ht
shed
e
are
,
from
own
MAX
us

AND (Aircraft Nose Down) commands. According to Wikipedia:

> *"In the next 10 seconds the trim moved back up to 2.3 units as a result of pilot input and the pilots agreed on and executed the stabilizer trim cut-out procedure, cutting power to the trim motor operated by MCAS".*

After the 2nd accident, the Boeing 737 Max was grounded world-wide, and over $6 billion dollars evaporated from the company's market cap virtually overnight.

The short-term impact is just the beginning, though. If Boeing doesn't fix the crisis effectively and restore the public trust, it

could materially impact the long-term stability of the company. According to Goldman Sachs, Boeing 737 sales account for 33 percent of Boeing's projected sales for the next five years.

The good news is, according to a Boeing official speaking to CNN Business:

> "We believe this can be updated through a software fix."

It seems that the MCAS system can fail and send the aircraft into a steep nose dive from which pilots are unable to recover. Whether this flaw is software or microprocessor hardware, it's a computing system failure of a life-critical system. Normally, for such systems, it's not uncommon to employ many lines of defense in order to assure quality control.

Some common lines of defense in software development include:

- Requirements Gathering
- Requirements Specification & Review
- Risk Analysis
- Test Driven Development (TDD)
- Software Linting
- Static Analysis
- Software Inspection / Code Review
- Model Simulation

- Formal Methods
- Acceptance Testing

I primarily write consumer applications governing *non-life-critical* systems for millions of users. All the teams I have led or been a part of recently have used systems for requirements gathering, specification review (using technical Readme Driven Development with peer review), TDD, production simulation using Continuous Integration/Continuous Delivery (CI/CD) to ensure that the software will work in the production environment, linting, static analysis, code review (a trimmed-down version of software inspection), and acceptance testing.

We invest heavily in software quality, and pay even our most junior engineers at least $100k/year no matter where they live in the world in order to attract the best software engineers we can find to produce high quality software that is easy to maintain.

We do employ junior developers but we ensure that there are always enough senior level engineers to provide mentorship, code review, and support. We also invest heavily in training and mentorship to ensure that every engineer on our teams are supported with the understanding they need to build reliable software systems.

If I were in charge of a life-critical system, I would double down on quality and consider adding more thorough training, mentorship, requirements gathering, risk assessment, software inspection (e.g., employ a language-relevant and data-backed variation of NASA's Power of 10 rules as a code inspection checklist), static analysis, and even formal methods for critical systems to mathematically prove that our algorithms are correctly specified.

If I had Boeing's budget, I'd also invest in research to see if the selection of programming language could make a difference. In my experience, investing in mentorship and the other quality measures I've already discussed makes a much bigger difference than language selection, but with life-critical systems, a data-driven decision that could reduce the likelihood of defects even a little bit could be the difference between life and death. We need better research on software quality, and with a market cap in excess of $200 billion to protect, I'd think funding these kinds of studies would be a no-brainer for Boeing.

We don't invest heavily in software quality because our teams have big budgets (they're a drop in the ocean compared to Boeing's engineering budget). We invest heavily in software

quality because it helps us move faster and save money in the long-run.

> In software, slow is fast.

Fast is Slow and Cheap is Expensive, So Why Are So Many Managers Cheap?

To cut costs, engineering managers often rush developers, impose arbitrary unrealistic deadlines, or in the case of Boeing, outsource engineering to cheap contractors to try to increase production bandwidth.

Boeing's cultural emphasis on cost savings seems to have trickled all the way down to the engineers working on the 737. One 737 contract software engineer from HCL, an Indian company Boeing outsourced to, illustrates the cost cutting culture on his resume:

> *"Provided quick workaround to resolve production issue which resulted in not delaying flight test of 737-Max (delay in each flight test will cost very big amount for Boeing)."*

Regardless of which systems these engineers worked on, if you foster an engineering culture where developers are made to feel that management values fast and cheap over continuous progress and quality software, that will have a

tremendously negative impact on both the quality and the timely delivery of your software. Buggy software takes longer to build.

