The Boeing 737 MAX Saga: Lessons for Software Organizations

PHILLIP JOHNSTON AND ROZI HARRIS

In March 2019, Boeing 737 MAX 8 and MAX 9 aircraft were grounded in more than 41 countries, including the United States, Canada, and China, after an Ethiopian Airlines crash resulted in the deaths of everyone on board. This was the second deadly crash involving a Boeing 737 MAX. A Lion Air (Indonesia) Boeing 737 MAX 8 crashed in October 2018, also killing everyone on board. As a result of these two incidents, Boeing paused delivery of these planes, although production continues. The Boeing 737 MAX story provides an alarming case study on the interrelationships between software and systems engineering, human factors, corporate behavior, and customer service. Although both crashes are still under investigation, and definitive answers are forthcoming, this article examines the events from the perspectives of safety and software quality and proposes five lessons practitioners can apply to their own projects to mitigate risk.

KEY WORDS

complex systems, risk, risk mitigation, safety, software quality, systems thinking

INTRODUCTION

The Boeing 737 is the best-selling aircraft in the world, with more than 15,000 planes sold. After Airbus announced an upgrade to the A320 that provided 14 percent better fuel economy per seat, Boeing responded with the 737 MAX. It was marketed as an upgrade to the famed 737 design, using larger engines to match the improved fuel efficiency from Airbus. Boeing claimed the 737 MAX was so similar to the original 737 that pilots already licensed for this aircraft would not need additional training and simulator time for the 737 MAX.

Because Boeing increased the engine size to improve fuel efficiency, the engines had to be positioned higher on the wings and slightly forward of the old position. Higher nose landing gear was also added to provide the same ground clearance. The larger engines and new positions destabilized the aircraft, but not under all conditions. The engine housings are designed so they do not generate lift in normal flight. However, if the airplane is in a steep pitch (for example, takeoff or a hard turn), the engine housings generate more lift than on previous 737 models. Depending on the angle, the airplane's inertia can cause the plane to over-swing into a stall.

To address increased stall risk, Boeing developed a software solution called the maneuvering characteristics augmentation system (MCAS), which is part of the flight management computer software. No other commercial plane uses software like it, though Boeing uses a similar system on the KC-46 Pegasus military aircraft. The MCAS takes readings from the angle of attack (AoA) sensor to determine how the plane's nose is pointed relative to

the oncoming air. MCAS monitors airspeed, altitude, and AoA. When the software determines the angle of attack is too great, it automatically performs two actions to prevent a stall:

1. Command the aircraft's trim system to adjust the rear stabilizer and lower the nose.
2. Push the pilot's yoke in the down direction.

The movement of the rear stabilizer varies with the speed of the plane; it moves more at slower speeds and less at higher speeds. By default, the MCAS is active when autopilot is off and AoA is high or when the flaps are up. The MCAS will deactivate once the AoA measurement is below the target threshold, or the pilot overrides the system with a manual trim setting, or the pilot engages the CUTOUT switch (which disables automatic control of the stabilizer trim). If the pilot overrides the MCAS with trim controls, it will activate again within five seconds after the trim switches are released if the sensors still detect an AoA over the threshold. The only way to completely disable the system is to use the CUTOUT switch and take manual trim control.

Boeing designed the MCAS so it would not turn off in response to a pilot manually pulling the yoke. This would defeat the original purpose of the MCAS, which is to prevent the pilot from inadvertently entering a stall angle. This is significant because a pilot's natural reaction to a plane that is pitching downward is to pull on the yoke and apply a counter-force to correct for the unexpected motion. For normal autopilot trim or runaway manual trim, pulling on the yoke does what is expected and triggers trim hold sensors.

People are under the impression that the column, yoke, steering wheel, gas pedal, and brakes fully control the response of the mechanical system. This is an illusion. Modern aircraft, like most modern cars, are "fly-by-wire." Gone are the days of direct mechanical connections involving cables and hydraulic lines. Instead, most of the connections are electrical and mediated by a computer. In many ways, users are being continually "guarded" by the computers that mediate these connections. It can be a terrible shock when the machine fights against the user.

