



Harmful Specifications & Midterm Review

Zachary G. Ives & Harry Smith
University of Pennsylvania
Fall 2026

Today's Big Question

What kinds of lessons can we learn from a high-profile failure?

- Last time: how to figure out whether the thing you say you want is really what you actually want.
- This time: what can happen if want the wrong thing.

The Boeing 737 MAX

Be on the lookout for...

- Specifications changing late in the development lifecycle
- Specifications designed without input & insight from all interested parties
- Documentation that doesn't represent the state of the project
- No guardrails enforcing that the *built* thing matches the *specified* goal.

...as we discuss what got us here:

What We Know About the Lion Air Flight 610 Crash

The new plane's automated systems may have played a role, but the precise cause remains unknown.

 Share full article



Ethiopian Airlines Plane Is the 2nd Boeing Max 8 to Crash in Months

 Share full article



 372



At least 150 people were killed when Ethiopian Airlines Flight 302 crashed early ⁴Sunday. It was the second such disaster involving a brand-new Boeing 737 Max 8 in months. Michael Tewelde/Agence France-Presse — Getty Images

Citing Sources

Most of what follows comes from an interview: **Peter Robison**, a Bloomberg reporter who covered Boeing for two decades, talking to Dave Davies about his book *Flying Blind*.

Source: [NPR *Fresh Air*, November 29, 2021](#) — full transcript. Every screenshot in this section is from it.

How Did We Get Here?

By around 2010 Boeing was under competitive pressure from Airbus and wanted an upgraded 737 quickly.

One early strategic decision to make this possible: the new aircraft **must not require simulator training** for pilots who already flew the 737.



DAVIES: Right. And just to clarify that, I mean, to bring people to a simulator, I mean, there aren't a lot of these. You got to fly him there. You've got to pay for their – to be there for three or four days. And they're very expensive to run. It gives you the actual feeling of being in – running an aircraft. But it's expensive, and you're talking about thousands of pilots, right? If every pilot that runs this thing, that's going to be a cost in – what? – hundreds of millions?

ROBISON: Right. Well, each simulator is an engineering marvel in itself. It can cost \$10 million itself. And to run through the training, you need the pilot

MCAS: Software Standing In for Hardware

Important constraints:

- MAX's larger engines had to sit further forward and higher on the wing
- Changing the airframe was expensive.
- Retraining pilots was forbidden.

So: software. **MCAS** (Maneuvering Characteristics Augmentation System) pushed the nose down automatically, so the aircraft would feel like the 737 those pilots already knew.

A requirement that could not be met by engineering or by training was met instead by software whose entire job was to make the system feel like something it wasn't.

Building an Airplane is Hard

DAVIES: You know, building an aircraft is this huge project that involves all kinds of design issues and can take a long, long time. And if you discover a flaw, something that needs correcting, it can be expensive to correct it. What was the old Boeing like when it came to tweaking and fixing things that might be a problem?

ROBISON: Right. Airplanes are incredibly complex products. They involve hundreds of thousands of parts. And in the flight testing program, there are often surprises. There was always a saying that managers should plan for **20,000 surprises** in any airplane program.

Finding problems late in a large program is the norm, actually!

Formerly: Interaction among Experts Identified & Fixed Problems

often surprises. There was always a saying that managers should plan for 20,000 surprises in any airplane program.

And there are several examples I cite in the book where Boeing did discover problems late in flight tests. There was one with, really, the original plane of the jet age, the 707, where an instructor at Boeing crashed the plane into a riverbank, and the investigation showed that it was his mistake in part, that he'd exceeded the maximum bank angle. But the chief test pilot at Boeing at the time, a fellow named Tex Johnston, decided that the design could also be improved, that there were changes that could be made to the tail to prevent the pilot from needing to take such drastic action, from needing to have superhuman skills. So the pilot went to the engineers, who said, we will fix it, and Boeing fixed it.

(This plane, the 707, was designed in the 50s.)

10

DAVIES: So when did the corporate culture change, and what drove it?

What Changed? Boeing Did!

ROBISON: The culture really started changing in the late 1990s, and that's a period when I was also a beat reporter for Bloomberg News covering Boeing, so I saw it firsthand. And the CEO at the time, Phil Condit, was very drawn to the bold vision of capitalism that he saw from especially Jack Welch at General Electric. Jack Welch was pioneering another way to run a manufacturing company, and his way was about services and financial engineering and mergers and acquisitions. And so from that point, the commercial airplane side of the business was considered just one piece of the business, that Boeing would find its future following the GE model. And that's when Boeing moved its headquarters from Seattle to Chicago, and that really symbolizes the shift from manufacturing to financial engineering.

