ABSTRACT

Why would a commercial airliner with a long history of safety and reliability experience two recent fatal crashes within minutes of takeoff? Is it, in fact, a new aircraft, with fundamentally different handling characteristics that required fundamentally different operational software -- and pilot training? And given the aircraft and airline industries' inherent interest in safety, how could a new aircraft have been introduced into fleets worldwide without the requisite training? Differences in costs -- and philosophies -- between hardware and software provide an explanation.

Gregory Travis
812 606 1199
greg@littlebear.com
3/15/2019
V3.1

INTRODUCTION

I have been a pilot and aircraft owner for thirty years. I have been a software developer for over forty years. I have written extensively about both aviation and software engineering over those years. Now it's time for me to write about both, simultaneously and in the context of another subject near and dear to me: "normal failure."

The Boeing 737 MAX has been in the news because of two crashes, virtually back-to-back and involving brand new airplanes. In an industry that relies, more than anything, on the appearance of total control, total safety, these two crashes pose as close to an existential risk as you can get.

The 737 first appeared in 1967 (when I was three years old). Back then it was a smallish aircraft with smallish engines and relatively simple systems. Airlines (especially Southwest) loved it because of its simplicity, reliability, and flexibility. Not to mention the fact that it could be flown by a two-person crew – as opposed to the three or four of previous airliners – a significant cost saver.

EVOLUTION OF AN AIRLINER

Over the years, market and technological forces pushed the 737 into larger versions with more electronic and mechanical complexity. This is not, by any means, unique to the 737. All airliners, enormous capital investments both for the industries that make them as well as the customers who buy them, go through a similar growth process.

The majority of those market and technical forces allied on the side of economics, not safety. They were allied to relentlessly drive down what the industry calls "seat-mile costs" – the cost of flying a seat from one point to another.

BIGGER IS BETTER

Sulzer RT 96C Marine Diesel Engine (New Sulzer Diesel)

Much was concentrated on the engines themselves. The third law of thermodynamics and something called Carnot efficiency both dictate that the larger, and hotter, you can make any heat engine the more efficient it becomes. That's as true for jet airliner engines as it is for chainsaw engines.

It's as simple as that. The most effective way to make an engine more efficient, i.e. use less fuel per unit of power produced, is to make the engine physically larger. That's the reason why the Lycoming O-360 engine in my Cessna has pistons the size of dinner plates. That's the reason why marine diesel engines stand three stories tall. That's the reason why Boeing wanted to put the huge CFM LEAP engine in its latest version of the 737.

Only one little problem: the original 737 had (by today's standards) tiny little engines that easily cleared the ground beneath the wings. As the 737 grew and was fitted with bigger engines, the clearance between the engines and the ground started to get a little, umm, "tight."

737 "Classic" with original engines. Note size and ground clearance

Various "hacks" (as we would call them in the software industry) were developed. One of the most noticeable to the public was the "ovalization" of the engine intakes. Most 737s today have non-round engine intakes, the better to clear the ground (oh-my!).

RENTON, WE HAVE A PROBLEM

With the 737 MAX the situation became critical. The engines on the original 737 had a fan diameter (the intake blades on the engine) of just forty inches. The engines planned for the 737 MAX have a diameter of seventy inches. That's a centerline difference of well over a foot and there just wasn't enough "ovalization" that could be done to the intake to hang the new engines beneath the 737 wing, without the engines scraping the ground.

737 MAX with CFM LEAP engines. Note "ovalization" of intake AND placement of engine ahead and slightly above wing. This placement is what causes the "pitch up" when power is applied (Boeing).

The solution was to extend the engine up and well in front of the wing. However, doing so also meant that the centerline of the engine's thrust changed. Now, when the pilots applied power to the engine, the aircraft would have a significant propensity to "pitch up" – raise its nose.

Pitch-up in an aircraft increases something called the "angle of attack." That's the angle between the wings and the airflow over the wings. Think of sticking your hand out a car window on the highway – if your hand is level, you have a low angle of attack. If your hand is "pitched up," you have a high angle of attack. When the angle of attack is great enough, the wing enters what we call an aerodynamic stall. You can feel the same thing with your hand out the window – as you rotate your hand, your arm wants to move up like a wing more and more until you stall your hand, at which point your arm wants to flop down on the car door.

This propensity to "pitch up" with power application thereby increased the risk that the airplane could stall when the pilots "punched it" (as my son likes to say). Particularly if the airplane was also flying slowly.

To add insult to injury, not only would a power increase cause a pitch up but because the engine nacelles were so far in front of the wing and so large, they actually produce lift. And they produce that lift at high angles of attack. In other words, the nacelles make a bad problem (pitch up with power application) much worse.