THE SUSPECTED PROBLEM

The MCAS is suspected to have played a significant role in both crashes. During Lion Air flight JT610, MCAS repeatedly forced the plane's nose down, even when the plane was not stalling. The pilots tried to correct it by pointing the nose higher, but the system kept pushing it down. This up-and-down oscillation happened 21 times before the crash occurred. The Ethiopian Airlines crash showed a similar pattern. The company's CEO believes the MCAS was active during the Ethiopian Airlines crash.

If the plane was not actually stalling, or even close to a stall angle, why was MCAS engaged? AoA sensors can be unreliable, a factor suggested in the Lion Air crash, where there was a 20-degree discrepancy in the plane's two AoA sensor readings. The MCAS only reads the AoA sensor on its active corresponding side of the plane and does not cross-check the other sensor to confirm the reading. If a sensor malfunctions, the MCAS has no way to know.

If the MCAS was enabled erroneously, why did the pilots not disable the system? This is where the situation becomes muddled. The likeliest explanation for the Lion Air pilots is that they had no idea the MCAS existed, that it was active, or how they could disable it. MCAS is a unique piece of software among commercial airplanes; it only runs on the 737 MAX. Boeing sold and certified the 737 MAX as a minor upgrade to the 737 body, which would not require pilots to re-certify or spend time training in simulators. As a result, it seems the existence of the MCAS was largely kept quiet.

After the Lion Air crash, Boeing released a bulletin providing details on how the MCAS system worked and how to counteract it in case of malfunction. The company also stated that emergency procedures that applied to earlier models would have corrected the problems observed in the Lion Air crash. The Lion Air pilots, though, likely fought against an automated system that was working against them. The system is most likely to activate at low altitudes, such as during takeoff, leaving the pilots little time to react. The Ethiopian Airlines pilots had heard about MCAS thanks to the bulletin, although one pilot commented, "We know more about the MCAS system from the media than from Boeing" (Fick and Neely 2019). Ethiopian Airlines installed one of the first simulators for the 737 MAX, but the pilot of the doomed flight had not yet received training in the simulator. All the authors know at the time of this writing is that the pilot reported "flight control problems" and wanted to return to the airport, and that the second crash resembled the first. They must wait for the preliminary report for more details.

COMPOUNDING FACTORS

The initial analyses suggest that the MCAS software system was poorly designed and caused two plane crashes. But this is a complex situation, involving many people and organizations. In addition, other pilots had successfully struggled against the MCAS system and safely guided their passengers to their destination. Four contributing factors, observed in the Boeing case, have also been observed in other catastrophic software failures. They are: poor documentation, rushed release, delayed software updates, and humans out of the loop.

Poor Documentation

After the Lion Air crash, pilots complained that they had not been told about the MCAS or trained in how to respond when the system engages unexpectedly. This lack of documentation and training is especially dangerous when automated systems are involved and previous training does not fully apply. Tragically, black box recordings indicate Lion Air pilots frantically attempted to find answers in the manuals before they crashed.

Pilots take their documentation extremely seriously. Following are three reports from the Aviation Safety Reporting System (ASRS), which is run by NASA to provide pilots and crews with a way to confidentially report safety issues. Three reports highlighted next focus on the insufficiency of Boeing 737 MAX documentation.

