Open Source Summit 2021

Planning for and avoiding failures

Slides: http://marc.merlins.org/linux/talks/Failure_Planning/

Questions: please wait until the end (many slides to go through)
Slides contain more content that the talk will have time to go into.

Marc MERLIN
merlin@google.com

Those who don’t learn from history are doomed to repeat it

- How often are we repeating the same programming mistakes?
- How often are sysadmins taking the mistaken shortcuts?
- Do you wish for “I should have known better” before the problem happened?
- What can we learn from the medical and aviation community?
- Never again will you walk by any “it this right?”, or “looks like I fixed the symptom, hopefully the problem won’t happen again”
- This talk will look at various failures across fields to encourage you to learn from them and have less of a need to create new ones :)
- Hopefully you can grow a spidey sense that will help you in the future.

If doesn’t feel quite right, it probably isn’t

- How often have you dealt with people who said “ok, the problem went away, so it’s solved, I can go back to whatever else I was doing”

NOOOOO!!!!
ICANHASCHEEZBURGER.COM

Is it fixed?

- I got home after a trip, my wireless was down. I power cycled it and it still didn’t recover.
- I then connected to it via ethernet and toggled a wifi configuration setting and it started working again.
- Did I fix it? Can I go back to other things?

NOOO!!!!!!
ICANHASCHEEZBURGER.COM

If it’s not root caused, it’s not fixed

- Yes, you have other things that need you, but maybe they can wait..
- … until you fully understand what went wrong and why
- If you can’t fix it right now, have a bug for you or someone else to fix soon
- Fixing symptoms is never an appropriate fix unless you understand the underlying problem and it can’t be fixed right away, or you’re ok with the problem happening again later
- “It went away, so we’re all good” is never a proper answer
- I deal with this all the time with support people. Please set a higher bar for yourself and others by example
- Don’t reward or measure people by amounts of cases closed or bugs fixed.

There are only 2 kinds of mistakes you can make

1) Honest ones that would have been hard to plan for

2) You being too proud / impatient / ”I know what I’m doing” / “This should (probably) work fine”

3) All the other kinds :)

#1 Is non trivial to fix but #2 can really be fixed with a single slide

The one slide that will prevent many failures

- Have you ever watched the old US TV show ER?
- When a doctor or intern lost a patient they were “invited” to a post mortem (a real one in this case)
- They were grilled on what they had done and hadn’t done and why
- Each time you are making a programming or sysadmin decision that you question just a little bit, ask yourself:

  “if this is all blows up, will my rationale for this decision make sense? Will people agree I was careful enough and didn’t miss an obvious better option?
- Will they (or I) not judge me for having acted impulsively?

When does this apply?

- Every time you question whether it should :)
- Each time you are taking a shortcut
- Doesn’t mean you should never/can’t
- Just that you weighed the pros and cons carefully
- So, if it all blows up, at least it was a careful decision.
- Companies are supposed to have processes to keep you honest, but you are the best person positioned to do it.

Earlier in my career

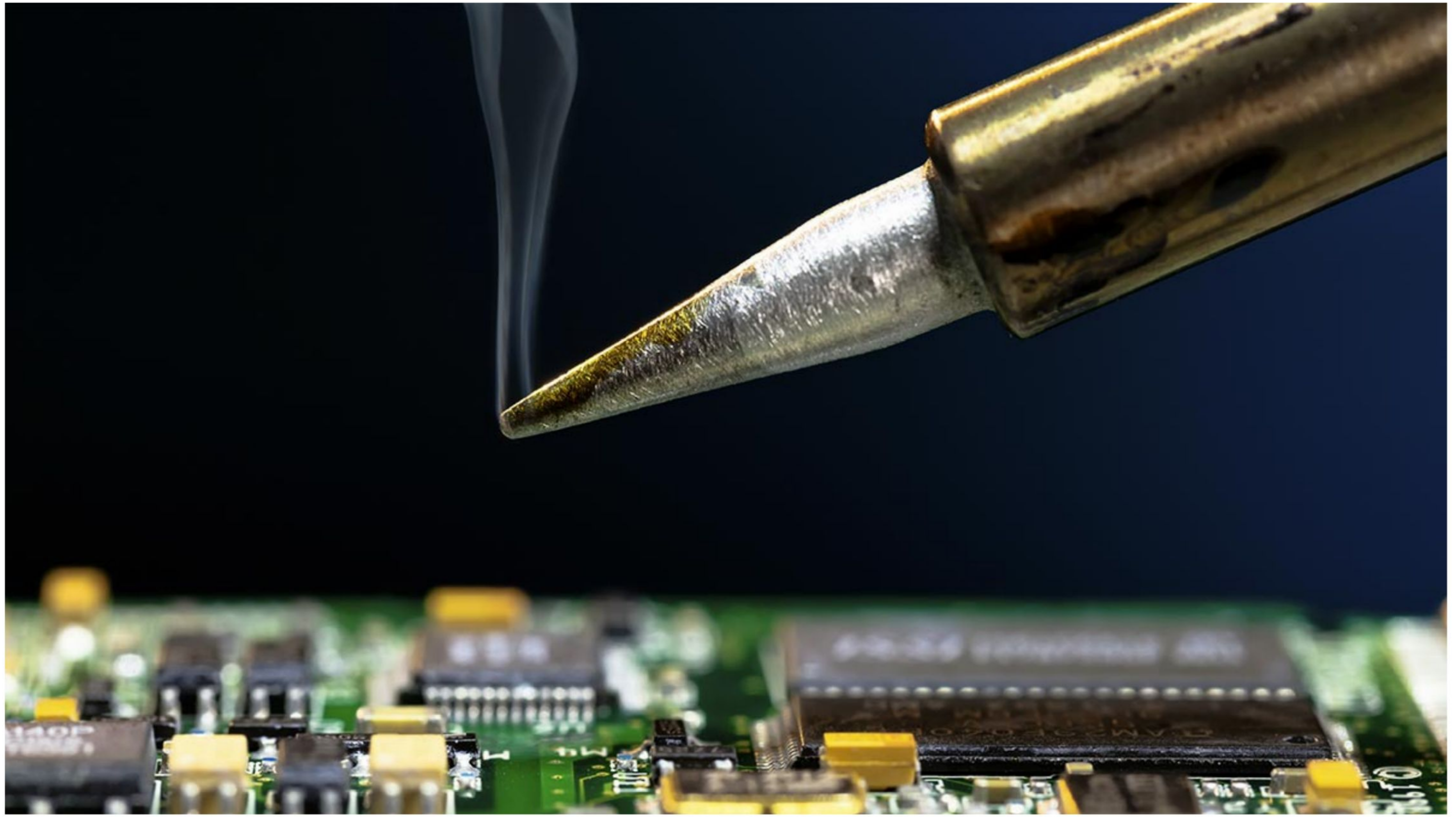
- I was young and conceit, and also good most of the time
- More than once, I did lots of changes at night with no easy revert and no warnings to peers. I even did them hours before going home to pack all night for an early flight to France/Xmas when no one was there to fix anything if it went wrong
- The only things I did right were:
    - I announced the changes I made.
    - I made very few mistakes.
    - I tested my changes to some extent before leaving.
    - I watched afterwards to some extent and was usually fixing before most people noticed anything was broken
- Still, this was stupid, and no big company should allow this.
- Don’t rely on anyone’s talent and continued heroics.

You’re done, thanks for coming

Ok, there is a little more

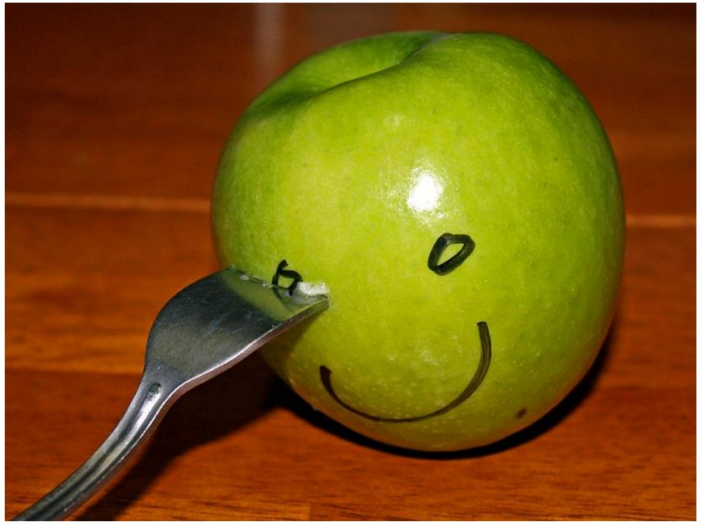
- But seriously, if you only remember the last 2 slides, you’ll be so far ahead of most people.
- The rest of the talk will look at lessons in learned in multiple areas, so that you can save the trouble of learning them from scratch

