedule antagonist for the past few years.

If you're in flight for longer than an hour you have to reboot several software systems because they just stop working.

I don't particularly trust those guidelines. C++ is a terrible language for safety-critical applications.

permalink **embed** **parent**

[–] **WallStProg** 2 points 2 months ago

Well, from reading between the lines of the news reports, it sounds more like it's the "supply-chain" software that is the problem with the F-35, not the aircraft software. I could be wrong -- I'd be curious if anyone knows of more detailed info on that.

But if not C++, then what? Any GC language would have to be right out -- aircraft software would be hard real-time, and so must be completely deterministic in terms of latency. Which is why, by the way, the JSF guidelines prohibit memory allocations except for once at startup.

The only other choice I can think of is C, and C++ is, well, a better C -- at least if used properly. I wrote about that a bit here: http://btorpey.github.io/blog/2014/09/23/into-the-void/

permalink **embed** **parent**

[–] **Leandros99** 3 points 2 months ago

First of all, most of the "mission critical" software is written in C (or, very few, C++).

There are standards dictating you what language features to what extent you're allowed to use, for example MISRA-C (for the automobile industry), the NASA Programming Guidelines or the Cert Secure Coding Standard.

Apart from that, testing, testing, testing. Did I already mentioned testing?

permalink **embed**

[–] **DrunkMc** 2 points 2 months ago

There is typically a lot of auto-mated testing, requirements and documentation required, but there is no real training.

permalink **embed**

[–] **BrayanIbirguengoitia** 2 points 2 months ago

Many of the comments and their sources talk about writing simpler, more predictable code. So, is multi-thread programming completely out of the window here? Or do these standards allow it with certain conditions?

permalink **embed**

[–] **miscjunk** 1 point 2 months ago

Others have already commented on the various software/firmware coding standards out there, so I'll space the doc numbers ...

There are formal trainings that are useful for mission critical software development:

- System engineering, requirements management, FMEA (failure mode effects analysis), and other formal design related processes
- DFM/T (design for manufacturability, design for test)

Beyond that, there's a lot more code review and integration testing (typically lasting longer than the core development process itself).

permalink **embed**

[–] **jwall013** 1 point 2 months ago

This is actually currently the subject of my research. Really, it's a combination of in-depth human analysis and machine-assisted analysis. If we're talking about software specifically, some potential documents to look at are Software Failure Modes, Effects, and Criticality Analysis (SFMEA), Commonality Analysis (for software product lines), and specifications in languages like PVS.

permalink **embed**

[–] **skulgnome** 1 point 2 months ago

None. It's all in the process, or so they hope.

permalink **embed**