> **ACN 1593017:** 737 MAX Captain expressed concern that some systems such as the MCAS are not fully described in the aircraft flight manual.
>
> "I think it is unconscionable that a manufacturer, the FAA, and the airlines would have pilots flying an airplane without adequately training, or even providing available resources and sufficient documentation to understand the highly complex systems that differentiate this aircraft from prior models. The fact that this airplane requires such jury rigging to fly is a red flag. Now we know the systems employed are error prone--even if the pilots aren't sure what those systems are, what redundancies are in place, and failure modes. I am left to wonder: what else don't I know? The flight manual is inadequate and almost criminally insufficient. All airlines that operate the MAX must insist that Boeing incorporate ALL systems in their manuals."
>
> **ACN 1593021:** 737 MAX Captain reported confusion regarding switch function and display annunciations related to "poor training and even poorer documentation."
>
> "This is very poorly explained. I have no idea what switch the preflight is talking about, nor do I understand even now what this switch does. I think this entire setup needs to be thoroughly explained to pilots. How can a captain not know what switch is meant during a preflight setup? Poor training and even poorer documentation, that is how. It is not reassuring when a light cannot be explained or understood by the pilots, even after referencing their flight manuals. It is especially concerning when every other MAINT annunciation means something bad."
>
> **ACN 1555013:** 737 MAX First Officer reported feeling unprepared for first flight in the MAX, citing inadequate training.
>
> "I had my first flight on the MAX… My post flight evaluation is that we lacked the knowledge to operate the aircraft in all weather and aircraft states safely. The instrumentation is completely different - my scan was degraded, slow and labored… we were unable to navigate to systems pages and lacked the knowledge of what systems information was available to us in the different phases of flight. Our weather radar competency was inadequate to safely navigate significant weather on that dark and stormy night. These are just a few issues that were not addressed in our training."

Rushed Release

Tight deadlines and rushed releases are not uncommon. When presented with a contract deadline or other similar requirement, the tendency can be to cut corners, make concessions, and ignore or mask problems — all to release a product by a specific date so the company does not lose business. At times like this, problems can be downplayed, and when they are observed by the customer, the work is deferred to a patch.

Apparently, the 737 MAX project was subject to the same treatment. Gates (2019) reported the 737 MAX was nine months behind the new A320neo. Boeing managers had prodded engineers to speed up the production process, and if there wasn't time for FAA staff to complete a review, FAA managers either signed off on the documents themselves or delegated the review to Boeing. The FAA explained this by noting a lack of funding and resources to carry out due diligence.

As a result of this rushed process, a major change slipped through; the system safety analysis on MCAS claimed the horizontal tail movement was limited to 0.6 degrees. Boeing engineers later found this number to be insufficient for preventing a stall in worst-case scenarios, so it was increased by a factor of four. The FAA was never informed of this engineering change, and FAA engineers did not learn about it until Boeing released the MCAS bulletin following the Lion Air crash.

Gelles et al. (2019) corroborated the miscommunication and the details of the rushed release: "The pace of the work on the 737 Max was frenetic…the timeline was extremely compressed…it was go, go, go." The workload, according to one designer, was double the norm. Engineers were under tremendous pressure, which is associated with increased levels of errors.

Delayed Software Updates

Weeks after the Lion Air crash, Boeing officials told the Southwest Airlines and American Airlines pilot's unions that they planned to have software updates available around the end of 2018. However, the FAA reported work on evaluating the new MCAS software was delayed for five weeks by the government shutdown (Pasztor and Tangel 2018).

Consumers are conditioned to wait for fixes and updates. Teams are prone to idealistic estimates, and problems take longer than expected to diagnose, correct, and validate. As a result, schedules are repeatedly overrun. Any of these expected behaviors can have dire consequences. For example, an "enhancement" was submitted to the FAA for certification on January 21, only four days before the shutdown ended. This software update, submitted four months later than the initial estimate, will still require many more months to test, approve, and deploy. It is no comfort to the families of those who lost their lives on Ethiopian Airlines Flight 302 that Boeing submitted a software fix for certification seven weeks before this fatal crash.

There is a real cost to the delay of software updates, and that cost increases significantly with the impact of the issue. It is always better to take the necessary time to implement a robust design, incurring as many internal failures as needed, to avoid external failures that will require a patch.

Humans Out of the Loop

The MCAS was designed to use a single data point from the AoA sensor on the active side of the plane, and the initial NTSC report on the Lion Air crash (ASN 2018) reveals that a single faulty AoA sensor triggered the MCAS. If a pilot or co-pilot noticed a strange AoA reading (such as a 20-degree difference between the two AoA sensors), he or she could perform a cross check by glancing at the reading on the other side of the plane. Additional sensors and gauges can be read to corroborate or disprove a strange AoA reading, and visual corroboration is also possible. What is even more troubling is that the system's behavior was opaque to the pilots. According to Boeing, the MCAS is (counterintuitively) only active in manual flight mode and is disabled under autopilot. MCAS controls the trim without notifying the pilots it is doing so.