Hardware



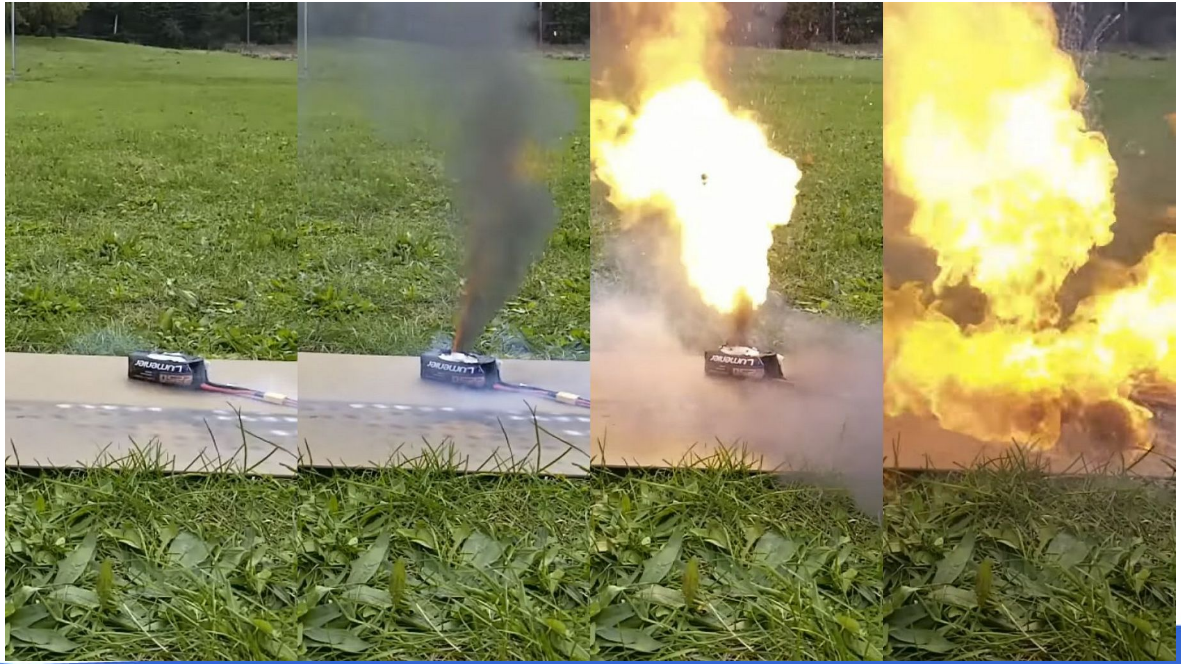
Basic Open Hardware Tip 1: Safety Goggles

Fingers are somewhat expandable (well the first one or 2), but you only have two eyes. You can treat the second one as a backup or a necessary one for stereo vision :)



Basic Open Hardware Tips: Lipos are out there to kill you

- Batteries are mostly a slower burning fire normally (but not always) redirected through wires, especially lithium ones
- Pro-tip: do not pour water on a lipo fire

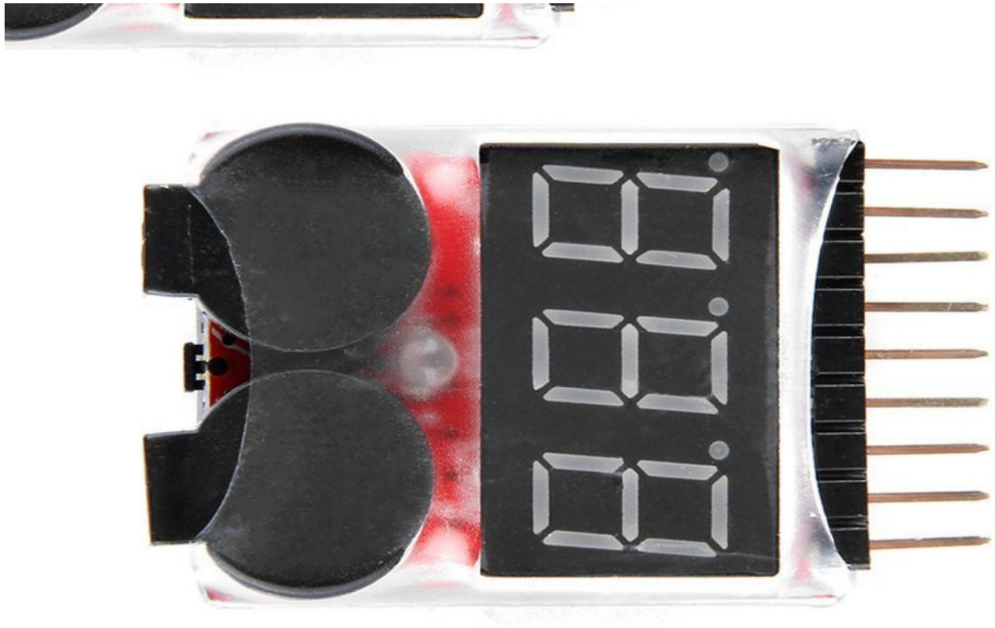


Basic Open Hardware Tips: Better Lipos are Safer

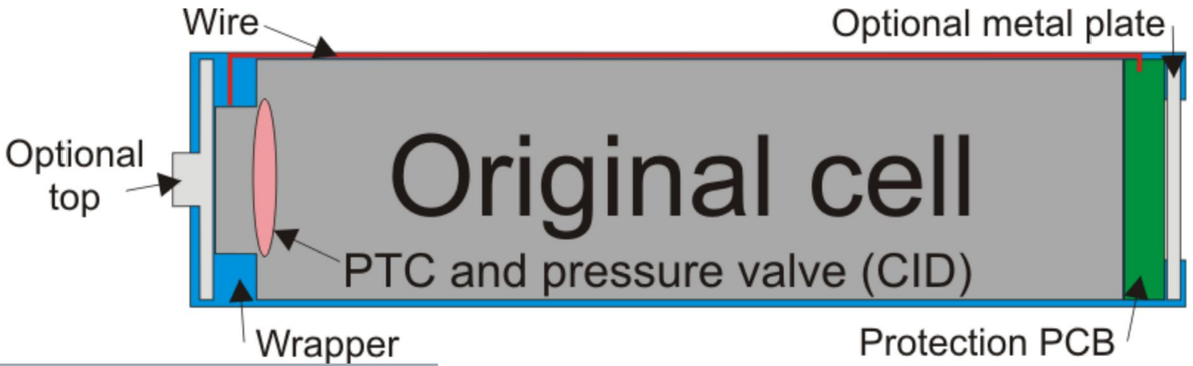
- The cell envelope is critical to preventing fires

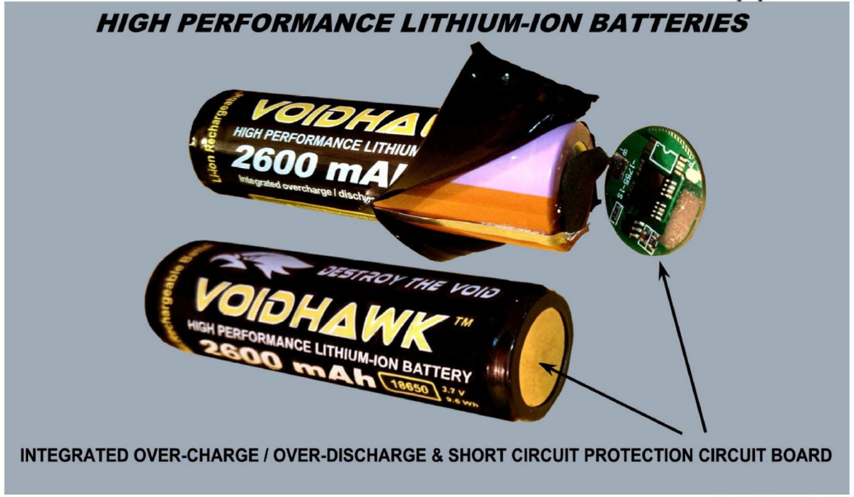
Basic Open Hardware Tips: Lipo Voltage Protection