I'll say it again: in the 737 MAX the engine nacelles themselves can, at high angles of attack, produce lift (like a wing). And the lift they produce is well ahead of the wing's center of lift, meaning the nacelles will cause the 737 MAX at a high angle of attack to go to a *higher* angle of attack. This is aerodynamic malpractice of the worst kind.

Pitch changes with power changes are common in aircraft. Even my little Cessna pitches up a bit when power is applied. Pilots train for and are used to it. Nevertheless, there are limits to what is allowed and still pass FAA certification. There are limits to what pilots will put up with.

Pitch changes with increasing angle of attack, however, are quite another thing. An airplane approaching an aerodynamic stall cannot, under any circumstances, have a tendency to go further into the stall. This is called "dynamic instability" and the only airplanes that exhibit that characteristic (fighter jets) are also fitted with ejection seats.

Everyone in the aviation community wants an airplane that flies as simply and as naturally as possible – that means that conditions should not change markedly, there should be no significant roll, no significant pitch change, no nothing whether the pilot is adding power, lowering the flaps, extending the landing gear, etc.

The airframe, the hardware, should "get it right" the first time and not need a lot of added on bells and whistles to fly predictably. This is aviation canon from the day the Wright brothers first flew at Kitty Hawk.

WHY MCAS? FOLLOW THE MONEY

Apparently the 737 MAX pitched up a bit too much for comfort on power application as well as at already-high-angles-of-attack. It violated that most ancient of aviation canons and probably violated the FAA's certification criteria. But, instead of going back to the drawing board and getting the airframe hardware right (more on that below), Boeing's solution was something called the "Maneuvering Characteristics Augmentation System," or MCAS.

Boeing's solution to their hardware problem was software.

I will leave a discussion of the corporatization of the aviation lexicon for another article but let's just say another term might be the "Cheap way to prevent a stall when the pilots punch it," or CWTPASWTPPI, system. Hmm, perhaps MCAS is better, after all.

MCAS is certainly much less expensive than extensively modifying the airframe to accommodate the larger engines. Such an airframe modification would have meant things like longer landing gear (which might not then fit in the fuselage when retracted), more wing dihedral (upward bend), etc. All of those hardware changes, compared to MCAS software, would be horribly expensive.

> "EVERTHING about the design and manufacture of the MAX was done to preserve the myth that 'it's just a 737.' Re-certifying it as a new aircraft would have taken years and millions of dollars. In fact, the pilot licensed to fly the 737 in 1967 is still licensed to fly all subsequent versions of the 737"
>
> Feedback from a 737 pilot for a major airline on an earlier draft of this article

What's worse: those changes could be extensive enough to require not only a re-certification of the 737 but to force an entirely new aircraft (i.e. a "Not 737"). Now we're talking *real* money, both for the manufacturer as well as the manufacturer's customers.

Because *the* major selling point of the 737 MAX is that it is just a 737 and any pilot who has flown other 737s can fly a 737 MAX without expensive training, without recertification, without another type rating. Airlines, like Southwest, want one "standard" airplane. They want to have one airplane that all their pilots can fly because that makes both pilots and airplanes fungible. That fungibility is the key to maximizing flexibility and minimizing costs.

It all comes down to money and, in this case, MCAS was the way to keep the money flowing, both for Boeing and its customers, in the right direction. The necessity to insist that the 737 MAX was no different in flying characteristics, no different in systems, from any other 737 was the key to the 737 MAX's fleet fungibility. That's also the reason why the documentation about the MCAS system was kept so far on the down-low.

Too much visibility, too much of a change to the aircraft's operating handbook, any training requirement whatsoever and someone – probably a pilot – would have piped up and said "hey. This doesn't look like a 737 any more." And then the money would flow the wrong way.

HOW IT WAS IMPLEMENTED

As mentioned earlier, there's something called "angle of attack." You can do your own angle of attack experiments just by putting your hand out the car door window and rotating it. It turns out that sophisticated aircraft have what is, essentially, the mechanical equivalent of a hand out the window: The angle of attack sensor.

You may have noticed this sensor when boarding a plane. There are usually two of them, one on either side of the plane, and usually just below the pilot's windows. Don't confuse them with the pitot tubes (we'll get to those, later). The angle of attack sensors look like wind vanes whereas the pitot tubes look like, well, tubes.

Angle of attack sensor

There's a reason the angle of attack sensors look like wind vanes and that reason is that's exactly what they are. They are mechanical hands designed to measure the angle of attack (or "relative wind") and they rotate, just like your hand out the window, in response to changes in that angle of attack.

