Sam Chami

Dr. Johnson

CMSI401: Software Engineering Lab

10/14/18

*The Lessons of ValuJet 592* outlines a deadly plane crash in 1996 that was caused not necessarily by a pilot or mechanical error, but rather a chain of events and a disregard for safety that occurred before takeoff to create a recipe for disaster. ValuJet was an up and coming airline that focused on maintaining low prices. With a newer airline and a lower revenue stream comes some shortcuts in the processes proceeding the flights. From recordings of the incident, investigators can conclude that an explosion caused an electrical failure and poisonous smoke to flood into the cockpit and cabin. It was unclear how events unfolded from here, but evidence implies that a large fire broke out and that pilots basically lost full control. The plane crashed into a swamp, disintegrating on impact. The article transitions to explaining the cause, something they call a system accident, that resulted from several errors in the weeks prior to Flight 592. When most airlines need maintenance, they hire outside companies. SabreTech, which handled ValuJet's maintenance, was hired to replace the oxygen canisters on the aircraft as they were approaching expiration. They replaced them but failed to properly install plastic safety caps around the firing pins before sending the old canisters back within Flight 592's cargo. Leading up to the flight, the plastic caps were ignored yet approved, the canisters were mislabeled as empty and then packaged alongside tires—all of which was forbidden yet still approved to enter the plane by the ramp agent. Consequently, the chemical oxygen generators exploded, burning the tires, causing the toxic gas, and eventually the death of 110 people. The article concludes that while safety is important, it will likely never be an airline's primary concern if prices remain competitive and airlines remain deregulated. The article concludes with the fact that most planes are safe, but pilot errors and mechanical failures do occur. It is the system failures that are the hardest to prevent.

Overall, the plane crash was caused by a breakdown in communication and a system with no backing accountability. When SabreTech did not have plastic safety caps, the mechanics could have informed their supervisors, inspectors, other mechanics, ValuJet or simply made a note of it on the work order. Instead, countless people who should have caught the issue either ignored it or did not know there was an issue in general. There were so many people involved with a relatively minor task, that not all information was updated and properly recorded. This issue is not exclusive to aviation. Within software engineering, communication is key. Consider a large project repository for example. Right off the bat, the environment setup, task list, commit style guide, and the branch/merge process has to be completely laid out for a smooth development process. In SabreTech's case, the equivalent was laid out. There was a process in place to allow for smooth repairs for something as simple as oxygen canisters. What gets complicated is the actual usage of the system that is in place. As the article implies, a system that is too verbose and bureaucratic with paperwork and checks is more likely to be ignored. Once ignored, a "normalization of deviance" will cause a pattern of disregard that renders parts of the system useless (Langeweische 1998). Within software engineering, it is standard to simply have a task list, complete the task on a well named branch, merge request and mark the task complete when finish, and then merge / deploy to a staging or production environment at the end of a forewarned amount of time. If SabreTech had an efficient system in place, chances are, the plastic safety caps would not be skipped over and the canisters not mislabeled because several people disregarding multiple steps in a process means the system needs to change.

The allowance of prohibited material aboard a commercial aircraft can be viewed in the same lens as software engineering Quality Assurance. Before anything gets sent out to production by a company, there should be a final check that reviews every change to the product since the last push. For ValuJet, this means a representative from their airline should have checked the air canisters currently installed and the cargo that SabreTech was sending back. If anything was not up to regulation, a delay would be necessary. Unfortunately, ValuJet had an aggressive business model which emphasized low prices, meaning "final" and "double" checks were a more skippable commodity. If something does not pass QA within software engineering, it can cause loss of clients, breach of security, and, in the worst-case scenario, errors that lead to loss of life. Within the aviation industry, many consumers would answer that the most important part of their flying experience is not that they remain safe, but that their trip is inexpensive and not delayed. ValuJet, like many others, wanted to keep their customers happy, which may have given them a reason to justify excluding safety measures. Unlike aviation, software engineering has a larger emphasis on quality assurance because pushing out an error prone product is more common issue. Every software engineer has received an error message. Not every mechanic has made a mistake that has hurt someone. Psychologically, this leads to something called the Optimism Bias, where the brain falsely assigns hope because of the statistic improbability of failure. Software engineers are more willing to follow their procedures, because it is much more common for issues to arise and many have in the past. Repair companies like SabreTech and ValuJet are more likely to glaze over checks that don't usually cause issues, until they do.

All this poses the question of if there's something that companies with important but tedious safety checks can do to ensure that they are properly performed? Software Engineers will get called out quickly if they push an error out to the company's user base because of version controls and commit history in place. In ValuJet's scenario, direct blame cannot be placed on one specific person. If a similar system of accountability was in place, workers may be more inclined to not skip steps because of the increased risk of losing of their job. In ValuJet and SabreTech's scenario, the fault can extend beyond the communication of the mechanics on the floor and to the protocol of the company, because it is just as easy to tell someone you are skipping a step if they were going to do the same thing with no repercussion.