- Make sure you carefully monitor all cells of any unprotected lipo



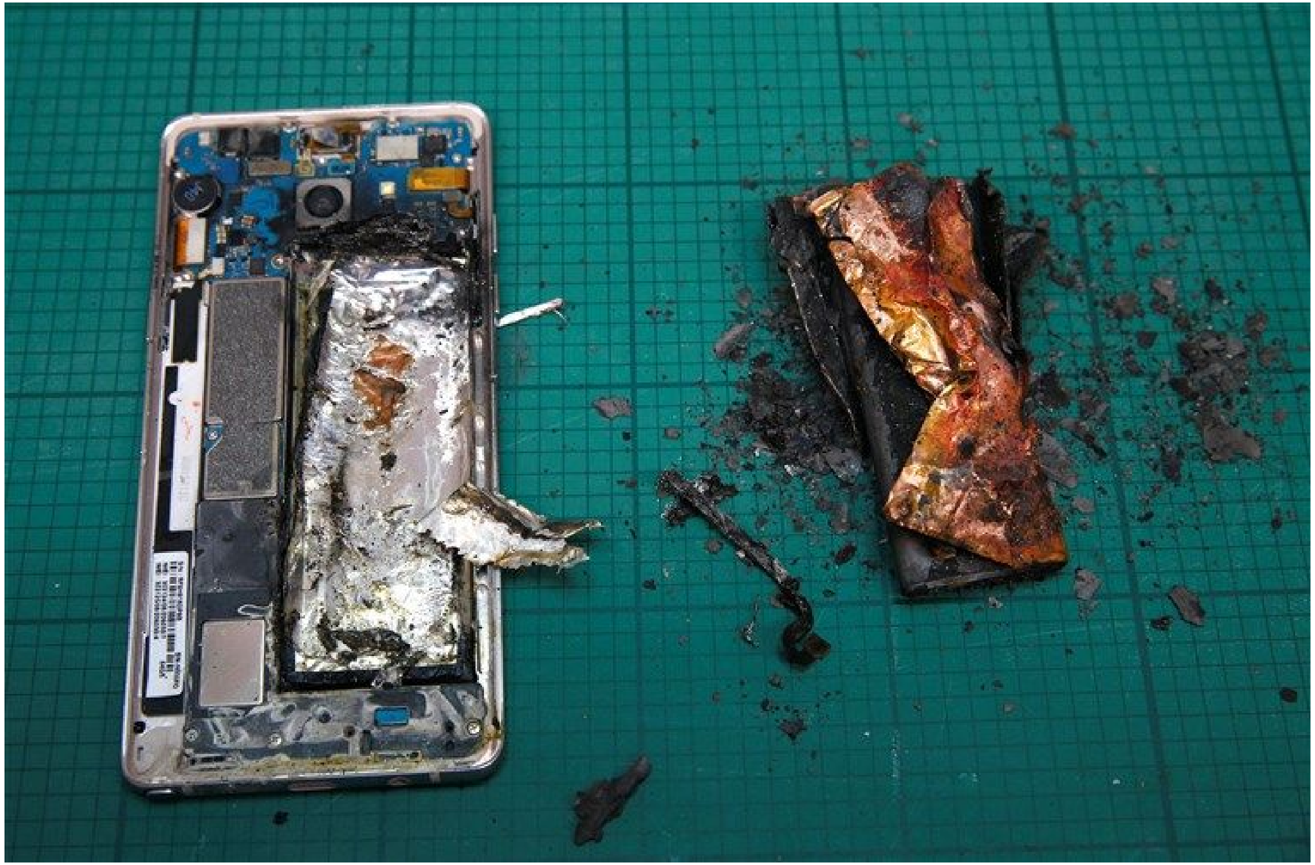
Basic Open Hardware Tips: Protected Cells are Best




HIGH PERFORMANCE LITHIUM-ION BATTERIES
VOIDHAWK
HIGH PERFORMANCE LITHIUM-ION BATTERY
2600 mAh
18650
DESTROY THE VOID
INTEGRATED OVER-CHARGE / OVER-DISCHARGE & SHORT CIRCUIT PROTECTION CIRCUIT BOARD

Remember those Samsung Note 7s that self ignited?

- The battery was upgraded last minute before shipping
- It mostly fit, but there was no room for expansion
- Manufacturing used 2 different battery manufacturers, one had batteries that were ever so slightly more thick
- Some batteries expanded into a screw or other scrap metal left by other shoddy manufacturing
- which breached the protective envelope
- which in turn allowed the lithium to react with the air
- and happily burn
- NOT PROFIT!



Morals of the Samsung Note 7s Story

- Let’s be honest, this is a complex multi level failure
- Each component passed its own tests
- They likely even passed integration tests

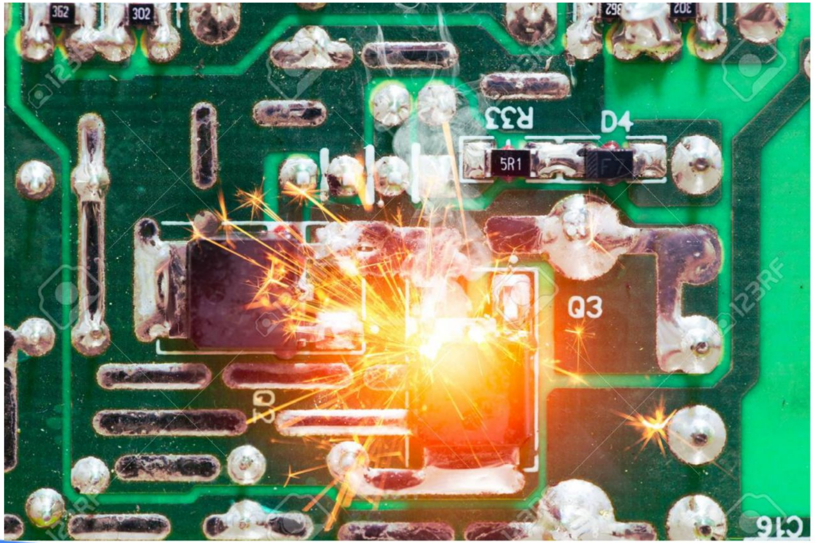
Lessons learned:

- Design redundant/multi level safeties
- Like a fuse in the battery and a separate fuse in the system
- If time/budget allows, don’t rely on end user testing only, but do include it as the last level
- Whenever possible, have flexible ship dates. Don’t force ship something everyone knows isn’t sufficiently tested/ready/finished, unless you really must (Xmas)

Basic Open Hardware Tip 3: Fuses are your Friends

Every circuit has a fuse

- either you chose where it is
- or the circuit will choose for you.



Basic Open Hardware Tip 4: Have Spares

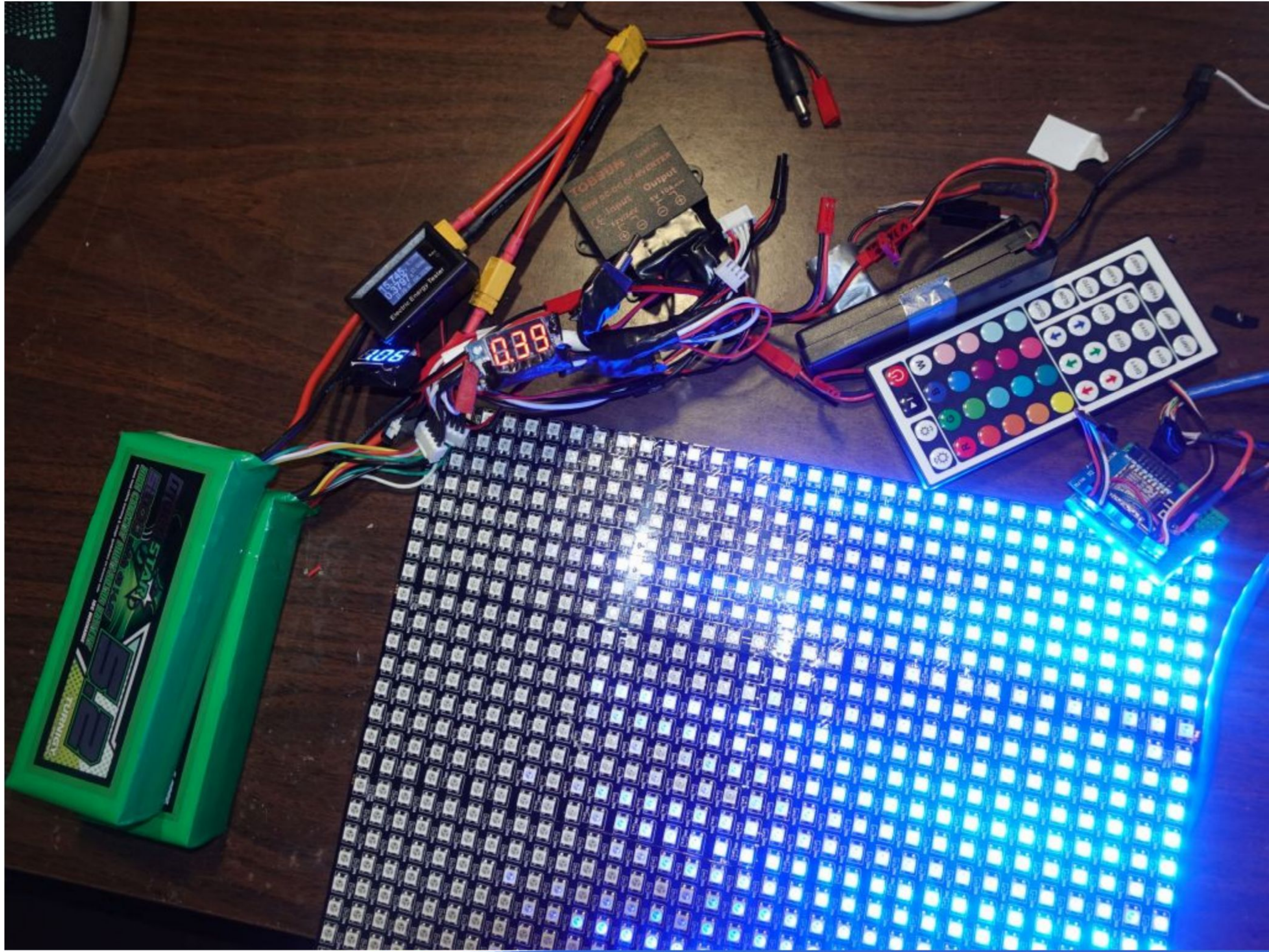
- You will make mistakes
- Software is easy to reset, hardware not always as much
- Sucks to wait days or weeks for new hardware after burning a single piece you needed
- If you can afford it and your time is precious, order spares, at least 2 of everything