Pitot tube

The pitot tubes measure how much the air is "pressing" against the airplane whereas the angle of attack sensors measure what direction that air is coming from. The pitot tubes, because they measure air pressure, are used to determine the aircraft's speed through the air. The angle of attack sensors measure the aircraft's direction relative to that air.

There are two sets of each. One on either side of the fuselage. Normal usage is to have the set on the pilot's side feed the instruments on the pilot's side and the ones on the co-pilot's side feed the instruments on the co-pilot's side. That gives a state of natural redundancy in instrumentation that can be easily cross-checked by either pilot. If the co-pilot thinks his airspeed indicator is acting up, he can look over to the pilot's airspeed indicator and see if it agrees. If not, both pilot and co-pilot engage in a bit of triage to determine which instrument is profane and which instrument is sacred.

Long ago there was a joke that in the future planes would fly themselves and the only thing in the cockpit would be a single pilot and a dog. The pilot's job was to make the passengers comfortable that someone was up front. The dog's job was to bite the pilot if he tried to touch anything.

On the 737, Boeing not only included the requisite redundancy in instrumentation and sensors, but it included redundant flight computers – one on the pilot's side and one on the co-pilot's side. The flight computers do a lot of things but the main thing they are there to do is a) act as the autopilot (i.e. fly the plane by computer) when commanded and b) make sure that the human pilots don't do anything wrong when the autopilot isn't flying the plane. The latter is called "envelope protection."

Let's just call it what it is: the bitey dog.

RAGE AGAINST THE MACHINE

Let's review what MCAS does: MCAS pushes the nose of the plane down when the MCAS system thinks the plane might exceed its angle of attack limits – in order to avoid an aerodynamic stall. Boeing put MCAS into the 737 MAX because the larger engines and, in particular the placement of the engines for ground clearance reasons, make an aerodynamic stall more likely in a 737 MAX than in previous 737 models.

In the 737 MAX MCAS is implemented in the flight computer software. When MCAS senses that the angle of attack is too high, it commands the aircraft's trim system (the system that makes the plane go up or down) to lower the nose. It also does something else: it pushes the pilot's control columns (the things the pilots pull or push on to raise or lower the aircraft's nose) in the down direction.

In the 737 MAX, like most modern airliners and most modern cars, everything is monitored by computer, if not directly controlled by computer. In many cases, there are no actual mechanical connection (cables, push tubes, hydraulic lines, etc.) between the pilot's controls and the things on the wings, rudder, etc. that actually make the plane move in different directions. And, even where there are, it's up to the computer to determine if the pilots are engaged in good decision making (that's the dog bite, again).

But it's also important that the pilots get physical feedback about what is going on. In the old days, when cables connected the pilot's controls to the flying surfaces, you had to pull up, hard, if the airplane was trimmed to descend. You had to push, hard, if the airplane was trimmed to ascend. With computer oversight there is a loss of natural sense in the controls as to what the airplane wants to do – how it is trimmed. In the 737 MAX there is no real "natural feel."

There is only artificial feel. There is only the feeling that the computer wants the pilots to feel.

And, sometimes, it doesn't feel so great.

POWER POLITICS

737 MAX Flight Deck, including pilot and co-pilot's control yokes and columns (Boeing).

When the flight computer trims the airplane to descend, because the MCAS system thinks it's about to stall, a set of motors and jacks push the pilot's control columns forward. It turns out that the flight management computer can put a LOT of force into that column. So much force,

in fact (hundreds of pounds), that a human pilot can quickly become exhausted trying to pull the column back, trying to tell the computer that this really, really, really should not be happening.

In fact, not letting the pilot regain control by pulling back on the column was an explicit design decision of MCAS. Because if the pilots could pull up the nose when MCAS said it should go down, why have MCAS at all?

HOW WE GOT HERE

MCAS needed because 737 MAX more likely to stall than earlier 737s. 737 MAX more likely to stall because Boeing hacked on huge engines in an attempt to win the seat-mile-cost competition. MCAS implemented in the flight management computer, even when the humans think they are flying it. In a fight between flight management computer and human pilots over who is actually in charge, flight management computer will bite humans until they give up and (literally) die.

Finally, need to keep the existence of the MCAS system on the extreme down-low lest someone say “Hey, this isn’t your father’s 737” and bank accounts start to suffer.

