Serena Zafiris

CMSI 401

Dr. Johnson

October 2nd, 2019

Learning from ValuJet

The article *The Lessons of ValuJet 592* by William Langewiesche discusses the crash of ValuJet flight 592 and the implications it has on safety in the airline industry as well as why things went wrong in the first place. ValuJet was a budget airline that was rapidly gaining traction in the market during the mid-eighties and nineties. ValuJet focused on making travel as cheap as possible, so safety took a backseat. The airplane itself went down in May of 1996 in the Florida Everglades due to electrical failures caused by improper cargo being kept in the hold. The cargo in question being oxygen generators that were taken out of planes due to them being expired or in need of repair. The canisters caught fire in the hold which in turn caused the wires and electrical panels of the plane to melt. This rendered piloting the plane to be impossible and the plane crash landed in the Everglades about twenty minutes after it took off. The crash itself was so violent all passengers and crew did not survive, that being over a hundred people. This crash then caused a widespread panic and anger. The FAA got involved, lawsuits were filed, and ValuJet eventually rebranded and got back into the air. The main question this article discusses is what the real cause of this crash was. Langewiesche conclusion was that it was a system accident, thus making the whole ordeal much more complicated. The author then goes in to explain how dangerous a system accident can be, and the long string of overlooked, poor decisions the lead to the practically inevitable crash of ValuJet Flight 592.

This event is not just a moment for engineers and other aircraft personnel to reflect. Nor is it purely a look into the danger of air travel and the current industry. Rather, this article allows one to look at the choices made by ValuJet and SabreTech and apply these lessons to one's own industry. While the ValuJet flight may have been caused by an egregiously overlooked safety violation, which resulted in hardware failure, the same could have happened if people on the software side of things maintained that same systematic carelessness and minimal attention to detail. For example, consider the issue of the oxygen generators being improperly capped and stored. This was not only an error on the end of the people assigned the task, but also the fault of the poor instructions left to them by upper management. This can relate to software because while designing software, one must create a software development plan as well as a software requirements document. Both documents are used to outline the base of the software clearly and with great detail. The entire product will be shaped and formed by these documents. The more precise and thought out they are, the more likely the final product will be the intended result and with little if not any errors. In the case of ValuJet, the engineer's instructions were to install the shipping cap if the generator has not been expended. Long story short, figuring this out is a complicated procedure, and this procedure was not clearly provided to the engineers nor did they bother to take the time to figure it out. This resulted in the improper storage of several oxygen generators. This same issue could relate to software because if the requirements documents and the development plans are not well written, then the code itself will not be written in a clean and coherent way. Whilst the implications may not seem that dire when coding something as insignificant as a basic application, it can be a big deal when designing things that put peoples lives at risk, such as an autopilot system.

Another main issue in the crash of ValuJet was the lack of attention paid by any of the managers or higher ups. This lack of attention spread to people lower and lower down the chain. SabreTech and ValuJet hired individuals very fast in order to keep pace with their increasingly demanding growth. While a lot of employees were completely competent and qualified to do their jobs, some were not. Since there was not an emphasis put on maintaining safety by checking in on employees and their duties, mistakes were allowed to permeate. Any inspector would let mistakes slide by. This same issue could happen in software development. If no one is checking in with developers and coordinating between developers and clients, then the product could go off the rails in terms of end goal, or severe errors could go left unchecked. One way these issues are battled in the software industry is through the practice of code peer review via pull requests. Before code is to be put into the master branch, it is best practice to have that code reviewed by at least two other members of the team. This allows for work to be checked for any errors or conflicts that might arise. This also helps other developers to have a good sense of what other members of their team is doing. While code review can slow the development process, it will save time in the long run in its prevention of devastating errors.

The Lessons of ValuJet can apply to most software industries. One prominent one that come to mind is the cybersecurity industry. While a code error within this industry may not result in the crashing of a plane, it is still vital to prevent any error. Allowing a mistake could result in a data leak which in turn could release sensitive information such as social security, credit card numbers, and the addresses of thousands of people. Since so much of todays world is run through some form of software, it is important that the software in question is invulnerable to any form of hacking or malice. In the example of iPhones, it may be easy to fall into the trap of pushing out a product as fast as humanly possible, just as ValuJet was focusing on growing as

fast as possible. However, if that phone is not properly secure, one is putting the risk of an individual's private data.

Overall, *The Lessons of ValuJet* is a critical look into the systematic mistakes that caused the tragic deaths of over a hundred people. It calls into question the ethical dilemmas that engineers face in both maintaining the safety of their builds while rushing to meet the aggressive demands of the industry. This relates well to software because with the increase in popularity of technology comes the ever-increasing demands for products and deliverables to be churned out at an alarming pace. While we will not be able to avoid all problems and accidents, it is important to look at our failures in order to prevent a repeated mistake in the future.