11

Before: "Working Together"

DAVIES: Right. There was a commitment to cutting costs, and one way that that was done was by outsourcing - right? - and moving things, like they - moving the training simulators from Seattle to Florida, I think. I mean, can you help us understand ways in which this undermined quality and safety?

ROBISON: Sure. So - but historically, Boeing's planes had succeeded because of communication. They - probably the last great airplane Boeing built was the 777. And for that airplane, Boeing created the slogan, working together. And it got these giant teams together every month to talk about their piece of the airplane because every small piece of the airplane contributes to the whole. And that had been Boeing's historic style.

After: Not So Much

The problem in the minds of people who wanted to pursue the standard corporate playbook was that it's more expensive. It's more expensive to have cross-functional teams. It's expensive to communicate and to have people traveling. And also, Boeing's unions had a lot of clout. And being based in

...

headquarters, so if an engineer had a question of a pilot, they could walk over to the simulators and run through the procedure and check it out. Boeing moved the simulators to Miami and started hiring more non-union pilots and more contractors, which eroded the communication.

Formal Reviews as Backstop

Should be OK to cut some of the "incidental reviews" as long as the final checks are all still solid!

Right...?

ROBISON: There was. There was a meeting in 2015, where a Boeing team, including Mark Forkner, who was a pilot who was working on the training requirements for the MAX, likened the FAA officials to, quote, "dogs watching TV." To the mind of people at Boeing, the FAA was just another box to be checked, that these are people who don't understand the airplane in the way that they do. And as this pilot Mark Forkner said, even he didn't understand it. And also before that meeting, someone said, with all the inexperience present, we should be able to gang up on them and steer it the direction we want. There was a sense that the FAA had been dumbed down over time, that

Oh. Ah. Okay. Nevermind!

The Outcome

1. Although two sensors purportedly existed, only one contributed to the firing of the MCAS system.
2. The single sensor experienced a fault, incorrectly reporting the aircraft's angle of attack. ("We're climbing too fast!")
3. MCAS fires repeatedly—once every five seconds—to pitch the nose of the plane down.
4. Pilots don't know what's happening and the plane continues to wobble and point down.
5. Two plane crashes kill 346 people.



Failure Mode 1 – The Spec Expanded

As originally designed, MCAS was only capable of moving the horizontal stabilizer a maximum of **0.6 degrees**.

However, in March 2016 [...] when Boeing test pilots found the MAX was not handling well when nearing stalls at lower speeds, Boeing redesigned MCAS to enable it to activate at lower speeds. Moreover, the new version of MCAS was capable of moving the horizontal stabilizer a maximum of **2.5 degrees** [...] **This was a significant technical change with dramatic operational implications.**

[House Committee Final Report on the Boeing 737 MAX](#), September 2020, p. 103.

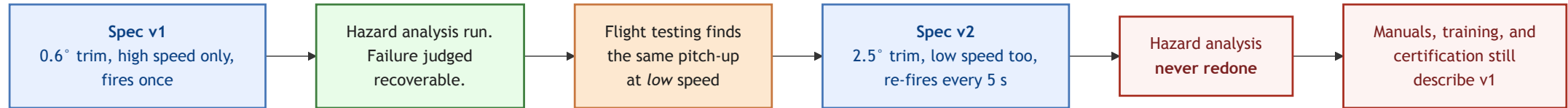
...and the Analysis/Documentation Didn't

The committee quotes the Joint Authorities Technical Review on what happened to the paperwork when MCAS changed:

The JATR team found that the **certification plans and some certification deliverables** (e.g., the preliminary system safety assessment (PSSA)) **were not updated to describe the expansion of the MCAS function** for the low Mach portion of the flight envelope and for compliance with stall-related requirements.

Same report, p. 105. The PSSA is the document that argues the system is safe — the artifact that should have been re-derived when the number changed.

Avoid Spec Drift



Remember: **specs are CI artifacts**. We said a stale spec should fail the build. Here the artifact that went stale was the *safety case*, and no process treated its staleness as a failure.

Failure Mode 2 — No Documentation in the First Place!

Pilots were not told MCAS existed. After the first crash, Boeing's CEO said on television that the relevant procedure was in the manual.