The human factor

- Your baby is never ugly, at least not to you.
- My LED outfit uses some amount of hand pieced electronics
- It’s modular so that I can bypass non essential components that fail
- Or replace essential ones
- But it looks like a mess, with wires (cut the red one or the green one?)
- And red LED numbers that don’t count down but look like they might
- I had cops detain me until they were convinced I wasn’t a threat. Got to meet bomb specialists at SFO airport and was denied entry at two EDM festivals
- In the software world, it translates as “UI and user appearence matter”

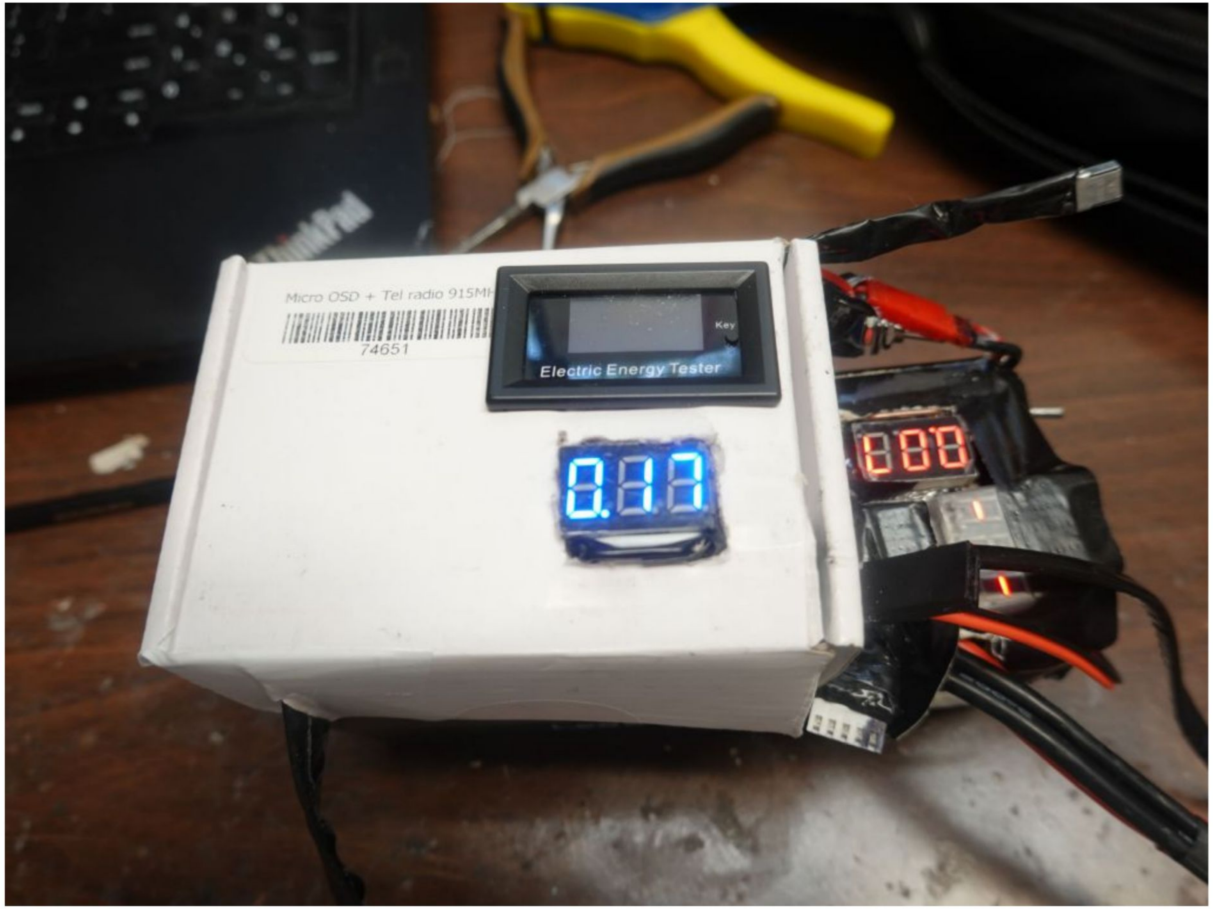
Before

- Mess of wires
- Risk of short
- Scary looking counters



Google

After


Micro OSD + Tel radio 915MH
74651
Key
Electric Energy Tester
0.17

Profit!

Software

Google Culture

- Revert first, discuss/point fingers later
- Code reviews or Change requests for ops
- Unittests
- Blameless post mortems
- Practice emergency/recovery

Code Reviews

- No matter how good we are, we all make mistakes eventually
- Just sending a code review, forces you to re-read your changes and think about a justification for all of them
- The amount of bugs found from sending code reviews, is staggering
- If you work solo, do a self code review when writing git checkin message
- Big amount of problems I fixed on my own while writing a detailed Email on what was wrong and what I had already tried
- The first reviewer of your code sent for review should always be you!
- Pair programming can used in lieu of TBR

Change Requests

- For (Sys)Ops, sending a plan of what you're going to change and why works similarly.
- Including why it's best to do it in the middle of the night/weekend (downtime for users)
- Weighed against why it's not (staff tired and motivated to go back to bed or weekend ASAP, not around to deal with delayed problems)
- Impact on other teams that may not be around/awake to notice breakage if not done during work hours (with suitable warning of course)

TBR: To Be Reviewed

- When time matters and “We really need to submit this, and I’m putting my job on the line for it”, we have TBRs.
- Your code still gets reviewed, but you can submit it before it is
- Obviously you should be ready to justify why you circumvented the normal process
- If you do this too often, your manager will likely have a chat with you, or even if you used it a single time but at a very wrong time.
- With great power comes great responsibility.
- This works surprisingly well. Checks and Balances…

Unittests

- By now, most serious companies and projects use unittests.
- They are necessary but don’t treat them as sufficient
- Sometimes unittests are nothing more than compile tests, try to strive for more
- Aim for end to end functional tests, using an emulated environment that mirrors the real one as much as possible (I sent ‘2’ in the function, and it sent back ‘3’ is a very low bar).
- Try to use a real system instead of mocking it away, if possible
- They are great for testing on architectures the developer doesn’t own

CQ/CI (Commit Queue, Continuous Integration)

- Catch regressions early to make them easier to find. I’ve seen too many problems that took so long to find after they were allowed in the tree due to lack of tests. Have tests and CI/CQ as soon as you can.
- Projects like Chrome/ChromeOS and Fuchsia use both, based on luci https://luci-scheduler.appspot.com/jobs/fuchsia/ci-fuchsia-x64-release
- Developers are supposed to do their own testing first and before they can send a change for review, it is run by CQ, for which fuchsia runs lots of tests in emulated environments (qemu in google cloud), and actual hardware of different types (CPU and peripherals)

CQ/CI costs and issues

- Hardware tests are often more expensive to scale, so tests are targeted accordingly
- Some tests are too expensive/too long to run as presubmit/CQ, so they are run in CI
- If a CI run fails, it notifies the list of CIs that were included.
  Running CI more often means fewer people get bundled in that warning Email, and probably confused.
- Running tests on real hardware is essential for projects like ChromeOS or Fuchsia, and useful for some projects like Chrome that use hardware to find some bugs, timing or power use regressions, but requires a dedicated team to roll out and maintain the hardware
- You now also need to deal with both software and hardware flakes. Retrying tests once helps, after that you need people to look into why they flake or failed.
- Aggressively go after flaky software tests and get them fixed or removed. A flaky test is worse than no test.

Percent rollouts / Safe file updates

- Anything that rolls out a text file (like /etc/aliases), should refuse to proceed if more than x% is being changed (catches accidental partial deletes)
- Just like unittests, make sure new config files have proper syntax and the intended behaviour before they get rolled out
- Back in the day, I wrote a DNS/bind config edit system that spun a test bind with the new config files, scanned logs for warnings and errors, and did test queries against it before the new config file was allowed to be checked in and rolled out (functional unittest with real parser instead of mocked one)

Blameless Post Mortems

- https://landing.google.com/sre/sre-book/chapters/postmortem-culture/
- When things go wrong anyway (not if, but when), revert first, argue later
- Anyone can request a post mortem
- Either because there is truly something to learn
- Sometimes also because the team(s) that made the mistake could use the experience and the time to reflect :)
- It states facts and timelines, not fault or blame.
- It’s often a bit humbling to have to write a postmortem, so, it’s also an incentive to try harder to avoid a future one :)
- Failure should be treated as a lesson to all learn from

Practice Emergencies when everyone is awake and ready

- We have a week per year when internal systems get to practice live failure and recovery scenarios (within reason)
- This includes having the main office swallowed by a major earthquake, and recovery having to be done by remote offices only
- These exercises often find documentation shortfalls and dependencies/compound failure scenarios that we didn’t know about.
- They need to be done carefully though, you don’t want to cause more damage than the potential for learning. This includes an abort/revert plan.
- That’s an occasion to define and practice your revert scenario (any rollout needs to have a revert plan. Run forward only should avoided).