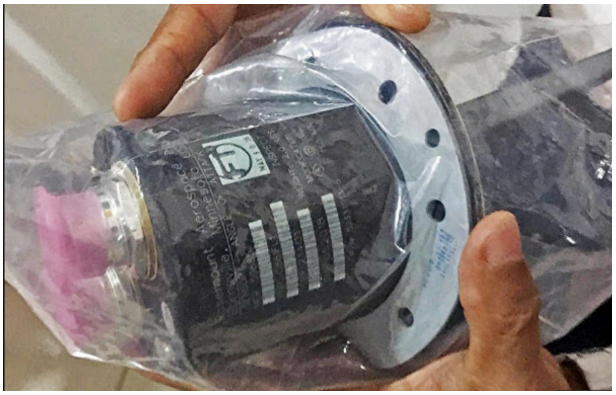
HERE’S WHERE IT GETS AS TRAGIC AS IT IS INTERESTING

The flight management computer is a computer. What that means is that it’s full not of aluminum bits, cables, fuel lines and all the other accoutrements of aviation. It’s full of lines of code. And that’s where things get dangerous.

Because those lines of code are created by people at the direction of those people’s managers. And all those people likely are not as in touch with the particular culture and mores of the aviation world as are the people who are down on the factory floor, riveting wings on, designing control yokes, and fitting landing gears. Those people have decades of institutional memory about what has worked in the past and what has not. What has left gigantic gaping holes in the ground. The software people do not.

In the 737 MAX only one of the flight management computers is active at once. Either the pilot’s flight management computer or the co-pilot’s flight management computer. And that computer takes inputs ONLY from the sensors on the side of the aircraft corresponding to which flight computer is in control.

Above I posited a (very common) situation in which one pilot's instruments were giving faulty readings. The solution for the humans is to look across the control panel to see what the other instruments are saying and then sort it out.



Angle of attack sensor from Lion Air crash. (TimothyMcLaughlin/Washington Post)

In the aviation lexicon this is called "cross check" and it's as natural to a pilot as putting on goggles and a leather jacket.

In the Boeing system, the flight management computer does not "look across" at the other instruments. It only believes the instruments on "its" side. In the Boeing system, the flight computer has no need for a leather jacket and goggles. It doesn't go old school. It's modern. It's software.

This means is that if a particular angle of attack sensor goes haywire (which happens all the frigging time in a machine that alternates from one extreme environment to another, vibrating and shaking all the way) the flight management computer just believes it. It doesn't check the angle of attack sensor on the other side to see if they agree.

It gets even worse. There are several other instruments that can be used to determine things like angle of attack, either directly or indirectly, such as the pitot tubes, the artificial horizons, etc. All of these things would be used by a human pilot ("cross check") to quickly diagnose a faulty angle of attack sensor.

In a pinch, a human pilot could just look out the windshield to confirm visually and directly that, no, the aircraft is not pitched up dangerously. That's the ultimate check and should go directly to the pilot's ultimate sovereignty. Unfortunately, the current implementation of MCAS denies that sovereignty. It denies the pilots what is clear before their eyes.

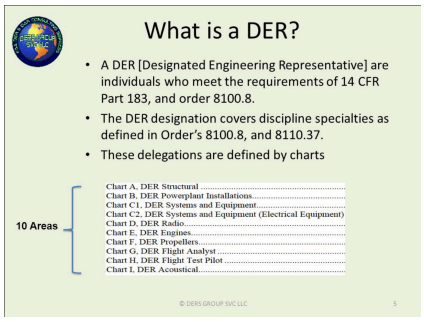
Like someone with narcissistic personality disorder, MCAS gaslights the pilots. And it turns out badly for everyone. "Raise the nose, HAL." "I'm sorry, Dave, I can't do that."

In the MCAS system, if a single angle of attack sensor on the side of the plane corresponding to the "in charge" flight management computer goes haywire, the flight management computer will believe it and is blind to any other evidence that it is wrong, including the pilot's own eyes.

Including the evidence in the form of the arms of the pilots desperately trying to pull back on robotic control columns that are biting them, and their passengers, to death.

HOW COULD THIS HAPPEN?

In the old days, the Federal Aviation Administration (FAA), had armies of aviation engineers in its employ. Those FAA employees worked side-by-side with the airplane manufacturers to determine that an airplane was safe and could be certified as airworthy.



What is a DER? (DERS GROUP SVC, LLC)

As airplanes became more complex and the gulf between what the FAA could pay and what an aircraft manufacturer could pay grew larger, more and more of those engineers migrated from the public to the private sector. And soon the FAA had no in-house ability to determine if a particular airplane's design and manufacture was "safe." So the FAA said to the airplane manufacturers, "why don't you just have your people tell us if your designs are safe?"

The airplane manufactures said "Sounds good to us." The FAA said "And say hi to Joe, we miss him."