Boeing did provide two optional features that would provide more insight into the situation: 1) an AoA indicator, which displays the sensor readings; and 2) an AoA disagree light, which lights up if the two AoA sensors disagree. Because these were optional and carried an additional cost, many carriers did not elect to buy them.

Computers can only perform pre-programmed actions, and a computer cannot take in additional data it was not programmed to read. Despite advances in deep learning, humans remain superior to computers in synthesizing complex information. For complex and safety-critical systems, humans must maintain the ability to override or overpower an automated process.

Boeing's Response

A rapid response was triggered on behalf of Boeing to correct the issues and address public safety. Fehrm (2019) reported on the MCAS software update:

"Boeing has developed an MCAS software update to provide additional layers of protection if the AoA sensors provide erroneous data. The software was put through hundreds of hours of analysis, laboratory testing, verification in a simulator and two test flights, including an in-flight certification test with Federal Aviation Administration (FAA) representatives on board as observers."

The update includes several significant changes, including:

- The flight control system will now compare inputs from both AoA sensors.
- If the sensors disagree by 5.5 degrees or more with the flaps retracted, MCAS will not activate automatically.
- An indicator on the flight deck display that

alerts the pilots to AoA disagree condition will now ship as a standard feature.

- MCAS can never command more stabilizer input than can be counteracted by the flight crew pulling back on the yoke.
- The pilots will continue to always have the ability to override MCAS and manually control the airplane.

In addition to the software changes, extensive training changes are being made. Pilots will have to complete 21 or more days of instructor-led academics and simulator training. Computer-based training that includes MCAS, crew procedures, and related software changes will be made available to all 737 MAX pilots. Pilots will also be required to review and acknowledge new operations manuals, checklists, and quick reference documents. Undoubtedly, Boeing internal reviews will examine why these elements were not part of earlier releases.

IS THE PROBLEM BAD SOFTWARE?

It is tempting to label the 737 MAX issues as "caused by software." At some level, this is true. However, the MCAS appears to be characterized by a quickly applied software patch introduced within a larger systems context:

1. Fuel is expensive, and more efficient engines will reduce that burden.
2. Competitive pressure from Airbus placed pressure on Boeing to respond with its own improved platform.
3. The timeline was largely dictated by Airbus, not the time Boeing engineers needed to complete the project.
4. Boeing wanted to stick to the 737 platform to achieve faster time to market, lower cost for production and certification, and improved pilot familiarity (meaning lower training costs).
5. Bigger engines did not fit on the existing 737 platform, so design modifications were needed.
6. These modifications changed the aerodynamics of the airplane, which should have driven additional requirements for certification and training.
7. Instead, Boeing created MCAS software to compensate for the aerodynamic impact of the new design, downplayed its novelty, and generated insufficient documentation and pilot training.

This is a systems engineering problem created by the company's design goals. Redesigning the airplane was out of the question because it would give Airbus a significant time advantage, necessitate expensive training (making the platform less attractive to buyers), and incur certification changes. To meet the design goals and avoid an expensive hardware change, Boeing created the MCAS as a software Band-Aid.

Applying software workarounds for silicon or hardware design flaws is a major part of the work of firmware developers. Fixing hardware is expensive in terms of both time and money, and at some point, it becomes too late to change the hardware. The schedule drives the decision to move forward with known hardware design flaws. The next line is predictable: "The problem will just have to be fixed in software." But software fixes do not always work. When the software workaround fails, companies seem to forget that they were already attempting to hide a problem.

