CMSI 4071

Eliasi, Galvin, Gutierrez, and Hunter

Section 1: Synopsis

In May 1996, a plane crash involving a Valujet aircraft occurred at the Everglades Holiday Park, shortly after taking off from Miami International Airport and heading towards Atlanta, Georgia. The plane hit the surface with so much force that it shattered on impact, killing both pilots, 3 flight attendants, and 105 passengers. This accident occurred as a result of a fire that began in the cargo hold from improperly packed oxygen generators, which were a result of safety shortcuts and negligence from various people along the way. The investigation after the accident revealed that the plane burned not because the airplane itself failed, but rather the airline did. After ValuJet gave an order to replace oxygen generators from refurbished planes, hired outside workers from SabreTech neglected to place the required safety caps over the firing pins, despite it being on the work order. As the author explains, no one at SabreTech had such caps nor did they care much about looking for them, resulting in the caps being forgotten and supervisors signing off on the work, with no thought about the caps. Eventually, the boxes containing the generators were prepared to be shipped off to Atlanta and loaded onto the plane despite federal regulations and the fact that ValuJet was not designated to carry hazardous materials. Then, either during taxiing or on takeoff, the first oxygen generator ignited, starting the fire in the cargo hold that eventually brought the plane down. This fire spread, filling the cabin and cockpit with thick, black smoke and melting electrical panels. The plane dropped 6,400 feet in 32 seconds, going 500 miles per hour as it dove down towards the swamp in the Everglades.

Even prior to the crash, inspectors were concerned about ValuJet's quick expansion, citing that they had neither the procedures nor the people to maintain safety standards. This

proved to be correct as supervisors and inspectors failed by not providing the required safety caps for the oxygen generators or even verifying that they were actually being used. If they had, despite all other errors, perhaps the plane would not have burned. In the end, the investigation served as a reminder to the other airlines of the consequences of cutting corners and costs.

The author also discussed how plane accidents can be classified and the differences between them. For instance, procedural accidents are the result from single obvious mistakes that should have had simple resolutions, while engineered accidents result from material failures that engineers should have anticipated such as mechanical failures that could be debugged. The third category, system accidents, lie beyond the reach of conventional solutions and are born from confusions within complex organizations such as contractors or the government. The one thing that was clear was that safety isn't the first priority for airlines but rather profit, money, and convenience. The author also emphasized that occasional crashes are inevitable, unless we are willing to end our affordable airline system. In the end, solutions may end up increasing the risk of accidents by adding to the complexity and obscurity of the industry.

Section 2: Application to Software Engineering

Throughout the software engineering industry there are numerous examples of system accidents that, similarly to the Valujet crash, are caused by a tangle of confusions that arise in a complicated system. Tech giants and start-ups alike face pressures like those that airlines do, to cut costs and maximize shareholder profits. These motivations can prevent companies from prioritizing the prevention of failures through practices like constructing robust cybersecurity systems or performing proper code reviews. In a large tech corporation, tall hierarchical structures can lead to situations similar to that which occurred in the Valujet case between the

airline, their contractor SabreTech, and the FAA. When SabreTech was handling the oxygen canisters, there were a collection of missteps that accumulated and eventually created the conditions for the accident to occur. The number of individuals involved and the division of responsibilities between a variety of roles allowed safety measures to be mismanaged, with no one fully comprehending the danger that was involved in the shipping process. On large software engineering teams, similar negligence can occur when responsibility is shirked, with engineers at each step of the way not taking responsibility and performing measures like proper testing or code reviews. If everyone assumes that someone else will take charge, it can lead to situations where there is a complete failure to consider edgecases.

Another issue related to the Valujet crash that transfers over to software engineering failures is the use of engineer-speak in a way that prevents proper communication of dangers and required safety measures to non-technical stakeholders. The FAA's regulations and the plane's maintenance guide were written in language that made it extremely difficult and unlikely a mechanic would understand the importance of the safety caps in relation to the oxygen canisters, and therefore they were ignored in favor of moving along with the process. In software engineering, a similar situation could occur where an engineer who is an expert in a certain niche topic writes a document that is extremely technical and difficult to understand. Then if that engineer were to leave the project and another engineer less familiar with the ins and outs of the topic were to take over, it is possible that this language could lead to future failures. While hopefully proper communication and research would take place, in a high pressure organization with quick deadlines, corners may be cut and errors could occur that would have been avoided if the original author had sought to lay out dangers more clearly.

These types of issues can chiefly be prevented by dedicating time to proper training on clear communication, and incentivizing robust testing practices. If a team's culture allows members to feel comfortable asking questions and raising concerns about edgecases, it is more likely that these types of system accidents can be avoided. These failures occur when pushing lots of code quickly is valued more than ensuring the quality of the product. This dilemma is one faced in almost every industry, but it is especially prominent in tech where companies can experience rapid growth and intense competition. Like airline workers, software engineers have to be mindful of the danger that a seemingly small decision could lead to, such as a weakness that could allow for a breach in security and a loss of user privacy.