Chmod-Gate

- mkdir -p -m 755 /usr/local/foo is safe, is it not?
- It started taking down part of google production before someone pushed the emergency switch and stopped the rollout
- TL;DR: /usr/local/foo is created with 755 but then the user umask is applied (often 0027 or 0077 for safety).
- During testing, the tester had permissions to traverse /usr/local
- At install time, /usr/local/ got created with permissions 700 and broke everything as non root processes got access denied.
- Ultimately it’s all about relative speed of rollout, watching and being able to stop and revert quickly.

Overachieving engineers and computers

https://landing.google.com/sre/sre-book/chapters/automation-at-google/#xref_automation_diskerase-sidebar

- We have a few servers in some colos we don’t own for various reasons
- To protect user and our data, we have a diskerase process.
- After trying to re-erase a rack that didn’t seem fully erased, automation determined the new set of machines to erase was empty
- Due to a bug, that empty set was treated as “everything”
- The nice automation complied and did so quicker than humans were able to stop it.
- Moral of the story: double check any destructive code and make sure it gets throttled.

Percent rollouts are not the ultimate safety

- Google uses percent rollouts within datacenters and within the global network, they’ve always worked fine
- Until…
- One system rolled out a new config file that specified a rollout
- But it ended up having a problem that crashed every rollout server that received the file
- The percent rollout didn’t save us and our efficiency in syncing within our network, cost us dearly
- Fix is to also gate rollout of config file changes

Learning From Post Mortems

- If you have to write it, makes it harder to wave off a mistake and do it again later
- If you get to read it, you will have those “I’m so glad I wasn’t the one who made that mistake” moments
- And there are those where a string of reasonable looking actions cause an unexpected failure. Those are the best ones to learn from and shape a more critical mind and your spidey sense
- “oh no, let’s not do this, team X did last year and turned out it was unsafe”
- https://landing.google.com/sre/sre-book/chapters/postmortem-culture/
- https://landing.google.com/sre/sre-book/chapters/reliable-product-launches/

My Personal Lessons

- From aviation and diving: have a plan before you need it, because once you’re in the middle of problems, the bigger the problem, the more IQ you lose on average
- For flying/diving: learn and practice emergencies before they happen
- Be super careful about “temporary fixes” or “live fixes”. There are times they are necessary, but no one involved should be allowed to go home before the fixes are submitted properly, or high priority bugs opened to fix it right. Also, oncall people must be notified of the temporary situation.

Summary

- Generally, a given person/company should make the same mistake twice, or you’re really doing it wrong
- For extra points, you’ll then also avoid doing similar mistakes you didn’t have to make yourself but thought about in the post mortem, and/or read about
- For more extra points, you’ll simply have read about that problem and avoided it without ever having to experience it yourself.
- For Ops/SRE, have a look at https://landing.google.com/sre/sre-book/ and other publications from others
- This is what aviation does a lot because relearning failures often leads to death
- Aviation says: “Experience is a cruel teacher, she gives the test first, and if you survive, then you get the lesson”.
- We can learn from aviation!

A...

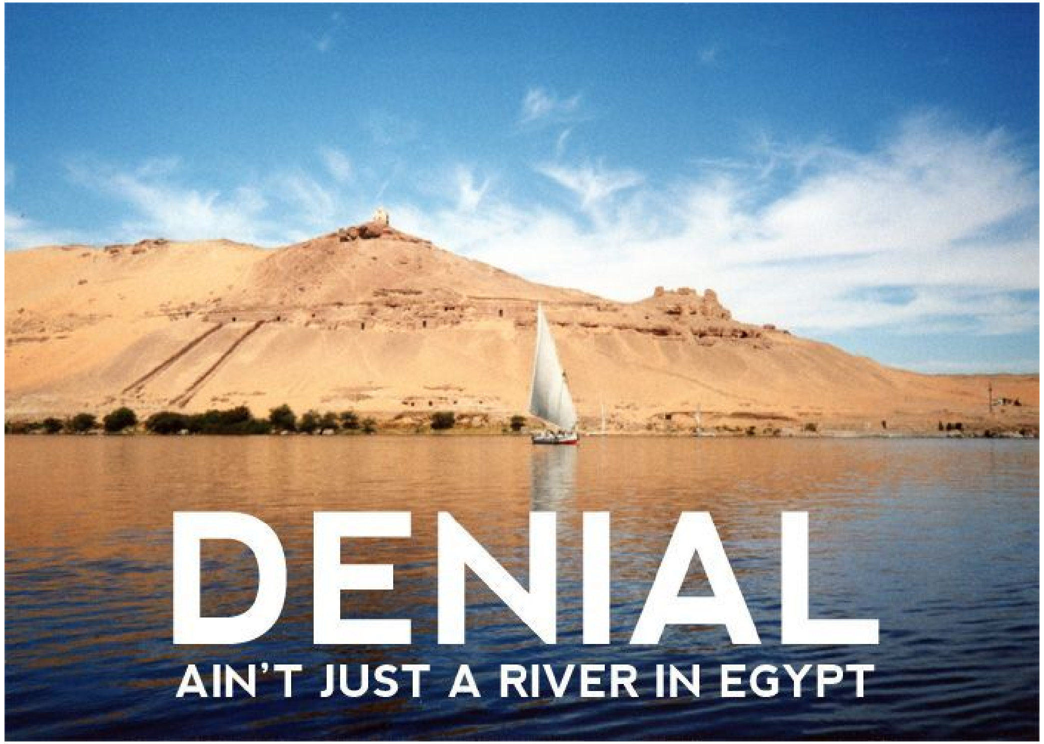
...via...

...tion

There is so much to learn from aviation

- I’m not a great pilot, I’m just an average one
- One thing I did learn though, was honest risk management
- Because it’s not about losing a bonus or getting a bad perf review
- It’s about not dying and not killing others either
- We are so good at rationalizing bad decisions while we are making them. It takes true willpower to break that
- Yet, our brains are really good at self denial

Remember, about denial...



There is so much to learn from aviation (2)

- If the fear of a bad review or writing a postmortem doesn’t do it
- The fear of dying hopefully will
- And if not, then you remove yourself from the gene pool
- So it all works out :)

FIREFIGHTER: WHAT HAPPENED?
N36638
PILOT: I DON'T KNOW, JUST GOT HERE MYSELF.
imgflip.com

JOKES ABOUT CRASHES
ARE JUST PLANE WRONG
quickmeme.com

The Automation Debate

- Tesla vs Waymo
- Boeing vs Airbus
- https://www.computer.org/csdl/magazine/sp/2015/05/msp2015050104/13rRUxASutL William Langewiesche's article analyzing the June 2009 crash of Air France flight 447 comes to this conclusion: “We are locked into a spiral in which poor human performance begets automation, which worsens human performance, which begets increasing automation” (www.vanityfair.com/news/business/2014/10/air-france-flight-447-crash).
- Is this relevant to computers? Hell yes! Automation is required, but you must know how to disable it and how to take over

The Automation Debate: Tesla vs Waymo

- I’m biased both ways. I work at Google, but I own a Tesla with AP3 since I can’t buy a Waymo car
- Tesla has a bold plan to rely on computer vision only because it’s cheaper, but they will be the first to make it work if they do. Waymo uses many more sensors, including Lidar.
- Waymo decided that drivers can’t be expected to take over when automation fails, so it must handle all cases, including main computer failure. Downside is that it’s a huge job and is being rolled out slowly.
- Tesla has “get it today” technology, but it’s very incomplete and works until it doesn’t. This includes the 2 tesla cars that got decapitated by a semi truck perpendicular to 2 lanes of traffic. They were invisible to radar and computer vision wasn’t ready for them, yet.
- This dichotomy is valid for computer automation too. Your engineers need to understand the automation if it goes wrong or it needs to be 100% fault proof

Airbus Automation and waning pilot skill: AF447

AF447, what went wrong

- 3 pilots, only one is seasoned, the other two are pretty junior
- Senior pilot goes for required sleep half way through the flight
- Bad thunderstorm over the equator, 3 pitot tubes iced up
- Plane automation turned off and reverted to alternate law
- Junior pilots poorly trained to actually fly a plane in alternate law and with no outside reference (no horizon)
- They can’t figure out how much to pull the nose up or down despite a standby instrument which admittedly is not easy to find in an emergency
- They panic and make many other mistakes including overriding one another => having someone in charge to synchronize efforts, matters

