Douglas Spaeth

Synopsis

The article The Lessons of Valujet 592 goes over the crash of ValuJet Flight 592 in May 1996, which killed all 110 people on board. The cause of this crash was traced back to a fire that started in front of the cargo hold, caused by oxygen generators that were not stored properly. The generators that should have been safe for travel were missing important safety caps, causing them to be a fire hazard were also packed in with cardboard boxes, tires, and many other flammable materials. When one of these ignited shortly after takeoff, one of the generators caught flame and quickly spread throughout the aircraft, disabling it and leaving the crew unable to regain control.

This incident shows how a small mistake at the maintenance level can have disastrous results. The people responsible for the oxygen generators failed to follow basic responsibilities, such as including the plastic safety caps, and still signed off that it was being done. These mistakes were not isolated examples, but rather show a much deeper problem with the system as a whole. ValuJet's business model, which was a low-cost airline, heavily relied on outsourcing to third-party websites, allowing the company to expand quickly while keeping costs low. This structure made it hard to pinpoint responsibility, created an incentive to cut corners, and also made an environment where speed was prioritised over quality.

The FAA, which is the governing body for aviation, was supposed to be a backstop to prevent things like this. But in reality failed to do anything to prevent the disaster. Inspectors had brought up the interesting practices, but their warnings were not acted upon. And because the industry prioritized growth over strict rules, leaving the FAA in a strange spot, it was unable to act before tragedy struck.

This crash was an example of what was called a system accident. Where it wasn't caused by one broken part, it instead happened when there were problems with many small parts.

How it relates to software dev

I think that this crash could relate to software development in a number of ways. Just like the aviation system, where small problems can snowball into giant catastrophic disasters. Software development involves many moving parts that all need to work together reliably. One important thing to learn from the flight situation is being able to take accountability. In software development, problems can arise when depending on 3rd party libraries or sources. Problems can arise if they are not fully understood, and they might not be noticed until they turn into a major failure.

Another takeaway could be to check all the small steps. Cus in the ValueJet situation, the small plastic caps that were missing were one of the major causes in the incident. It might be like ignoring warnings and not thoroughly testing code. Each one of the small steps might not cause a failure, but together they might cause everything to break.

The manager of a project could also be put in a similar situation to the FAA, where they need to decide if they want to prioritize speed and safety, as in the case of software development, it would be how well it runs. And project managers might face pressure to release something quickly, over overdo thorough testing. This would lead to getting short-term gain while facing long-term consequences.

And a system accident also applies to software development, as a bug in one module can cause issues in other areas, which might only be seen in combination when it is time for production. And a lot of the time, problems like this aren't the fault of a single programmer but can be the fault of multiple programmers across a large system.

The ValuJet 592 shows that big disasters usually don't come from one mistake, but many small problems adding up over time. This same concept applies to software devolopmenmt. And success also relies on being careful and looking at each step to make sure that problems get stamped out while they are small and don't turn into a disaster. And also skipping small steps and not taking the right precautions can also lead to catastrophic failure. And in both the aviation and the software development fields, it is important to prioritize safety over speed.