Simon Wroblewski

CMSI 401 Lab

Dr. Johnson

10/10/2018

**Problems and Questions Raised by ValuJet 592 Crash**

**Synopsis:**

Walton little, an engineer and private pilot, was fishing when he saw an airplane crash. He alerted the emergency dispatcher and said that a large aircraft had crash landed in the Everglades Holiday Park. He finished with the fact that they wouldn’t find the aircraft in one piece, and sure enough, he was right. Everyone in that plane perished. Although what happened was horrifically sad, raises questions about how we can stop these types of tragedies from ever happening again.

Some airplanes crash due to simple mistakes known as procedural errors, others because of engineering errors (which should have been caught during testing). ValuJet 592’s crash type was known as a “system failure.” System failure is described as an accident that occurs from the confusion and complexities that are inherent and unavoidable parts of complex systems. These are the complex comedy of errors, manifested into accidents, that ValuJet can help us learn from.

Shortly after the crash, one of the plane’s black internal data boxes was recovered and finally, analysis of the accident could take place. What the analysts discovered was that within the plane’s cargo hold resided three tires, cardboard, and old tanks of oxygen generators. ValuJet employees believed the oxygen generators were empty when they were loaded onto the plane due to a mistake. Although no one knows exactly what happened, it is likely that sometime during, before, or after takeoff, the first oxygen generator had ignited, causing a plethora of tragedies to follow, some of which, caused passengers to die in agony.

Knowing about the suffering that ensued, the public went on the hunt for the party responsible. Although many parties were at fault and received punishment, one of the main problems was the fact that the engineers involved wrote instructions to the non-engineers without translating them to laymen’s terms. In fact, the system failure that occurred during flight 592 can me summed up as a series of mis-communications and high disregard to detail. In light of all this, how can we translate these lessons such that they better our practices in software engineering?

**Reflection/Application:**

Although the article about flight 592 references a system failure that primarily deals with hardware, that does not mean we cannot derive knowledge from ValuJet’s mistake and apply them to a software engineering environment. If we act accordingly on that information, we might be able to reduce the chances of similar tragedies such as flight 592. One theme to point out from this article is the manner in which a group attempts to achieve organization in a complex company or software development team. I’d also point out the need for strong communication, not just among groups but also between all groups. Last, I’d like to address culpability. In situations as complex as this, it is difficult to assign liability to one person or party, making it difficult to give those who grieve from these accidents some sense of closure.

Attempting to achieve organization in a large and complex company is, in my opinion, analogous to a clown attempting to juggle an absurd number of items. Similarly, when creating a large piece of software, multiple groups are involved and need to be on the same page. This is where efficient, safe, and robust design methods come in handy. So, then what makes a good design method? Well it depends on what the method is for. In this case, the design method is in reference to software engineering which requires everyone to share and learn from the people around them. Good design methods would include standardizing your format when writing code in certain languages, commenting all your code with a simple yet precise description, never editing other employee’s code without permission or announcement, taking turns while adding to or removing from the master version of the source code to avoid merge conflicts, and many more.

Strong guidance and communication that is both frequent and healthy is necessary not just between a department but between all departments as well. When writing a piece of software on a large scale, communication should be similar to how a cell and its organelles exchange information from the lowest levels to highest levels at all times. In other words, not only should departments be constantly up-to-date on what everyone is doing in their department, but department heads should also be aware of all that goes on in any other department that they work with directly. In doing this, we reduce confusion and create a network of communication between all departments. This is crucial because not all software is compatible with all hardware, certain programming languages support some but not others… The list goes on and on but the point is that compatibility is key and without communication, there will be compatibility problems without a doubt.

Last but absolutely not least, culpability. When tragedy strikes, everyone wants to be able to point a finger, but when we don’t know who to point the finger at… a greater question get’s asked. The question evolves from “Who is to blame!?” to “How to we assign fault?” and the matter becomes much more intricate. Due to the inherently multi-faceted and cooperative characteristics of software engineering, blame becomes incredibly difficult to assign to any one person or even group. In fact, just this past March in Phoenix, a pedestrian was killed by an automated self-driving car being tested by Uber. Just ten days later, Uber reached a settlement but insisted it be referred to as “resolved”, rather than settled, due to the fact that no one admitted liability. This is the perfect example of a horrible tragedy that, once happened, had no one to point the finger at other than a company owner who was indirectly involved but did not make for a satisfying conviction. In the future, it would be nice to see some groundwork for liability on those who sign on of allowing things that have the potential to be harmful. This would force them to risk something and, in all likelihood, cause them to think twice before undergoing such potentially dangerous experiments.

As a result of these former arguments, a closing statement will be made. As time moves forward, so does our technology. However, that does not imply that our politics move at the same rate. In fact, it is becoming more and more obvious to the public that our legal system just simply isn’t prepared for the advancements science is and will continue to take. No time to act is better than the present, now is the time to have specialists on a large scale congregate and postulate solutions to this illusive issue.