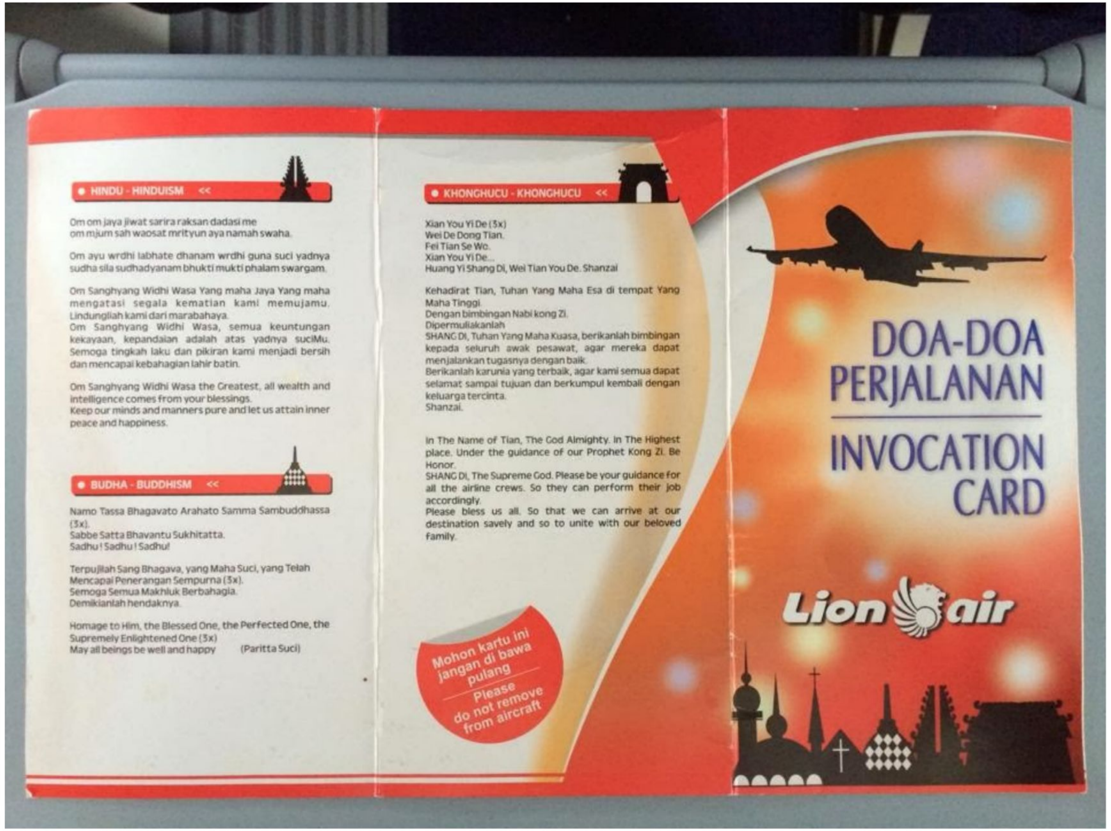
AF447, what went wrong (2)

- Pilot pulls nose up so much that the plane enters a deep stall
- It gets so slow that the **stall warning turns off while in a very deep stall** because programmer thought a plane couldn’t actually fly that slowly
- An airbus would never allow a pilot to do this, but in alternate law, it has to.
- Plane falls out of the sky, the least experienced pilot pulls the stick to continue the stall (rookie mistake) while the somewhat better one pushes it down
- Airbus averages both inputs without pilots really knowing.
- Plane stays in a deep stall and crashes in the ocean at 3000 meters/mn falling rate. Everyone dies, it takes over a year to find the black boxes.
- Airbus programmers said it was the pilots’ fault for both trying to fly at the same time against procedures. Airbus planes still average input from 2 pilots without force feedback to either pilot.

QF 32, Airbus A380 with terrible Rolls Royce Engines

- One of too many Roll Royce engine failures for that plane due to shoddy manufacturing.
- Uncontained engine failure that severs part of the wing, including some hydraulics and control to one engine
- Throws out around 100 errors on the plane computer.
- 4 pilots were kept busy acknowledging all those errors sent in random order and deciding which ones were important
- It took superior airmanship and 4 well trained pilots working together to barely land the plane. Everyone survived
- It’s unclear if airbus airplanes now prioritize errors in order or urgency

The case for Airbus automation: countries like Indonesia



● HINDU - HINDUISM <<

Om om jaya jiwat sarira raksan dadasi me
om mjum sah waosat mrityun aya namah swaha.

Om ayu wrdhi labhate dhanam wrdhi guna suci yadnya sudha sila sudhadyanam bhukti mukti phalam swargam.

Om Sanghyang Widhi Wasa Yang maha Jaya Yang maha mengatasi segala kematian kami memujamu. Lindungliah kami dari marabahaya.
Om Sanghyang Widhi Wasa, semua keuntungan kekayaan, kepandaian adalah atas yadnya suciMu. Semoga tingkah laku dan pikiran kami menjadi bersih dan mencapai kebahagian lahir batin.

Om Sanghyang Widhi Wasa the Greatest, all wealth and intelligence comes from your blessings.
Keep our minds and manners pure and let us attain inner peace and happiness.

● BUDHA - BUDDHISM <<

Namo Tassa Bhagavato Arahato Samma Sambuddhassa (3x).
Sabbe Satta Bhavantu Sukhitatta.
Sadhu ! Sadhu ! Sadhu!

Terpujilah Sang Bhagava, yang Maha Suci, yang Telah Mencapai Penerangan Sempurna (3x).
Semoga Semua Makhluk Berbahagia.
Demikianlah hendaknya.

Homage to Him, the Blessed One, the Perfected One, the Supremely Enlightened One (3x)
May all beings be well and happy (Paritta Suci)

● KHONGHUCU - KHONGHUCU <<

Xian You Yi De (3x)
Wei De Dong Tian.
Fei Tian Se Wo.
Xian You Yi De...
Huang Yi Shang Di, Wei Tian You De. Shanzai

Kehadirat Tian, Tuhan Yang Maha Esa di tempat Yang Maha Tinggi.
Dengan bimbingan Nabi kong Zi.
Dipermuliakanlah
SHANG DI, Tuhan Yang Maha Kuasa, berikanlah bimbingan kepada seluruh awak pesawat, agar mereka dapat menjalankan tugasnya dengan baik.
Berikanlah karunia yang terbaik, agar kami semua dapat selamat sampai tujuan dan berkumpul kembali dengan keluarga tercinta.
Shanzai.

In The Name of Tian, The God Almighty. In The Highest place. Under the guidance of our Prophet Kong Zi. Be Honor.
SHANG DI, The Supreme God. Please be your guidance for all the airline crews. So they can perform their job accordingly.
Please bless us all. So that we can arrive at our destination savely and so to unite with our beloved family.

Mohon kartu ini jangan di bawa pulang
Please do not remove from aircraft

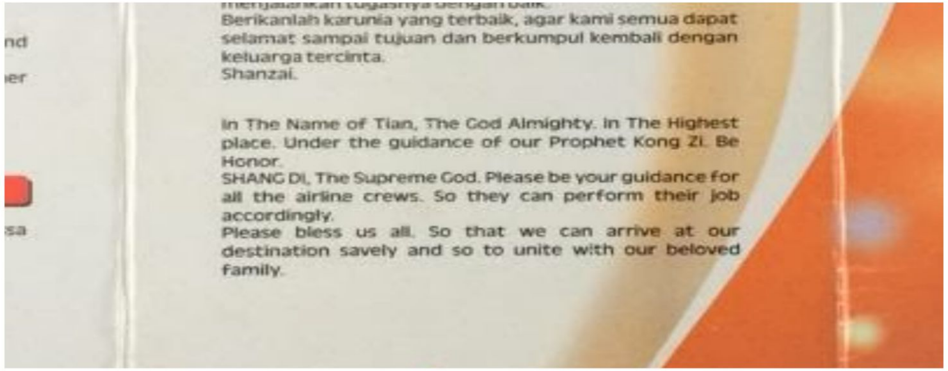
DOA-DOA PERJALANAN

INVOCATION CARD

Lion air

Google

Indonesia (and some other countries)



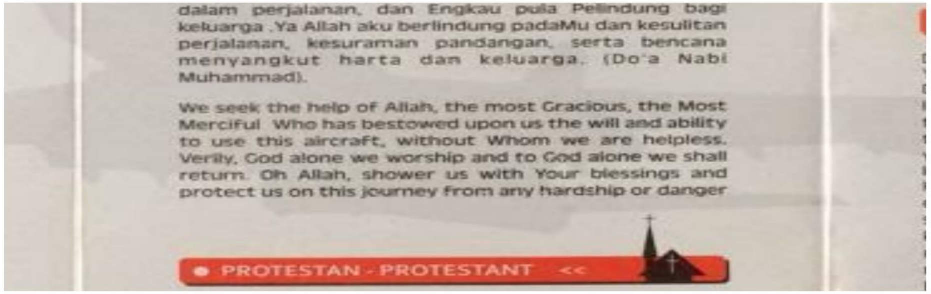
nd

ier

sa

menjalankan tugasnya dengan baik.
Berikanlah karunia yang terbaik, agar kami semua dapat selamat sampai tujuan dan berkumpul kembali dengan keluarga tercinta.
Shanzai.