ROBISON: *"He was asked directly by Maria Bartiromo if MCAS was in the manual. And he said, yeah, it's in the manual. It's an existing flight procedure. In fact, it was only in the glossary."*

Remember: **documentation as part of the system.** "It's in the glossary" doesn't really cut it.

Failure Mode 3 — One Sensor

The redundancy argument made at design time was that MCAS would read two independent inputs, but only one made it in!

ROBISON: *"The thinking at the time was that that's permissible because this software is tied to two different sensors... And that's precisely what happened in both crashes. You had one sensor fail, left the pilots in this tough spot."*

Remember: **desirable properties should be designed in!** For example, you should guarantee verification of inputs if those inputs have enough authority to move the plane!

Failure Mode 4 – The Override Was an Assumption

The safety case assumed the crew would recognize the symptom and respond within seconds. That assumption was never checked against crews who had not been told the system existed.

ROBISON: *"If they had been provided training and told the software existed, they could have responded adequately. But they weren't provided training because... the management at Boeing had insisted that pilots not be given simulator training on the MAX because it would be too expensive."*

"The human will catch it" is a requirement levied on a human being. It belongs in the spec, and it has to be tested against the person who will actually be there, in the state they will actually be in.

Fortunately, We Know How to Avoid These Mistakes!

Every failure mode you just saw is something this course already treats as a practice — three of them before today.

WHAT WENT WRONG	WHERE WE FIGURED OUT HOW TO FIX IT
Spec expanded, safety case not re-run	a stale spec fails the build
System removed from operator documentation	non-slop documentation as part of the system
Single unvalidated input, high authority	holistic properties designed in
Asking a question made expensive	rapid prototyping; precise specification with verification
Reviewer treated as a box to check	(later) PR review gates

What You Do Instead

Before you accept a change that expands what a system is allowed to do, ask three questions that no test suite will ask for you:

- **What else believed the old bound?** Documentation, training material, downstream consumers, another team's assumptions, an ADR.
- **What happens if the input driving this is wrong?** "Wrong" can be a mistake, or it can be malicious, too.
- **Who is expected to catch it, and have we ever checked that they can?**

None of these are code review questions. They are the reason the engineer is still accountable for a system the agent wrote.

Accountability: The Question with No Settled Answer

Who is responsible?

- The engineer who reviewed the diff and merged it?
- The organization that shipped it and set the schedule?
- The vendor whose model generated it?
- Nobody in particular — which is the current default?

Civil engineers stamp drawings and carry personal liability. Physicians hold licenses that can be revoked. Software engineering has neither...

We Already Have One Data Point

Boeing entered a deferred prosecution agreement in January 2021 totaling **\$2.51 billion** — a \$243.6 million criminal penalty, \$1.77 billion to airline customers, and a \$500 million fund for the families of the 346 dead.

- Robison points out that the airline & family penalties weren't really consequences of the Justice Department settlement
- The \$243.6mil penalty is what the simulations would have cost anyway.

One individual was criminally charged: Mark Forkner — the pilot from the "dogs watching TV" meeting. A mid-level employee, not a decision-maker. He was acquitted in 2022.

Overall, basically no targeted accountability!

Source: [Boeing Charged with 737 Max Fraud Conspiracy and Agrees to Pay over \\$2.5 Billion](#) — U.S. Department of Justice, January 7, 2021.

The Profession's Own Answer

The **ACM Code of Ethics** is the closest thing software has to a professional standard. Three clauses bear directly on this course:

- **1.2 — Avoid harm.** Including harm you did not intend and harm that is diffuse.
- **2.5 — Give comprehensive and thorough evaluations of computer systems and their impacts,** including analysis of possible risks.
- **3.1 — Ensure that the public good is the central concern** of professional computing work.

Unfortunately, this part is entirely up to you.

Does adding an AI code generator to that chain make accountability clearer or more diffuse — and who benefits from the answer? Should engineers who deploy generated code into safety-critical systems be personally liable, licensed, or neither? What would you personally need to be true before you would sign your name to a system you directed but did not write?

Midterm Prep

Midterm Details

Tuesday, October 13

- Here, in this room
- 90 minutes
- Use a cheatsheet if you like—**1 page, handwritten**
 - (that's single sided!)
- Two major sections:
 - Explanation of important ideas
 - Responding to case studies

What It Covers

Everything through today — Weeks 1 through 7, which is the first four modules of the course.

