Moriah Tolliver

Software Engineering Lab

October 2, 2019

**Response to “The Lessons of ValuJet 592”**

In his article, “The Lessons of ValuJet 592”, William Langewiesche argues that the ValuJet crash of May of 1996 was not due to a procedural mistake, or due to faulty engineering, but was instead a “system accident which may lie beyond the reach of conventional solution.” According to Langewiesche, it is hard to pinpoint the cause of a system accident which also makes it extremely difficult to develop solutions for. As opposed to procedural or engineering solutions where regulations can be put in place to prevent what caused them, in system accidents, the systems in place and the interconnectedness of these systems collectively causes accidents, so even if a cause were to be found, it would be difficult to reform that part of the system without reconstructing the entire system. This system could be an entire organization or in the case of ValuJet as Langewiesche argues, the entire industry.

After describing the events of the flight leading up to the crash, Langewiesche further describes a system accident by explaining contributing factors like daily complacency in the workplace and the use of difficult language and industry jargon. He writes, “The people involved do not consciously trade safety for money or convenience, but they inevitably make a lot of bad little choices.” In this particular example, he explains the many parts of the system put into place to prevent such accidents. Firstly, at SabreTech, a contractor of ValuJet, put unlabeled and improperly capped oxygen generators in a place where ValuJet shipments are stored. The shipper then came to do their job and simply packed all of the ValuJet objects into a single box to be shipped out. This box was then not inspected and was improperly labeled by another person at the company. Then another failure occurred when the ValuJet ramp agent did not inspect if the package was labeled correctly because this agent trusted that it would be. All of these factors and seemingly small steps contributed to a fatal and tragic event.

While many of the particular occurrences that contributed to the crash of ValuJet 592 consisted of hardware problems or knowledge, the same systems in place in the airline industry can be seen in the tech industry and in software engineering. As Langewiesche explains, “ValuJet flight 592 burned and crashed not because the airplane failed but, in large part, because the airline did.” Often, the software that gets sent out to clients or users is not *technically* wrong, meaning it passes all the tests and checks all the boxes that the engineers set out to accomplish. However, there are so many steps in terms of delivering and using this software that can go awry and that can ultimately lead to failures. In most designed systems, there is no way to guarantee complete safety or accuracy all of the time, but knowing this, companies should have systems in place that are designed to catch these flaws before they affect the lives of their customers, users, or employees.

One problem Langwiesche explains that I find particularly applicable to software engineering is “engineer speak”. These are terms and jargon that are fine to use between engineers, but when they are used to communicate with people outside of the development process like users or clients, or even non-technical people inside of the development process, these terms just leads to confusion and this confusion can be catastrophic. Software engineers tend to be surrounded by those who are familiar with specific tech jargon most of the time, and because of this, certain words may start to seem like common knowledge, while in reality they will not be common knowledge for everyone who needs to interact with these documents.

Another idea Langewiesche described that I found particularly applicable to software engineering is the idea of “pencil whipping.” He explains how this contributed to the failure of the flight:

What they did not do was place the required plastic safety caps over the firing pins--a precaution spelled out on the second line of ValuJet’s written work order...At the end of the job, in the rush to complete batches of paperwork on all three MD-80s, two mechanics routinely ‘pencil-whipped’ the problem by signing off on the safety-cap line as well as on the others, certifying that the work had been done.

In this example, Langewiesche notes that the mechanics were rushing to finish paperwork and as a result, simply signed every line, not really considering whether or not those requirements had been fulfilled. I have been guilty of a similar at while signing up for a new account on a phone application as I suspect many people have, and these mechanics were simply adhering to a habit. In this example, it was people on the production side that were doing the pencil whipping, while in software engineering it is often users or clients that quickly scroll to the end of a privacy agreement or similar document and select “agree.” However, this again goes back to confusing jargon and presenting information in a long and hard-to-read format that is easily passable. Instead, this information could be written more succinctly in a way that encourages the user to be informed by what they see instead of encouraging them to give up and just sign.

The problem of solving such mistakes, is that they are not isolated events, but the consequence of a lot of little events that are hard to track. As Langewiesche explains, the traditional fix of adding more regulations simply adds to the systems already contributing to the problem. However, there is a larger attitude within most industries that could potentially be addressed that langewiesche sums up in this way: “The falsification [ValuJet] committed was part of a larger deception--the creation of an entire pretend reality that includes unworkable chains of command, unlearnable training program, unreadable manuals, and the fiction of regulations, checks, and controls.” The problem here is not the existence of manuals, checks, controls or the lack thereof, but of the environment in which people pretend that signing off on the paperwork means that it has been completed. The mistakes do not only arise because people failed to follow certain procedures, but because no one dared check that these procedures were followed correctly and the way the checks are invoked really lend themselves to being ignored. It is not a concrete solution, but one way to prevent systems accidents is to always investigate the system. Do not simply accept that a package is safe because it has been marked that way, or that an application does not share user data because a little box was checked when the user made an account, but truly investigate what is going on behind the scenes in order to help users and developers alike, not resort to habits of omitting or overlooking details.

**Bibliography**

"The Lessons of ValuJet 592" by William Langewiesche. Atlantic Monthly, March, 1998. (pp 81-98).