Thus was born the concept of the "Designated Engineering Representative," or DER. DERs are individuals in the employ of the airplane manufacturers, the engine manufacturers, and the software developers who certify to the FAA – as if they themselves worked for the FAA and the public trust – that it's all good.

Now this is not quite as much of a sinister conflict of interest as it sounds. As I mentioned at the beginning of this, it is in nobody's interest in the aviation industry that airplanes crash. The industry absolutely relies on the public trust and every crash is an existential threat to the industry. No manufacturer is going to employ DERs that just pencil whip the paperwork. On the other hand, though, after a long day and after the assurance of some software folks, they might just take their word that things will be OK.

That no one who wrote the MCAS software for the 737 MAX seems to have even raised the issue of using multiple inputs, including the opposite angle of attack sensor, in the computer's determination of an impending stall is mind-blowing. As a lifetime member of the software development fraternity, I don't know what toxic combination of inexperience, hubris, or lack of cultural understanding led to this.

> "But investigators have speculated that incorrect data — **including a 20-degree differential between two sensors** designed to measure, essentially, the difference between the pitch of the plane and direction it is moving through the air — could have mistakenly triggered both the stick shaker and the anti-stall system, which is called MCAS."
>
> *New York Times, 3/20/2019*

But I do know that it's indicative of a much deeper and much more troubling problem. The people who wrote the code for the original MCAS system were obviously terribly far out of their league and did not know it. How can we possibly think they can implement a software fix, much less give us any comfort whatsoever that the rest of the flight management software, which is ultimately in ultimate control of the aircraft, has any fidelity at all?

If I have not been clear, so far, let me say it succinctly. Boeing produced a dynamically unstable airframe, the 737 MAX. That is big strike #1. Boeing then tried to mask the 737's dynamic instability with a software system, similar to the systems used in dynamically unstable fighter jets (though those jets are fitted with ejection seats). Big strike #2. Finally, the software system relied on systems known for their propensity to fail (angle of attack indicators) and did not appear to include even rudimentary provisions to cross check the outputs of the angle of attack sensor against other sensors, including the other angle of attack sensor. Big strike #3.

None of the above should have passed any muster. None of the above should have passed the "ok" pencil of the most junior engineering staff, much less a DER.

That's not a big strike. That's a political, social, economic and technical sin on more levels than Dante' could ever conceive.

THE ECONOMICS OF AVIATION

It just so happens that, during the timeframe between the first 737 MAX crash and the most recent 737 crash, I'd had the occasion to upgrade and install a brand-new digital autopilot in my own aircraft. That aircraft, a 1979 Cessna 172, is the most common aircraft in history, at least by production numbers. Its original certification also predates that of the 737's by about a decade (1955 vs. 1967).

My Cessna 172's control panel. Note location of autopilot circuit breaker. Also note autopilot disconnect switch (red button atop left-hand side of pilot's yoke). Control cables physically link the control yokes to the flying surfaces in my plane.

My new autopilot consists of several very modern components, including redundant flight computers (dual Garmin G5s) and a sophisticated communication "bus" (CANBUS) that lets all the various components talk to each other, irrespective of where they are located in my plane. CANBUS derives from automotive "drive by wire" technology but is otherwise very similar in

purpose and form to the various ARINC buses that connect together the components in the 737 MAX.

My autopilot also includes electric pitch trim. Meaning it can make the same types of configuration changes to my 172 that the flight computers and MCAS system in the 737 MAX can make to it. During the installation, after the first 737 MAX crash, I remember remarking to a friend that it was not lost on me that I was potentially adding a hazard similar to the one that brought down the Lion Air crash (see "normal failure," below).

Finally, my new autopilot also implements "envelope protection." If my Cessna is NOT being flown by the autopilot, the system nonetheless constantly monitors the airplane to make sure that I am not about to stall it, roll it inverted, or a whole host of other things. Yes, it has its own "bitey dog" mode.

As you can see, the similarities between my $20K autopilot and the multi-million dollar autopilot in every 737 are direct, tangible, and relevant. What, then, are the differences?

For starters, the installation of my autopilot required paperwork in the form of what's called a "Supplemental Type Certificate," or STC. In other words, the autopilot manufacturer and the FAA both agreed that my 1979 Cessna 172 with their (Garmin's) autopilot was so significantly different from what it was when it rolled off the assembly line that it was *no longer the same Cessna 172.* It was a different aircraft, altogether.

In addition to now carrying a new (supplemental) aircraft type certificate (and certification), my 172 required a very large amount of new paperwork to be carried in the plane, in the form of revisions to and addendums to the aircraft operating manual. As you can guess, most of those addendums revolved around the autopilot system.