When management over-emphasizes cost savings, developers feel rushed. When developers feel rushed:

- Mentorship & reviews halt
- Bugs pile up
- Tests get skipped
- Communication suffers
- Developers burn out
- Productivity suffers

Mark Rabin, who worked in a Boeing flight-test group that supported the 737 Max, told Bloomberg that the decision to outsource engineering to HCL was “controversial because it was far less efficient than Boeing engineers just writing the code. […] It took many rounds going back and forth because the code was not done correctly.”

According to Rabin, one manager shared at an all-hands meeting that Boeing didn’t need senior engineers because its products were mature. That anecdote is what prompted me to write this article. In case there are any doubts, letting junior engineers code without senior engineers to review their work

and guide them is equivalent to stashing time bombs all over the codebase.

When it comes to engineering, fast is slow, and cheap is expensive.

Everybody seems to know it’s true, but very few companies are good at putting it into practice: particularly when they’re under pressure and they need it the most.

Boeing was facing a serious competitive threat from their rival Airbus. In 2010, Airbus announced the new A320neo which was 7% more fuel efficient, and sold more in a week than Boeing 737 sales for the entire year of 2010.

Boeing needed to answer back, fast. To make the offering attractive to airlines, they decided to base their answer on the existing 737, which both saved the time of designing a new platform from scratch, and also locked them into a legacy platform with similar controls and instrumentation to all the previous models, dating back to the 1960’s. That choice was also an important sales talking point for airlines who would need to re-certify pilots if they made too many changes. This would be a serious engineering challenge.

To avoid losing too much ground to Airbus, they decided they would try to make the new 737 available just a few months after Airbus shipped the A320neo. They gave themselves 6

years to deliver the new model, which shaved more than a year off the time it took them to ship the 787 and 777.

In other words, they started the project with an optimistic deadline in mind, but instead of making it aspirational, they committed to it hard. Before they even began, they had locked all of the constraints of the project management triangle.

Scope was constrained by the ambitious engineering goals that were preset by the competition. Time was locked in order to avoid a default win by Airbus. That meant that the only lever they had left to ensure quality was to ensure that the project had an adequate budget. But the culture of Boeing was already locked into a cost cutting mandate. The only thing left to give was quality.

The problem with cost cutting culture is that companies like Boeing get so obsessed with it that it ends up costing them far more than they save. Boeing's initial budget for the 737 Max was $3 billion. In spite of the efforts to save time and money, they landed several billion dollars over budget and delivered late.

The ensuing disasters cost them tens of billions in market cap, $30 billion in airline order cancellation threats, and unmeasurable damage to the Boeing brand. They could have

invested billions of dollars more to follow better engineering practices and still come out far ahead.

This all serves as a great lesson for all software managers and engineers: In software, fast is slow and cheap is expensive.

> An investment in quality is an investment in productivity, cost savings, and stronger sales.

What could have been a huge win for Boeing had they not been obsessed with cost cutting turned into a big disaster that took a hatchet to the Boeing brand.

Don’t repeat Boeing’s mistakes.

Related Articles for JavaScript Developers

If you’re a JavaScript developer, these resources can help you improve your software quality process:

- TDD Changed My Life
- TDD Day — A day dedicated to TDD training
- Streamline code reviews with ES Lint and Prettier
- The Essential Guide to Building Balanced Development Teams

Eric Elliott *is the author of the books, “Composing Software” and “Programming JavaScript Applications”. As co-founder of EricElliottJS.com and DevAnywhere.io, he teaches developers essential software development skills. He builds*

and advises development teams for crypto projects, and has contributed to software experiences for ***Adobe Systems, Zumba Fitness, The Wall StreetJournal, ESPN, BBC,*** *and top recording artists including* ***Usher, Frank Ocean, Metallica,*** *and many more.*