- **Modern software development.** What changed, the engineer's new role, and how an LLM is built and trained.
- **Managing agentic tools.** Light JS/TS reading fluency, agent failure modes, and context engineering.
- **Building good software with AI.** Architecture, decomposability, databases and ORMs, code review, documentation, technical debt.
- **Identifying and validating needs.** Turning requirements into useful specifications, testing, prototyping.

Frameworks You Should Be Able to Name and Use (1/2)

Several ideas in the course come as named, enumerated frameworks. Being able to recall the pieces *and* apply them to a situation you haven't seen is the level to aim for. This is not exhaustive!

- The ReAct (Reason -> Plan -> Act -> Observe) Loop
- LLM training, from pre-training to supervised fine-tuning and then RLHF
- What context mistakes look like (hallucination, recursive errors, "disobedience")
- The five context failure modes (why context mistakes happen)
 - Poisoning, Distraction, Confusion, Clash, Injection
 - How to fix them
- What makes agents dangerous and how to fix it (the lethal trifecta)
- The Approval Ladder

Frameworks You Should Be Able to Name and Use (2/2)

- Using `git` as an "acceptance gate"
- Using AI to review & the AI Review Quadrants
- Deep modules, and why agents benefit
- Safe refactoring patterns
- What different kinds of testing are good for
- What pieces make up a full specification
- RFC 2119 keywords and the spec formality spectrum
- The virtues of iterative prototyping

Questions Worth Being Able to Answer #1

If you can answer these without notes, you are in good shape.

Choose one refactoring pattern. Explain briefly how it works and what kinds of issues it helps to avoid.

Model Answer #1

Choose one refactoring pattern. Explain briefly how it works and what kinds of issues it helps to avoid.

Strangler Fig. Put a facade in place that routes each request to either the old or the new implementation being built. Shift traffic to the new one gradually until nothing routes to the old code. The system keeps working the whole time, so there is never an outage and you can roll back by changing the config.

Questions Worth Being Able to Answer #2

If you can answer these without notes, you are in good shape.

Describe the two extreme ends of the Approval Ladder for accepting agent actions and how "auto-mode" can be understood as a useful compromise between these ends. Finally, explain how "auto-mode" might be understood as risky or optimistic.

Model Answer #2

Why can "auto-mode" be understood as a useful improvement over either "manual mode" or "accept edits"? Explain how "auto-mode" might be understood as risky or optimistic.

On either end, you have an "accept all" and "review all" (plan mode) for agent choices. Auto-mode uses a classifier to make reasonable choices about these actions, making it work like "accept all good & safe". Since it is a classifier, it is a general-purpose model that assumes acceptable actions are recognizable from the commands rather than the context, e.g. `git diff` vs. `git diff > leaks.txt`

Case Study Structure

You review a session started by a coworker looking to modify the permission system. The coworker started by having the agent review the Developer Documentation, which reported a method that doesn't exist. Over the rest of the session, the agent fixates on building that method and continues to try to use it even after being pointed to the other method that implements that functionality.

1. Name the failure mode behind each symptom and give the evidence. Say which symptoms share a single root cause, and when that cause entered the context. (6 points)
2. Give a concrete recovery procedure. Say what you keep and what you do differently.
3. Identify the ways in which the failure mode that emerged was due to Agent error. Also identify the ways that emerged due to Human error. (5 points)
4. Name two things you would put in place before the next session so this is less likely, or at least shows up sooner. (5 points)

How to Study

Re-read your own work first. Labs 1 through 6 are the course in the form you already made your own — your CLAUDE.md, your architecture diagram, your Developer Guide, your spec and its CI check.

Sources for the 737 MAX Section

- ['Flying Blind' author says Boeing put profit ahead of safety with the 737 MAX](#) — Peter Robison interviewed by Dave Davies, NPR *Fresh Air*, November 29, 2021. **Every screenshot in this lecture is from this transcript**, and it is worth reading in full — it takes about fifteen minutes.
- [Boeing Charged with 737 Max Fraud Conspiracy](#) — U.S. Department of Justice, January 7, 2021.

Further Reading

- [Flying Blind: The 737 MAX Tragedy and the Fall of Boeing](#) — Peter Robison
- [ACM Code of Ethics and Professional Conduct](#)
- [Mars Climate Orbiter Mishap Investigation Board Phase I Report](#) — NASA

Questions?

`zives@engineering.upenn.edu` · `sharry@engineering.upenn.edu`