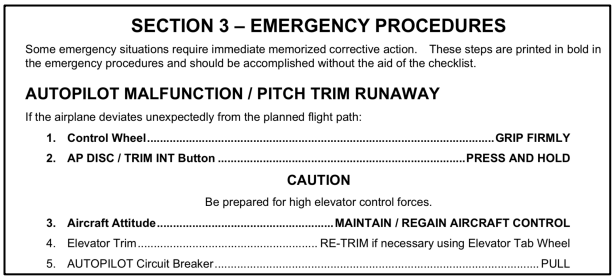
GARMIN Ltd. or its subsidiaries
c/o GARMIN International, Inc.
1200 E. 151st Street
Olathe, Kansas 66062 U.S.A.

FAA APPROVED AIRPLANE FLIGHT MANUAL SUPPLEMENT

GFC 500 Autopilot with ESP

Installed in

Textron Aviation 172E / 172F / 172G / 172H / 172I / 172K / 172L / 172M / 172N / 172P / 172Q / 172R / 172S



SECTION 3 – EMERGENCY PROCEDURES

Some emergency situations require immediate memorized corrective action. These steps are printed in bold in the emergency procedures and should be accomplished without the aid of the checklist.

AUTOPILOT MALFUNCTION / PITCH TRIM RUNAWAY

If the airplane deviates unexpectedly from the planned flight path:

1. **Control Wheel** ..... **GRIP FIRMLY**
2. **AP DISC / TRIM INT Button** ..... **PRESS AND HOLD**

CAUTION

Be prepared for high elevator control forces.

3. **Aircraft Attitude** ..... **MAINTAIN / REGAIN AIRCRAFT CONTROL**
4. Elevator Trim ..... RE-TRIM if necessary using Elevator Tab Wheel
5. AUTOPILOT Circuit Breaker ..... PULL

Extracts of the addendums to my Cessna's operating manual as a result of the autopilot upgrade. This documentation is carried in the plane at all times (Garmin International).

Of particular note in that documentation, which must be carried in the plane at all times and must be studied and understood by whomever pilots it, are various explanations of the autopilot system, including its command of the trim control system and its envelope protections.

There are instructions on how to detect when the system malfunctions, through terms such as "pitch trim runaway," *and how to disable the system, immediately.* Disabling the system means pulling the autopilot circuit breaker and instructions on how to do that are strewn throughout the documentation, repeatedly. Every pilot who flies my plane becomes intimately aware that it is NOT the same as any other 172.

This is a big difference between what pilots of my plane are told and what pilots stepping into a 737 MAX are (or were) told.

Another difference between my autopilot system and that in the 737 MAX. All of the CANBUS-interconnected components constantly do the kind of instrument "crosscheck" that human pilots do and that, apparently, the MCAS system in the 737 MAX does not. For example, the autopilot itself has a self-contained attitude platform that checks the attitude information coming from the G5 flight computers. If there is a disagreement, the system simply goes off line and alerts the pilot that she is now flying manually. It doesn't point the airplane's nose at the ground, thinking it's about to stall.

Perhaps the biggest difference between the system in my 172 and the 737 MAX is the amount of physical force it takes for the pilot to override the computers. In my 172 there are still cables linking the controls to the flying surfaces. There is no computer to mediate and what computer is there has to press on the same things that I have to press on. And its strength is nowhere near as much as mine. So, even if there were to be an error – the computer in my plane thought it was about to stall when it wasn't -- I can easily overcome the computer.

In my Cessna, unlike the 737 MAX, the humans still win a battle of the wills, every time. That used to be a design philosophy of every Boeing aircraft, as well, and one they used against their arch-rival Airbus (who had a different philosophy). But it seems that, with the 737 MAX, Boeing has changed philosophies about human/machine interaction as quietly as they've changed their aircraft operating manuals.

TWO SETS OF RULES

I've brought my Cessna into the discussion because I thought it important to illustrate the way the process is designed to work, namely that momentous changes to an aircraft's flight characteristics, behavior, and equipment come with momentous changes to its documentation and certification as well as to pilot inculcation. In my Cessna, the documentation itself is several pounds, all of which must now be carried around permanently -- in an aircraft in which every pound counts.

The degree to which society, "the system," is rigged is a platform component in the upcoming US elections. The public perception is that a tiny majority of the population holds and controls the vast majority of the population's wealth. And not a day goes by without a story of corporate or individual malfeasance on a Titanic scale, yet with Lilliputian consequences. That this is real (it is) and that the public has become inured to it represent another kind of

existential threat: an existential threat to the United States as a moral, technical and economic leader in the World.