The authors are not alone in the view that this is not a "software problem." Sumner and Kammeyer (2019), the latter a pilot, explained that on both ill-fated flights, the following problems were experienced, each compounded by the next:

- **Sensor problem.** The AoA vane on the 737 MAX appears not to have been reliable and gave wildly incorrect readings.
- **Maintenance practices problem.** The previous crew had experienced the same problem and didn't record the problem in the maintenance logbook.
- **Pilot training problem**. On Lion Air, pilots were never even told about the MCAS, and by the time of the Ethiopian flight, an emergency bulletin had been issued, but no one had done simulator training on this failure.
- **Economic problem.** Boeing sells an option package that includes an extra AoA vane and disagree light, which would let pilots know that this problem was happening. Both airplanes that crashed were delivered without this option. No 737 MAX with this option has ever crashed.

- **Pilot practice problem.** If the pilots had correctly and quickly identified the problem and run the stab trim runaway checklist, they would not have crashed.

Their closing point is austere: "Nowhere in here is there a software problem. The computers and software performed their jobs according to spec without error. The specification was just ****ty. Now the quickest way for Boeing to solve this mess is to call up the software guys to come up with another Band-Aid."

What would prevent this cascade? Maybe the idea of a safety culture? When optimizing for time-to-market and reduced costs, will safety ever be a priority? After the resulting deaths, loss in market position, and destruction of trust, one must wonder if Boeing will ever realize the time and cost savings it hoped the software fix would provide.

LESSONS LEARNED

A complex system operated in an unexpected manner, and 346 people are dead as a result of two tragic and catastrophic accidents. Though the lives cannot be restored, if many systems and software engineers can learn as much as possible about this case, such deaths can be prevented in the future.

People Cannot Bend Complex Systems to Their Will

Boeing took an existing system and tried to change it to force a specific economic and time-bound outcome. But all changes to complex systems have unintended consequences. Meadows (2001) explains "self-organizing, nonlinear, feedback systems are inherently unpredictable…not controllable…understandable only in the most general way…. For any objective other than the most trivial, we can't optimize; we don't even know what to optimize. We can't keep track of everything." Although complex systems cannot be controlled, they can be designed, redesigned, and nudged. Through observation and interaction, people can learn how to work with and within these systems.

These points are echoed by Ackoff (1994): "A system is not the sum of the behavior of its parts, it is a product of their interactions. The performance of a system depends on how the parts fit, not how they act taken separately." Boeing changed a few parts of the plane and expected overall performance to be improved, but the effect on the overall system was more complex than it anticipated.

A system cannot be optimized by optimizing the individual parts. Similarly, eliminating an unwanted component or defect does not guarantee an improvement in system performance. Most software engineers are familiar with the experience of fixing a bug, only to have a new bug (or several bugs) appear as a result of the initial fix. Finding and removing defects is not a way to improve the overall quality or performance of a system.

The larger engines on the 737 airframe resulted in undesirable aerodynamic characteristics (excessive upward pitch at steep AoA). Boeing responded by attempting to address this defect with the MCAS, but the MCAS does not unilaterally improve the overall quality or performance of the aircraft. Software and engineering professionals can address these challenges by asking what aspects of a system are they trying to force. By broadening one's perspective and examining different approaches at the system level, a better solution may be discovered.

Set the Right Aim

Boeing's aim was to keep up with Airbus, leading to an aggressive time-to-market. It also wanted to minimize design changes to ease certification and ensure that pilots did not need to receive new training. Those are the principles that appear to have guided the company's actions. Safety was still a concern, but it was not the focus of the organization, system, or schedule.

Ackoff (1994) explains that "an improvement program must be directed at what you want, not at what you don't want." On one level, Boeing aimed for a new aircraft with improved fuel efficiency to compete with Airbus. At the same time, Boeing wanted to do this on a timeline that would not delay the company significantly with regard to the Airbus launch. Boeing also wanted to minimize crew training requirements, which further constrained the design. The focus was on the things it did not want. A focus on safety might have led to a redesigned airframe that would support larger engines, albeit at greater time-to-market.

Treat Documentation as a First-Class Citizen

If other people will use a product, the developer of the product needs to provide useful and comprehensive documentation and training. This is extremely important not only to the users, but also to the engineers and managers who will maintain and develop the product in the future.

