Thomas O'Brien

CMSI 402

10/10/18

ValuJet Article Summary

ValuJet 592 was a flight scheduled on May 11th, 1996 to go from Miami to Atlanta. After a successful takeoff and six minutes out from Miami, the co-pilot, Richard Hazen, made a request to immediately return to Miami. A radar controller, Jesse Fisher, quickly responded while noticing the potential urgency behind the request. Hazen informed Fisher that there was smoke in the cockpit and cabin. Fisher promptly called for the support of a supervisor and began to give various turn and altitude directions for the ValuJet aircraft. The transmission between the control station and the aircraft began to become corrupted and harder for the control station to understand. Fisher began to get the Miami airport crew ready to support ValuJet 592 for landing, but things were beginning to look grim. The aircraft took an unexpected steep dive and soon regained control. Regardless of an unintelligible transmission after that, the aircraft began to make a steep turn and soon crashed into the Everglades.

Immediately a rescue crew and investigation were deployed. In the investigation, it was revealed that numerous kinds of bad practices were happening under the ValuJet corporation. The issue behind the ValuJet592 flight was that there were expired oxygen generators stored on the aircraft that were not properly sealed. Workers put green tags on the containers that specified that they were empty and/or needed to be refurbished. When an inspection was due at ValuJet, the workers decided to move these canisters from Miami to Atlanta so the potential customers would not see them. Little did these workers realized that without the proper safety caps, these

canisters were lethal and highly sensitive to hot temperatures. This caused the canisters to catch on fire during the ValuJet 592 flight and take down the plane.

The key idea presented in the ValuJet 592 accident is the complex issue of a system accident. A system accident, also coined as a normal accident by Charles Perrow, is an accident that derives from systems that have become so complicated and deeply nested that it is prone to have unpredictable vulnerabilities. When systems are so complex and difficult to understand, bad habits begin to form. Instead of looking for the best and safest solution for numerous tasks, workers look for solutions that simply get the job done. This comes from a lack of understand of both the overall system and the individual contributions the worker is adding to the chain.

One massive danger behind this is the issue of workers completing assigned jobs with "band aid" solutions that could break if certain scenarios were to rise up and down the chain of work. Going back to the ValuJet 592 accident, the workers who signed off on the expired canisters likely did not fully understand the danger behind the expired canisters going on an aircraft without the proper safety caps. This knowledge was deeply contained within their documentation and was such an edge case that the workers likely did not believe it could be dangerous. Even with this mistake, there is a good chance that this was not the first time those workers signed off on expired oxygen generators. Thus, the workers would have the previous experience of believing this decision to not have any consequences. However, when another worker decides to ship these hazardous oxygen generators to Atlanta, another worker then decides to place them in the hull of a commercial flight, and following workers continue to make bad decisions down the chain, this can lead to the disaster of the ValuJet 592. System accidents occur when workers in the chain continue to make the incorrect decision and are not informed

that they made a mistake. These kinds of accidents are extremely dangerous due to how difficult they are to trace before an accident happens.

A system accident in the software engineering world is just as frequent as it is in complex hardware systems. Industrial scale websites like Facebook and Google have exceptionally convoluted system designs that are not understood by the average engineer. With a lack of understanding like this, it is possible to design code that manages to run properly on your local environment and even passes various unit, integration, and regression tests, but still contains some kind of flaw that can interfere with another system or simply not deploy at scale. A modern example of this was the security break discovered at Facebook on September 16th, 2018. Due to a bug in the user interface, hackers were able to grab API tokens that exposed up to fifty million user's personal information. This small bug exposed millions of people's personal data that are now in the hands of unknown hackers. Through this recent example and thousands of others ones, it is clear that a system accident is just as possible in the software engineering world.

In my opinion, it is impossible to completely prevent system accidents. Given how fast technology is growing, systems are only going to become more and more complex. With systems becoming more complex by the day, general engineers slowly become more out of touch with how applications function under the hood. Despite this inevitable fate, it is our responsibility as engineers to develop more thorough testing systems into our technology stacks.

An easy approach to building safer software is to put a larger emphasis on operational work. When developing software, there is a balance between development, programming new features for an application, and operations, improving the current code in production. By slowing down the focus on new features and turning to enhancing the current systems that are in place,

this leads to more reliable code existing in production. Slowing down changes to production is guaranteed to lead to safer software that is less likely to crash or get breached.

One lesson that software engineers could learn from the ValuJet incident is to focus on operational tolerance. ValuJet was not checking the quality of work happening at each step of the development chain. ValuJet should have been strictly managing the work happening at each level and holding the proper people accountable for mistakes. In software engineer terms, by adding in more QA Engineers, it will be easier to discover issues in software before the code goes out to production. QA Engineers help discover bugs from both new and existing software. These kinds of engineers are essential to ensuring high quality code in software today.

Another approach that should become a staple part of software development are post mortems. While engineers can write the finest code in the world, a system will always be violated at some point in its life. Therefore, it is key for engineers to run proper and organized post mortems. In a post mortem, engineers discuss the incident, how to fix the issue, and what they will do in the future in order to prevent it from occurring again. Adopting a culture like this will not only lead to securer code, but also allow engineer to learn from their mistakes and not make the same mistake twice. Post mortems are in my opinion a solid way of handling errors in our software and amending for them.

Overall, there is a lot to learn from the ValuJet 592 accident. It is a classic case of a disastrous system accident that lead to the loss of over one hundred lives. The biggest lesson that both hardware and software engineers can learn from it is that complex systems are bound to eventually fail, even if certain habits and trends are currently working. Even when software or products are running smoothly, it is the job of the engineers to constantly dig for issues within their respective system to guarantee high quality software and safety.