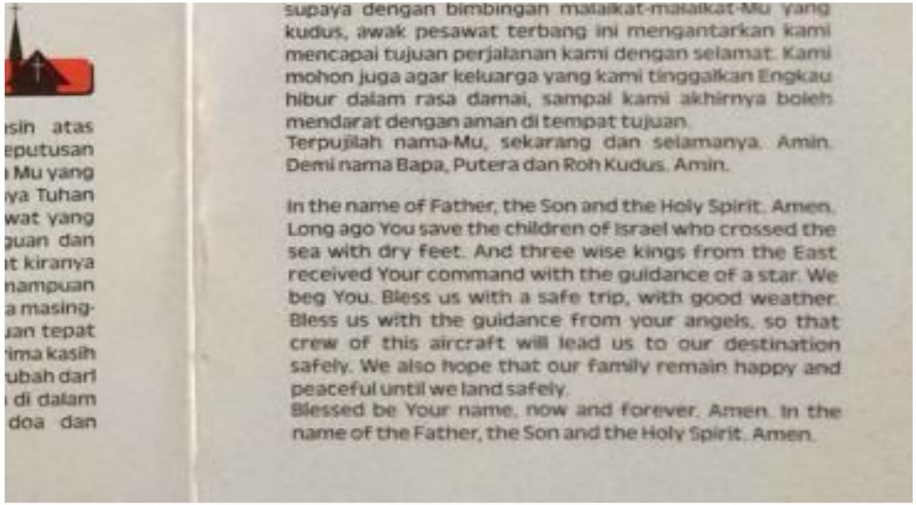
In The Name of Tian, The God Almighty. In The Highest place. Under the guidance of our Prophet Kong Zi. Be Honor.
SHANG Di, The Supreme God. Please be your guidance for all the airline crews. So they can perform their job accordingly.
Please bless us all. So that we can arrive at our destination savely and so to unite with our beloved family.



dalam perjalanan, dan Engkau pula Pelindung bagi keluarga .Ya Allah aku berlindung padaMu dan kesulitan perjalanan, kesuraman pandangan, serta bencana menyangkut harta dan keluarga. (Do'a Nabi Muhammad).

We seek the help of Allah, the most Gracious, the Most Merciful Who has bestowed upon us the will and ability to use this aircraft, without Whom we are helpless. Verily, God alone we worship and to God alone we shall return. Oh Allah, shower us with Your blessings and protect us on this journey from any hardship or danger

● PROTESTAN - PROTESTANT <<



sin atas
eputusan
Mu yang
ya Tuhan
wat yang
guan dan
t kiranya
nampuan
a masing-
uan tepat
rima kasih
ubah dari
di dalam
doa dan

supaya dengan bimbingan malaikat-malaikat-Mu yang kudus, awak pesawat terbang ini mengantarkan kami mencapai tujuan perjalanan kami dengan selamat. Kami mohon juga agar keluarga yang kami tinggalkan Engkau hibur dalam rasa damai, sampai kami akhirnya boleh mendarat dengan aman di tempat tujuan.
Terpujilah nama-Mu, sekarang dan selamanya. Amin.
Demi nama Bapa, Putera dan Roh Kudus. Amin.

In the name of Father, the Son and the Holy Spirit. Amen.
Long ago You save the children of Israel who crossed the sea with dry feet. And three wise kings from the East received Your command with the guidance of a star. We beg You. Bless us with a safe trip, with good weather. Bless us with the guidance from your angels, so that crew of this aircraft will lead us to our destination safely. We also hope that our family remain happy and peaceful until we land safely.
Blessed be Your name, now and forever. Amen. In the name of the Father, the Son and the Holy Spirit. Amen.

Check for bit flips, secret military bases, and solar flares

- Mayday Air Crash Investigation is an addictive series https://en.wikipedia.org/wiki/List_of_Mayday_episodes
- 2 planes had an unexplained uncontrolled pitch down towards the ocean in the same general area.
- Detailed and thorough investigation showed the angle of attack and speed data streams got crossed for a few samples and then crossed again.
- Computer received the wrong data samples and interpreted as stall imminent and did an emergency pitch down to recover.
- Planes self recovered because as the data streams recovered. Best guess was that a nearby base sent radio signals that created enough EMI to jam the planes' internal data buses.
- Airbus programmers relied on their data bus being perfect and didn't plan for data corruption on the bus.
- It's easy to notice one data sample is out of wack and ignore it. Program defensively. Ardupilot does a better job with bad data input.

So, how about boeing?

- Older boeings didn’t limit the pilot in any way
- You could actually roll a plane upside down, and lesser pilots could also crash them more easily, which did happen.
- Newer ones have more automation that helps/gets in the way
- I’ll skip the windows that don’t close, don’t dim enough to properly block the sun in the dreamliner/787.
- Which brings us to the ill fated 737 Max
- The 737 Max wanted to use more efficient/bigger engines that didn’t clear the ground to compete with the newer Airbus NEO

737 Max

- Engines had to be mounted higher up on the wing so as not to scrape the ground. Note that said engines were not designed to fit the 737
- It changed the plane stability in a way that it could pitch up in an irrecoverable way (plane stalls, nose can’t be pitched down, more power makes things worse)
- Instead of fixing the problem for real or re-designing the plane, boeing added a software patch: MCAS
- It’s supposed to pitch the plane nose down if the angle of attack is detected to be too high.
- Questionable fix, but it could have worked, except...

737 Max, how many things went wrong?

- To save on money, MCAS was rushed, deadlines “had to be met”
- The prime directive was to pretend the plane was unchanged. MCAS was not to be mentioned in the pilot manual, robbing pilots of any chance of knowing what could go wrong with it and how to recover
- MCAS relied on an angle of attack indicator. The plane only shipped with one, a 2nd one was optional equipment for an extra price. It used to be standard, but bean counters got in the way.
- Even if you got 2, MCAS was only connected to the main one.
- If the single AOA failed, it could force the plane to pitch down towards a crash.

737 Max, how many things went wrong?

- During certification, MCAS was eventually given so much control authority that pilots could not overpower it (trim controls the full horizontal tail, and elevator only a tab on it. At higher speeds, elevator cannot over power the tail)
- A single AOA failure would cause the plane to pitch down to a crash
- The only way to recover was to turn on the autopilot, something no pilot should never do in an emergency, or pull the correct fuse quickly enough
- But pilots were not told about MCAS, how it worked, how it could fail, or how to disable it
- End results: 2 planes crashed with everyone dead before all planes were grounded.

737 Max, how much money they try to save?

- Decided to fit an engine that didn’t really fit on an airplane for profits
- Refused to re-categorize the aircraft
- Patched hardware with software and refused to document it
- Used an unreliable AOA vane and didn’t bother improving/fixing it
- Made all these savings by relying on a new system, MCAS
- So, surely the essential MCAS system was at least assigned to be designed and written by top senior engineers, right?

737 Max, MCAS Engineers

- How much would you pay those crucial engineers writing that crucial piece of software essential to the lives of everyone onboard?
- $50/h?
- $100/h?
- $150/h?
- $200/h?
- More?

737 Max, MCAS Engineers...

- Or you could just outsource this to India...
- ... who then used recent grads, paid **as little as $9/h**
- And then they did not even properly test or integrate the software they send back to Boeing.
- This will help cut costs, right?
- Except for the 300+ people who died as a result
- And the 60+ billions Boeing is now paying airlines to avoid lawsuits for the money they are all losing with their grounded planes, canceled flights

737 Max, Regulation didn’t save us

"Let's quit messing around about the chances of this happening being rare," Mr. Reed remembered saying. “The company outlined how a single sensor that measured the angle of the plane's nose would trigger MCAS, Mr. Reed recalled, but argued a failure wasn't likely and the system would kick in only in extreme conditions. "If it can happen, it's going to happen."

The Boeing spokesman said a single sensor "satisfied all certification and safety requirements," and potential additional training wasn't considered when assessing MCAS hazards.

From the start, safety-assessment documents Boeing provided to the FAA assumed pilots would be able to handle misfires. Regulators endorsed that determination, along with the single sensor. The FAA certification rules under which the MAX was allowed to fly assume pilots react correctly to certain emergencies 100% of the time.”