Documentation also promotes deeper understanding: if a person cannot explain something in simple terms, then that person does not understand it. If it is not explained, nobody has a chance of understanding it.

In aviation, pilots are fanatical about documentation, and for good reasons. In this case, improved documentation would have led to a better understanding of the system forces at work, and alone could have potentially saved hundreds of lives. But one high-ranking Boeing official said the company had decided against disclosing more details to cockpit crews due to concerns about inundating average pilots with too much information – and significantly more technical data – than they needed or could digest (Pasztor and Tangel 2018).

Software teams often take this view of their users. Perhaps it is simply a rationalization for not wanting to put the effort into creating and maintaining documentation. How can one predict what information people need to know? What is too technical? What is enough information? Won't the details change as the system evolves? How will the documentation be maintained? Software teams hinder themselves when they neglect documentation. The documentation process can help new team members learn how a system is designed and functions. Ideas for simplification will reveal themselves, and novel ways to use the software in ordinary cases and edge cases will be encountered. Most importantly, poorly understood system aspects can suddenly become obvious.

Keep Humans in the Loop

People must maintain the ability to override or overpower an automated process. Especially when trained properly, humans excel at dynamic information collection and synthesis, and can improvise and make novel decisions in response to new situations. A computer, which has been preprogrammed to read from a limited amount of information and perform a set of specific responses, is not capable of improvising.

System designers and programmers are not all-knowing. When translating from design to operations, humans should be kept in every feedback loop. Operators should be able to override automated processes when and if they determine it is necessary to do so. "What we have here is a 'failure of the intended function,'" remarked industry research analyst Colin Barnden. "A plane shouldn't fight the pilot and fly into the ground. This is happening after decades of R&D into aviation automation, cockpit design and human factors research in planes" (Yoshida 2019).

Testing Doesn't Keep People Safe

Koopman, Kane, and Black (2019) recently described "The Insufficient Testing Pitfall." They note: "Testing less than the target failure rate doesn't prove you are safe. In fact, you probably need to test for about 10x the target failure rate to be reasonably sure you've met it. For life critical systems this means too much testing to be feasible."

There is no doubt the airplane and all its software were intricately and extensively tested. In addition, the software would have been extensively leveraged in simulators and in test flights. But it seems Boeing did not test the system enough to encounter these problems. And even if it did, what other problems would still be missed? Companies need better plans for proving their software works safely. Reliance on testing is not enough.

