Vidit Pokharna

The Boeing 737 MAX Crashes: Understanding the Causes

In late 2018 and early 2019, the aviation world was shaken by the tragic crashes of two Boeing 737 MAX aircraft. These accidents, which resulted in the loss of 346 lives, were not merely technical failures but also highlighted broader systemic issues in how complex systems are designed, tested, and operated. The crashes were ultimately traced back to a critical flaw in the Maneuvering Characteristics Augmentation System (MCAS), a software system intended to enhance the plane's stability. However, the deeper issues that contributed to these tragedies are shockingly similar to the ones observed in earlier catastrophic incidents, such as those involving the Therac-25 medical device (Leveson, 1995).

The Technical Cause: A Flawed Software System

The immediate technical cause of the Boeing 737 MAX crashes was a malfunction in the MCAS, a software system designed to automatically push the plane's nose down if it detected that the aircraft was climbing at too steep an angle. The intent was to prevent a stall, but the system relied on data from a single angle of attack sensor. When this sensor failed, the MCAS activated repeatedly, forcing the plane into a nosedive despite the pilots' desperate attempts to regain control (Gates & Baker, 2019). Compounding this issue was the fact that pilots were not adequately informed or trained on the MCAS’s existence and operation, leaving them unprepared to counteract its erroneous commands (Boeing, 2019).

Relevant Causal Factors: Complexity, Testing, and User Interface Design

The tragic outcomes of the Boeing 737 MAX crashes were not solely the result of a software bug. They were the consequence of a series of decisions and oversights that endured a troubling resemblance to those identified by Nancy Leveson in her analysis of the Therac-25 accidents.

Three key factors stand out: the complexity of the system, inadequate testing and simulation, and poor user interface design (Leveson, 1995).

Complexity of the System

One of the major issues with the Therac-25 was its sheer complexity. The system was designed with a reliance on software to manage critical safety functions, a move that significantly increased the potential for unforeseen failures. Operators and engineers alike struggled to fully understand all the interactions within the system, leading to tragic outcomes (Leveson, 1995). In the case of the Boeing 737 MAX, the MCAS was similarly embedded within a highly complex network of flight control systems. The software's integration was so deep that its actions could override pilot input, yet it operated largely in the background, leaving pilots unaware of its potential to take control during critical moments (Gates & Baker, 2019). This complexity made it difficult for both the system designers and the pilots to foresee all possible failure scenarios, much like in the Therac-25.

Inadequate Testing and Simulation

Another critical factor in both the Therac-25 and Boeing 737 MAX cases was the failure to adequately test and simulate real-world conditions. For the Therac-25, testing was insufficient to uncover how the system would behave under certain simultaneous conditions, leading to deadly overdoses of radiation (Leveson, 1995). Similarly, the testing of the MCAS system did not fully simulate the conditions under which a sensor might fail or how pilots would respond to unexpected falls or declining paths. The system was not tested with the rigor needed to identify these critical vulnerabilities, allowing a single point of failure—the angle of attack sensor—to have catastrophic consequences (Gates & Baker, 2019). This inadequate testing is a stark

reminder of the dangers of assuming that software will behave as intended in every scenario, particularly in safety-critical systems.

User Interface Design

User interface design played a crucial role in both the Therac-25 and Boeing 737 MAX tragedies. The Therac-25's interface was notorious for being non-intuitive, offering little feedback to operators about what was happening within the system. Operators often had to make decisions based on cryptic error messages or vague indicators, leading to fatal mistakes (Leveson, 1995). The Boeing 737 MAX's MCAS system, while not a user interface in the traditional sense, suffered from a similar issue: it operated in a way that was not clearly communicated to the pilots. The system could activate without the pilots fully understanding what was happening or how to counteract it (Boeing, 2019). In both cases, the lack of clear, immediate feedback to the user contributed directly to the fatal outcomes. The pilots of the 737 MAX, much like the operators of the Therac-25, were left to navigate a complex system without the tools or information they needed to make life-saving decisions.

Conclusion

The Boeing 737 MAX crashes, like the Therac-25 incidents before them, underscore the critical importance of simplicity, rigorous testing, and clear communication in the design of safety-critical systems. These tragedies reveal the risks of over-reliance on complex software, particularly when that software is not thoroughly tested or when its operation is not transparent to the users who depend on it. As technology continues to advance, it is vital that we remember these lessons and apply them to prevent future catastrophes. Safety must always come before convenience or cost-saving measures, especially when lives are on the line.

References

737 MAX software update. The Boeing Company Official Website. (n.d.-b). https://www.boeing.com/commercial/737max/737-max-software-updates.page

Gates, D. (2019b, March 21). *Flawed analysis, failed oversight: How Boeing and FAA certified the suspect 737 MAX flight control system*. The Seattle Times. https://www.seattletimes.com/business/boeing-aerospace/failed-certification-faa-missed-safety-issues-in-the-737-max-system-implicated-in-the-lion-air-crash/

Leveson, N. (1995). Medical Devices: The Therac-25 Accidents. http://sunnyday.mit.edu/papers/therac.ps