What Boeing Top Management pushed for:

- Pushed to fit an engine that wasn’t meant to fit the plane
- Relied on a questionable patch to keep the plane from stalling
- Cared so little about it that they outsourced it to India and to lowly paid junior engineers
- They didn’t verify it actually worked safely or was integrated properly
- During certification, MCAS was eventually given so much control authority that pilots could not overpower it (trim controls the full horizontal tail, and elevator only a tab on it. At higher speeds, elevator canont overrule the tail)
- A single AOA failure would cause the plane to pitch down to a crash. Using both AOA sensors available in planes, would have been more work and delayed shipment.
- Are 2 sensors even enough? AF447 had 3 pitot tubes so that you can rule out a bad reading with the other 2.

737 Max, how many things went wrong?

- They forced everyone to pretend the plane was unchanged
- They didn’t tell pilots about this critical system
- But yet, they expected pilots to diagnose the failure of that unknown system and know how to disable it in 4 seconds by pulling the correct fuse, or everyone would die
- 4 seconds is barely enough to diagnose and fix a rehearsed failure.
- The failure happened to untrained crews
- They didn’t know what to do
- Everyone did die. Twice!
- All planes were finally grounded.

Boeing became unable to ship planes they were building

They had to halt production at 400 after running out of room

Fines, lawsuits, and grounded planes cost > 60 billion USD

The problem with certification

- Boeing washed their hands of potential TCAS problems with “the pilots can handle it” (and have to within seconds)
- But they also ensured TCAS wasn’t mentioned in the aircraft manual to better sell the plane as “doesn’t need new training”
- The FAA’s job was to catch this, but airplanes have become so complex that the FAA is not appropriately staffed to look into all this and failed miserably at their job “you had one job…”

FAA: You had one job... (ok, more than one, but still...)

FAA: you failed the first time, let's give you a 2nd chance?

Agencies that do certifications vs complex systems

- The FAA washed their hands of all this complex computer stuff and allowed Boeing to “self certify”. Doesn’t it sound like “mmmh, I can’t understand your code, how about you do your own code review?”
- Will that remain the case? Is the FAA appropriately staffed?
- Have you seen Karen Sandler’s talks on her implanted defibrillator, how insecure they are and how easy it would be to remotely trigger them and kill their users?
- The FDA is no better than the FAA, they don’t sufficiently understand computer systems, and even still allow insecure windows XP medical devices connected to the internet!

737 Max, The small print (you can read later)

https://www.wsj.com/articles/the-four-second-catastrophe-how-boeing-doomed-the-737-max-11565966629

To use less fuel, the design called for larger engines that would be moved forward and higher than in the previous model. The changes affected how the plane handled, though. Its nose pitched up in certain high-altitude conditions, risking a stall, the term for a sudden loss of force called lift that keeps planes aloft.

Boeing wrote: “same pilot type rating, same ground handling, same maintenance program, same flight simulators, same reliability.” Equally important was that it had to have the same flying characteristics. This was a regulatory necessity if the Max was to escape onerous reclassification as a new airplane. And there was a problem. Boeing test pilots discovered that the Max had unusual stall characteristics when the wing flaps were up and the engines were thrusting.

One of Boeing’s bewildering failures in the MCAS design is that despite the existence of two independent angle-of-attack sensors, the system did not require agreement between them to conclude that a stall had occurred.

Apparently the captain noted nothing about the failure of the newly installed angle-of-attack sensor, or the activation of the stick shaker, or the runaway trim, or the current position of the trim cutout switches. If true, it was hard to conclude anything other than that this was severe and grotesque negligence. Dave Carbaugh, the former Boeing test pilot, had the most charitable view of the matter. “I suspect that the pilot wrote what you see in the log, and he verbally told maintenance that, ‘Hey, the trim was running down, and we had to use the stabilizer cutout switches, and we flew the airplane back manually,’ ” he said. “And **maintenance took no action on that, because the airplane had made it back to Jakarta. They just checked the fault messages and cleared them and called it a day**. That’s my best guess. They were just hellbent to release that airplane.”

According to the written record, the mechanics responded to the captain’s reports by flushing out a couple of air-data units, cleaning an electrical plug and declaring success after running some ground tests.

=> does this sound like "unittests passed, let's ship it" ?

In the aftermath of the accident, it turned out that a warning light indicating disagreement between the two angle-of-attack sensors was absent from the airplane because it was being offered on the Max only as part of an optional angle-of-attack instrumentation package. On previous models, the light had come as standard equipment.

trim controls the full horizontal tail, and elevator only a tab on it. At higher speeds, elevator cannot overrule the tail.

The airplane was now so far out of trim that Getachew had to hold his control column halfway back (meaning half up-elevator) to keep the nose from dropping. It was urgent to retrim the airplane using the manual trim wheel. But there was a big problem: The pilots had still not throttled back from takeoff thrust, and the airplane now in level flight was going extremely fast, at least 25 knots faster than the maximum operating speed of 340 knots, and was rapidly accelerating into realms beyond the flight-testing range. The excessive speed was amply clear in the cockpit, where an overspeed clacker was sounding off, but neither pilot thought to reduce the thrust and slow. The flight-data recorder later showed that the throttles remained at a takeoff setting until the end.

The speed, meanwhile, was producing such large aerodynamic forces on the tail that the manual trim wheel lacked the mechanical power to overcome them, and the trim was essentially locked into the position where the MCAS had left it — not fully nose-down, but dangerously out of whack. The only way to retrim the airplane at these speeds would be to use the much more powerful electrohydraulic mechanism associated with the thumb switches, which, however, would require the pilots first to flip the cutout switches back to “normal,” exposing the airplane to further attacks from the MCAS.

This airplane had heavy pressures on the controls — remember, Getachew was muscling his control column halfway back. Now, in apparent desperation to persuade the autopilot to engage, Getachew did the unthinkable and released his pressure on the control column. The column snapped forward, and the airplane responded by violently pitching down, 20 degrees below the horizon. Just then, with the stick shaker still rattling, the MCAS kicked in and achieved full nose-down trim, doubling the angle of the dive. As the speed shot through 450 knots, the pilots hauled back on their control columns to no effect. Six minutes after takeoff, the airplane hit the ground doing approximately 600 miles an hour. It buried itself into a 30-foot-deep crater in farmland about 32 miles southeast of the airport. Within a week, the Boeing 737 Max was grounded worldwide.

They would have had to remember to cut power before it was too late, but they were paniced. Once the nose was pointed down enough, even reducing power would have done nothing, there was not enough elevator control to counteract the trim. **Turning autopilot back on would likely have saved the airplane, but a pilot is trained to never trust the AP when things are going wrong. They had no way to know it could have helped.**

737 Max, more articles

How the Boeing 737 Max Disaster Looks to a Software Developer
“Design shortcuts meant to make a new plane seem like an old, familiar one are to blame”
https://spectrum.ieee.org/aerospace/aviation/how-the-boeing-737-max-disaster-looks-to-a-software-developer

What Really Brought Down the Boeing 737 Max?
https://www.nytimes.com/2019/09/18/magazine/boeing-737-max-crashes.html

Aviation teaching from crashes

- If you are interested in aviation, a few videos you can watch at home:
- https://www.youtube.com/watch?v=WfNBmZy1Yuc
- https://www.youtube.com/watch?v=5ESJH1NLMLs
- https://www.youtube.com/watch?v=FF2Fhi-TT0o
- https://www.google.com/search?q=Captain.+Warren+Vander Burgh

What about places without all that fancy training

- Those aviation talks are paramount to pilot pre-training in case of an unexpected emergency in the future
- Not all countries have such training. Some countries fly you around with pilots with less flying time than me, and I’m just a recreational pilot
- Read up how safe/unsafe flying is in some countries. My dad ended up in a 4 engine indian plane that glided as all engines stopped getting fuel for a while
- If praying is not a strategy, rely on superior automation computers (Airbus) and hope they don’t fail and pilots don’t manage to mismanage fuel
- Can you find parallels with computers?

When things unexpectedly go wrong

- Undo whatever was done last
- It doesn’t always work, but it has saved many airplanes
- It often works for computers too
- Never spend time arguing whose fault it is or why it should or shouldn’t be happening
- Revert first, talk later!
- If revert doesn’t help, try to fall back to pre-learned recoveries. Impressive recovery of a plane that was doomed to kill everyone: https://youtu.be/WfNBmZy1Yuc?t=2031

Conclusions

- Learn from other people’s experiences/mistakes.
- Continuous improvement, read tech magazines from your field (sysadmin, programming, pilot, etc...)
- If you see something, say something! Be a whistle blower!
- Think ahead for problems, grow your spidey sense
- Don’t let people bully you out of your real concerns. Maybe you won’t have a space shuttle blowup on your conscience for a problem you pointed out and was ignored, but don’t let things go if you know they are wrong.
- Be honest with yourself, know if you are pushing past safety limits or taking too many risks. Be sure you can explain what you did or didn’t with a straight face.
- For aviation, wouldn’t it be nice if it were all open source so that researchers can help the FAAs of the world check for bugs? Did Airbus ever fix some of the software issues I mentioned? (never mind Boeing)

Q&A

Slides:
http://marc.merlins.org/linux/talks/Failure_Planning/

Marc Merlin
merlin@google.com