I would never pretend that the differences in process between what Boeing was required to do with the 737 MAX (nothing) and what I was required to do with my Cessna (significant) is emblematic of class struggle. As a white male conceived lucky, I started life in the ninety-eighth percentile (maybe ninety sixth, but you get the point) and I have not had to look in the rear-view mirror since I was born.

The point is that the 737 MAX saga teaches us not only about the limits of technology and the risks of complexity, it teaches us about our real priorities. Safety is never first, no matter what the market campaigns would like us to believe. Money is first and safety's only utility in that regard is in helping keep the money coming.

SOFTWARE VS. HARDWARE

Hardware defects, whether they are engines placed in the wrong place on a plane or O-rings that turn brittle when cold, are notoriously hard to fix. And by hard, I mean expensive. Software defects, on the other hand, are easy and cheap to fix. All that you need to do is post an update, push a patch, etc. What's more, we've enculturated consumers to consider this normal with everything from updates to desktop operating systems, like Windows, to the monthly software patches that get posted automatically to my Tesla, while I sleep.

Back in the 1990s, there was something called the "FDIV" bug that affected early Intel Pentium processors. I wrote an article back then comparing the relative complexity of the Pentium Processors of that era, expressed as the number of transistors on the chip, compared to the complexity of the Windows operating system of the time, expressed (as I recall) as number of lines of code (LOC). What I found was that the complexity of both the Pentium processors of the time and the contemporaneous Windows operating system was roughly equal, certainly within an order of magnitude of one-another.

Now the FDIV bug was relatively obscure. It affected only a tiny fraction of Pentium users. Windows also was affected by similar defects, also affecting only fractions of its users.

But the effects on the companies were quite different. Where Windows addressed its small defects with periodic software updates (the process was much more cumbersome back then), Intel felt obligated to recall the (slightly) defective processors, at a cost of $500 million (nearly a billion dollars, today).

WE'VE MET THE ENEMY AND HE IS US

I believe the relative ease, not to mention the lack of tangible cost, of software updates has created a cultural laziness within the software engineering community. Moreover, because

more and more of the hardware that we create is monitored by and controlled by software, that cultural laziness is now creeping into "hard engineering."

Like building jet airliners.

By laziness, I mean that less and less thought is being given to getting a design correct, and simple, up-front. Because it's so easy to fix what you didn't get right the first time, later.

> "I'm a software developer turned network engineer and have written airliner avionics software in the past. It was interesting how many hoops we had to jump through to get an add-on board for the computer certified, while software certifications were nil (other than "cannot run on Windows", "must be written in C++"). This was, admittedly, nearly 10 years ago, and I hope that things have changed since."
>
> *Anonymous*, personal correspondence

When I was running a software development team, I used to lean out my door and yell "ship it!" I was only half tongue-in-cheek because the dirty little secret in any software firm is that the customer is the cheapest, and most effective, quality control department your company can have. Let the customers find the problems and then we can come up with a patch and send it out for free. It's easy to offer a warranty when the cost of warrantying is zilch.

And, because of the prevalence of software control, that same software laziness is creeping into hardware building. As I am constantly reminded, by the software updates pushed to my Tesla, the updates pushed to the Garmin flight computers in my Cessna, even by the updates to my Nest thermostat and the TVs in my house – none of those "things" were complete when they left the factory. Because their builders realized they didn't have to be complete, the job could be done at any time in the future – with a software update.

Walt Kellly

What needs to happen, I think, is for liability to accrue where it is generated. For too long the costs of software imperfection have been, as economists say, "externalized." They've been externalized to the aforementioned customers and now they're being brutally externalized to the traveling public, whether in the form of an autonomous car wreck or an airliner crash. It was perhaps OK when a Windows defect meant your copy of Tetris didn't load. It's not OK when a firmware defect means your car runs into a guardrail.

And kills you.

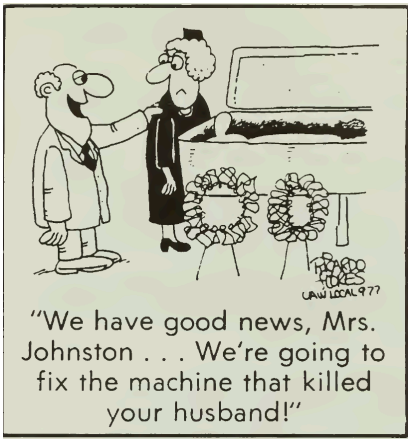
WHAT HAPPENS NEXT, MAX?

