Ian Lizarda

Dr. Johnson

CMSI-401

2 October 2019

ValuJet Article Reflection

In William Langewiesche's article titled The Lessons of ValuJet 592, he recounts the horrific tragedy of the events that occurred at Everglades Park in 1996 through the eyes of Walton Little. On that day, Walton described how there was "no smoke, no strange engine noise, no debris in the air… no apparent deformation of the airframe…", which he noticed thanks to his experience as a private pilot. Yet, the plane managed to crash its way into the water, emanating a shock wave throughout the park. Unfortunately, this crash, which managed to kill all of the 110 people on board, was mostly a product of incompetence.

Three types of crashes can describe these plane accidents, according to Langewiesche: procedural, engineered and system accidents. Procedural accidents are the simplest of mistakes: don't fly into storms, don't take off with ice on the wings, etc. The next type of accidents, called engineered accidents, can be material or mechanical failure that should have been predicted by engineers, but were not. Lastly, system accidents are borne out of the confusion of integrating complex technologies managed by complex organizations. Things like power generation, chemical manufacturing, and nuclear-weapons control apply here. The ValuJet incident includes all three types.

The reason for the ValuJet crash was because a contractor, SabreTech, stored full chemical oxygen generators illegally and ValuJet employees failed to check. These oxygen canisters are usually stored in the top compartments for depressurization. However, they were illegally stored because they were missing a key component: their safety caps. Since the canister temperatures can go up to 500 degrees, they likely exploded due to having no safety cap. This was a fault on both SabreTech and ValuJet, because SabreTech should have never installed those oxygen generators on board, without the caps, and ValuJet should have been there to supervise it. In regards to software engineering, the first thought that came to my mind was, of course, unit testing. I've had my fair share of pushing mistakes into the master branch, but this was even more so applicable when it came to team projects. The ValuJet incident, in my head, was like a team project. ValuJet are the frontend engineers, and SabreTech are the backend engineers. ValuJet should have verified the correct information was running through, and that the build was working before deployment. How can we check code like this? Through unit testing. For ValuJet, not only did they fail to test whether things were properly stored on the plane, but even just the lack of inspecting the plane (much like inspecting code) was a huge error that could have been prevented.

This does not mean SabreTech is clear from faults. What they did was pretty much the equivalent of exposing an untested and uninspected endpoint into a backend system. The same solution applies to SabreTech–they should have unit tested to make sure that the data they were even sending was legal and worked. This was a dev team that did not care about the product they produced, and it shows in their lack of concern for literally putting oxygen generators illegally. With code, every piece of it should be tested and verified, especially all the edge cases. In a situation where literal lives are on the line, it is quite baffling how a subcontractor like SabreTech would put the most minimal effort possible.

ValuJet's accident, however, is something that also applies to another mishap in software engineering: communication. I consider testing to be the first line of defense, which doesn't necessarily rely on any communication. A developer writes a piece of code, and then write tests to verify it does what that developer wants. Another instance that this incident reminded me about is in regards to miscommunication, mainly on student projects. For my 401 project that I'm currently working on, we have many branches that eventually get pushed into master. Sometimes, however, small fixes get pushed into master immediately that are deemed to be too insignificant to push into a branch. When those changes are pushed, however, sometimes the code that gets pushed might change how things work in other people's branches. If those things are not notified or communicated, then people can have builds that fail to run properly. The solution: proper communication within the development team of what gets pushed and when.

For SabreTech and ValuJet, having proper communication would have been the second line of defense. The problem, however, was that ValuJet had English speakers and people from SabreTech had Spanish speakers. While this is not necessarily a problem, companies need to ensure that their workers can communicate properly and efficiently. If a team of frontend and backend engineers does not communicate properly, lots of time will be wasted refactoring code or fixing code that is broken. In this scenario, there was a concerning lack of communication between both parties, which led to the unfortunate plane crash.

Interestingly enough, another solution within the software development process that this incident reminded me of was having the DSP model: development, staging, and production models. Earlier I mentioned unit testing–the DSP model, in my opinion, is like having testing branches for each iteration of an application. I think every application, at some scale, should take on some form of the DSP model, as it ensures that customers get the best product every time through continually integrating, testing, and deploying the builds. If the practice is put in to never push to master, then having the dev, staging, and prod environments pretty much ensure that nothing will ever get pushed to customers that do not work.

For the ValuJet incident, this is not as applicable but the essence of having different stages and environments can still be applied. For example, the entire team can go through phases of inspection: maybe check a week before, then a few days before, then a few hours before take-off. It is never wrong to test multiple times, and having the DSP model ensures that testing is done multiple times over different branches. The ValuJet was an unfortunate occurrence, but one that shines a light on the importance of testing and communicating for all types of products–software and hardware included. Especially in software, it is important to implement good business models and practices so that developers can work efficiently and publish products that work well. These applications apply everywhere as well, and those who want to make products for people should take notes from the ValuJet incident, where not enough care, communication, and testing was implemented.