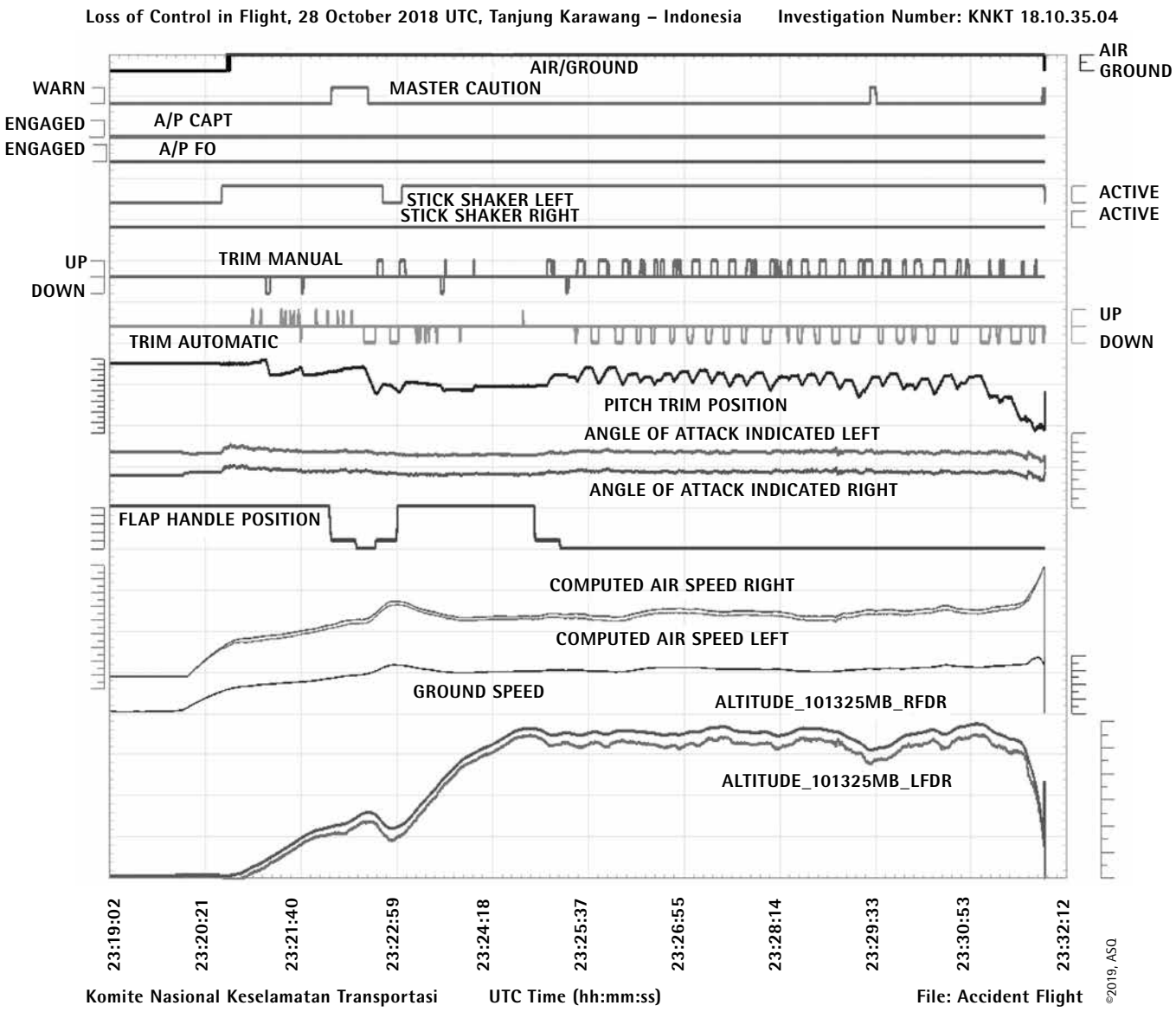
This Can Happen in Other Organizations

It's easy for people to read about these disasters, or any other similar incidents caused by humans, and think they personally would never have let such a thing happen. But most engineers have worked on fast-paced engineering projects and have had to make compromises to meet deadlines. Some compromises may have been personally initiated, while others were suggested or ordered by management. Regardless of the source, catastrophic events have far-reaching psychological and emotional consequences for those who are closely involved. After the Lion Air crash, Boeing offered trauma counseling to engineers who had worked on the plane. A technical manager reported that his staff were devastated.

One must also remember that nobody at Boeing wanted to trade human lives for increased profits. All human organizations — including families, companies, industries, governments — are complex systems and have a life of their own. Despite individual beliefs and priorities, organizations can make and execute decisions that none of the participants truly want, such as shipping a compromised product or prioritizing profits over safety.

The organization rallied around the goals of time-to-market and minimizing required pilot training. Momentum and inertia kept the company marching toward its aim, even if individuals disagreed. The deprioritization of safety was likely silent: there was no

FIGURE 1 While the automatic trim (Trim Automatic in the below figure) was forcing the aircraft down, the pilots countered by pointing it back up (see Trim Manual above Trim Automatic). (KNKT 2018)



villainous CEO forcing his or her minions to compromise the product. There was not an entire organization whose individuals decided to collectively disregard safety.

Boeing made the same kinds of incremental decisions that are being made in many other organizations on a daily basis. Everyone has a duty to aim higher.

ACKNOWLEDGEMENTS

The authors wish to thank Stephen Smith for reviewing early drafts of this essay. His feedback and discussion points were incorporated into the version of this essay at https://embeddedartistry.com/blog/2019/4/1/what-can-software-organizations-learn-from-the-boeing-737-max-saga. In addition, thanks are offered to the journalists and aviation enthusiasts who have published brilliant coverage and analysis for the 737 MAX case from which the material herein has been distilled.