Boeing is in the process of rolling out a set of software updates to the 737 MAX flight control system, including MCAS. I do not know, specifically, but I suspect that the center of those updates will be two things:

1. To have the software "crosscheck" indications, just like a human pilot would. Meaning, if one angle of attack indicator says the plane's about to stall, but the other one says it's not so, at least hold off judgement about pushing the nose down into the dirt and maybe let a pilot or two know you're getting conflicting signals.
2. Back off on the "shoot first, ask questions later" design philosophy. Meaning, look at multiple inputs (see above) and if the angle of attack indicators say you're about to stall but the pitot indicators say you're flying too fast for that to be possible, just trip offline. Don't push the nose down into the dirt. Let a pilot or two know you're getting conflicting signals. Because nothing's perfect, certainly not hardware, and shit happens.

For the life of me, I do not know why those two basic aviation design considerations, bedrocks of the "simplify, then add lightness" mindset that has served the industry so well until now, were not part of the original MCAS design. And, when they were not, I do not know or understand what part of the DER process failed to catch the fundamental design defect.

> "The FAA last week said it planned to mandate changes in the system to make it less likely to activate when there is no emergency. The agency and Boeing said they are also going to require additional training and references to it in flight manuals."
> *Time* 3/19/2019



Processed World, vol 17.

But I suspect that it all has to do with the same thing that brought us from Boeing's initial desire to put larger engines on the 737 and to not have to internalize the cost of those larger engines. In other words, to do what every child is taught is impossible: get a free lunch.

Rhetorically I can question why adults forget what children know. And that's because we don't want to believe that there isn't a free lunch. We want to be able to conjure magic, to perform some kind of adult fiscal alchemy, and prove the suckers wrong. We want to prove that you can have your cake and eat it, too. That you can take it with you when you go.

EPILOG: Normal failure and "it's the software, dummy!"

In short, the concept of normal failure is this: As systems become more complex, failures of those systems become more "normal." Nowhere is this more acutely felt than in systems designed to augment or improve safety. From the point of safety and reliability, every increment, every increase in complexity, ultimately leads to decreasing rates of return and, finally, negative returns.

> "In interviews late last week, aviation experts said there was no reason for broad alarm about the sensors. But six experts said that the risks posed by a faulty angle-of-attack sensor are amplified by the increasing role of cockpit automation. It is an example of how the same technology that makes aircraft safer — automated software — can be undone by a seemingly small problem." *Washington Post 3/18/2019*

This is the root of the old engineering axiom: Keep It Simple, Stupid (KISS) and its aviation-specific counterpart: "Simplify, then add lightness."

An understanding of this is being lost. I do not know of a single aviation accident involving the 737 that was the result of an inadvertent stall by its pilots. I can only speculate, but I speculate that – even without MCAS – that would have continued to be true of the 737 MAX.

The original FAA Eisenhower-era certification requirement was a testament to simplicity. Namely, that planes should not exhibit significant pitch changes with changes in engine power. That requirement was written long before the advent of computers, in the days when there was a direct connection between the controls in the pilot's hands and the flying surfaces on the airplane. Because of that, when written, the requirement rightly imposed a discipline of simplicity on the design of the airframe itself.

The airframe, not the systems that support the airframe, had to inherently be immune to pitch changes with changes in power.

Between when that requirement was written and today, software came onto the scene. And now software stands between man and machine. And no one seems to know what, exactly, is going on. Because things have become complex beyond the ability of an organization to process.

The DER process worked well when airplanes were made of metal and not code. But when the promise of a complex software solution to a vexing business problem (make a new 737 without having to admit it was new) appeared, the various organizations were unable to ascertain the risk, they only wanted the reward. And that was the reward of a free lunch: increased complexity that would guarantee increased safety.

It doesn't work that way. And we must consider the damage we do by the promises we make.

I cannot get the parallels between the 737 MAX accidents and the Space Shuttle Challenger accident out of my head. The Challenger accident, another textbook case-study in normal failure, came about not because people didn't follow the rules. It came about because people

followed the rules. In the Challenger case, the rules said that they had to have pre-launch conferences to ascertain flight readiness. It didn't say that a significant input to those conferences couldn't be the political considerations of delaying a launch. The inputs were weighed, the process was followed, and a majority consensus was to launch.

And seven people died.

In the 737 MAX case the rules were also followed, likely to a "T." The rules said you couldn't have a large pitch-up on power change. The rules said that an employee of the manufacturer, a DER, could sign off on whatever you came up with to prevent a pitch change on power change. The rules didn't say that the DER couldn't take the business considerations, much less their own career, into their decision-making process.

And three-hundred and forty-seven people are dead.

It is likely that MCAS, originally added in the spirit of increasing safety, has now killed more people than it could have ever saved. It doesn't need to be "fixed" with more complexity, more software. It needs to be removed, altogether.