Kevin Patterson
10/2/2019
401
ValuJet Article

Imagine being so bad at what you did that you started a new trend in safety and regulations just to make sure that you never get to exist again. That is the story of ValuJet told in this article, however the failures of the system extended far beyond the airliner, but reflected on the FAA and all the contractors that work on airlines as well. The article starts at the end of flight 592 with an abrupt landing in the Florida Everglades. It opens with an interview from a civilian who witnessed the crash and called 911 to report the emergency. Even from the ground it was evident that the flight had experienced catastrophic failure which caused the flight to crash. From there the article moves on to describing the styles of accidents that occur. First there are procedural accidents, then there are engineering accidents, and finally there are system accidents. The author argues that the ValuJet crash was a system accident where the entire system designed to keep the plane safe and air worthy was undermined by a desire to cut costs and do “just enough” to make sure that the airline could stay in business.

In cataloging the events that led up to the crash we are also walked through the failures of the system at each step in the process. This is done not to place blame, but to understand how a series of mistakes all led up to the ultimate failure of the aircraft. It was a combination of laziness, indifference, and greed that led to unsafely handled boxes of highly flammable oxygen canisters with no safety caps to be packed, approved, and loaded onto the flight by workers who were paid to little to care about the rules. Once under the cockpit it is believed that they were on fire by takeoff, and by 6 minutes the plane was already doomed to crash.

The lessons taught by ValuJet are still relevant to this day. While it is unlikely that we will be designing systems that are responsible for human lives in the near future it is probable that at least some of my classmates will be involved in these systems within their lifetime. As programers we will face very similar problems to those that airlines face. Our goal is to test and test and test and then when we think we are done we test some more. We want to find every user case that we can think of and make sure that our program handles the input properly. Just like in airline design they are constantly testing for new conditions and scenarios that the pilots and planes will find themselves in. For both professions it is inevitable that there will be oversights that we could not predict. It then becomes our duty to recognize the problem and create a solution as swiftly as possible. Even if subconsciously, we live our lives through the mantra what can go wrong usually goes right, but eventually the error will be brought to our attention.

As a programmer this idea is very humbling. So often we strive for perfection and elegance in our programs, but you have to acknowledge that perfection is unattainable because, especially as your program is commercialized, there will be users unconsciously trying to break your code, and crackers trying to see how strong your code really is. Sometimes there will even be employees who are trying their best to undermine your work whether they are doing it consciously or not. We both face a complex structure of engineering and bureaucracy in commercial development that only adds to the complexity of creating a working system within a work system.

Technology can be a blessing because it allows for ease of use or automation for many menial tasks, but its integration is not always seamless. A great example of a failure of a programming system is the Boeing 737 MAX aircraft which had a technology that was not properly introduced, or even tested. The tech was created to help maintain balanced flight when the engines were moved, however it was not tested properly and had many system failures in the field. The readings from the instruments on the nose of the plane improperly told the program that the nose needed to directed down to maintain balanced flight, but when the program took control the pilots, who could see the error in the autopilot, could not override the program. This is a failure of multiple systems. The program and instruments used to implement the program were faulty and not tested thoroughly, this is a failure. The creation of a program instead of fixing the underlying aeronautics problem was a quick fix, and another failure of the system. Not publicizing the autopilot program, or at least educating the pilots on what the program did and how to override it was another system failure. While this example is another airplane failure it is important because it shows that even a technology based system failure is more than just the code, where we live, but a failure of the hardware around it, and the people designated to implement it. I think this is important to notice because a lot of the time your programs won't exist in a vacuum, but instead they will be implemented by people other than you, and they will be used on systems vastly different from your own.

We can learn a lot from ValuJet. We know that it might be easier to cut corners or take the easy way around our problems, but that this will only hurt us later. Errors and mistakes are inevitable, so it is our responsibility to test and test and test to make sure that we cover as many as possible. This is almost like a software existential crisis. Nothing you write will ever be good enough and users will always break it. Testing is useless because you cannot test for the new levels of idiocy that users will introduce. Regardless of the futility of the efforts of testing, it is important. I know that I struggle to write tests and that is one of my biggest flaws, but reading about ValuJet helps contextualize the necessity for them. I cannot stop all the idiots, but I can stop the obvious ones.