REFERENCES

Ackoff, R. L. "Beyond Continual Improvement." Lecture given at W. Edwards Deming: Learning and Legacy, 1994.

Aviation Safety Network (ASN). 2018. NTSC Indonesia publishes preliminary report on JT610 Boeing 737 MAX 8 accident. Available at: https://news.aviation-safety.net/2018/11/28/ntsc-indonesia-publishes-preliminary-report-on-jt610-boeing-737-max-8-accident/.

Fehrm, B. 2019. Boeing presents MCAS fix to pilots, regulators and media. Leeham News and Analysis. Available at: https://leehamnews.com/2019/03/27/boeing-presents-mcas-fix-to-pilots-regulators-and-media/.

Fick, M., and J. Neely. 2019. Ethiopia crash captain did not train on airline's MAX simulator: source. Reuters (March 21). Available at: https://www.reuters.com/article/us-ethiopia-airplane-simulator-exclusive/ethiopia-crash-captain-did-not-train-on-airlines-max-simulator-source-idUSKCN1R20WD.

Gates, D. 2019. Flawed analysis, failed oversight: How Boeing, FAA certified the suspect 737 MAX flight control system. Seattle Times (March 17). Available at: https://www.seattletimes.com/business/boeing-aerospace/failed-certification-faa-missed-safety-issues-in-the-737-max-system-implicated-in-the-lion-air-crash/.

Gelles, D., N. Kitroeff, J. Nicas, and R. R. Ruiz. 2019. Boeing was 'Go, Go, Go' to beat Airbus with the 737 Max. The New York Times (March 23). Available at: https://www.nytimes.com/2019/03/23/business/boeing-737-max-crash.html.

Komite Nasional Keselamatan Transportasi (KNKT). 2018. PRELIMINARY KNKT.18.10.35.04 Aircraft Accident Investigation Report, Pt. Lion Mentari Airlines, Boeing 737-8 (MAX); PK-LQP Tanjung Karawang, West Java, Republic of Indonesia. Available at: https://reports.aviation-safety.net/2018/20181029-0_B38M_PK-LQP_PRELIMINARY.pdf.

Koopman, P., A. Kane, and J. Black. 2019. Credible Autonomy Safety Argumentation. In Proceedings of the 27th Safety-Critical Systems Symposium, February. Available at: https://users.ece.cmu.edu/~koopman/pubs/Koopman19_SSS_CredibleSafetyArgumentation.pdf.

Meadows, D. "Dancing With Systems. "Whole Earth Review 106 (Winter 2001): 58-63.

Pasztor, A., and A. Tangel. 2018. Boeing withheld information on 737 model, according to safety experts and others. Wall Street Journal (November 13). Available at: https://www.wsj.com/articles/boeing-withheld-information-on-737-model-according-to-safety-experts-and-others-1542082575.

Sumner, T., and D. Kammeyer. 2019. BEST analysis of what really is happening on the #Boeing737Max issue from my brother in law @davekammeyer, who's a pilot, software engineer & deep thinker. Available at: https://threadreaderapp.com/thread/1106934362531155974.html.

Yoshida, J. 2019. Boeing 737 Max: Is automation to blame? EET Asia (March 19). Available from https://www.eetasia.com/news/article/Automation-and-Boeings-B737-Max-Crash.

BIOGRAPHIES

Phillip Johnston and **Rozi Harris** are principals at Embedded Artistry, an embedded systems consulting firm focused on improving early-stage hardware product development. They build a solid foundation for new products with systems architecture, modular firmware development, automated software quality processes, and organization-specific product development strategies. Their experience spans consumer electronics, defense, automotive, robotics, cameras, drones, publishing, packaging, product design, industrial design, product lifecycle management (PLM), materials & assembly, supply chain, and manufacturing. Johnston can be reached by email at phillip@embeddedartistry.com.