Samuel Savini

This article discusses the disaster that occured in May of 1996 to a commercial airliner during a flight. The procedures followed by multiple agencies and workers resulted in a failure to ensure airplane safety, and eventually led to the death of over a hundred people, both passengers and crew. The cause of the crash sparks a discussion on the flaws in the safety measures that are supposed to protect those using the aircraft. Certain methods for communicating the state of airplane functions and parts are seen as main reasons that this disaster took place in the first place.

ValuJet 592 was on its second flight of the day. This particular airplane was older than most, as was the entire fleet that the ValuJet company owned. ValuJet was known as a company that took a leisurely approach to the flight process. Their employees wore simple clothing, and their logo was a cartoon airplane known as a 'critter'. The state of safety measures that the airline took before departing were similar. A crucial mistake was made prior to takeoff that resulted in the airplane's demise. Before flight 592 was even departed, the ground crew loaded into it more than a hundred oxygen generators in a cardboard box covered with a thin layer of bubble wrap. What the loading crew did not realize was that these generators were in an extremely volatile state, and could ignite at any moment. The flight took off, and several minutes into the flight, the pilots began to notice that their systems were shutting off. After hearing a distinct pop underneath the airplane, the pilots could only watch as their airplane's systems failed. As the pilots are trying to figure out how to control the airplane, the fire from below the vessel had spread smoke into the main cabin, and eventually the fire itself. As passengers ran to the back of the airplane to escape the flames, the inferno only grew. Flight attendants broke through the

cockpit doors to alert the pilots of the problem occuring behind them. Opening the cockpit enabled the smoke from outside to seep in. To avoid more oxygen worsening the fire, the pilots did not deploy oxygen masks, even for themselves. This dire choice led to the inhalation of the dangerous fumes. Most of the passengers had died from suffocation before the airplane had even touched the ground. The pilots lost consciousness, and the airplane flew nose first into the Florida marshlands. There were no survivors. The ultimate cause of this tragedy was contained in a cardboard box underneath the main cabin, but how did this get so out of hand?

Oxygen generators were designed by engineers to be understandable...to other engineers. The canisters had no warnings on them about the dangers of loading them into an aircraft, and the measures taken to symbolize whether they were safe or not were vague at best. Now, if an engineer had loaded these generators into the airplane, there would have been less of a problem. They probably would have figured out the ones that were depleted contained a hazardous material that commercial airliners do not have the licence to transport, and that the full and unused yet expired generators in the box could be transported with a covering over the trigger pin. Those generators that were new and unused also required a safety covering to prevent a pin firing unintentionally. This is the problem however; the safety measures for the generators were not designed for a regular airline worker to understand easily. They were made by engineers in a way that was clear to them, which in no way is clear to others. The generators also had a green tag attached to them, which symbolized them being ‘unusable’. This means they are expired and although ready for use, cannot be used anymore. A worker understood that the green tag meant that a generator was ‘defective’, which they took to mean that the generator was empty, and

therefore could not pose a threat if transported. This miscommunication between safety designers and workers poses a threat to all functions of an aircraft, and is known as a 'system error'.

What does this article mean to me in terms of my future as a computer scientist? It shows me that I have a responsibility to make sure my code is understandable and clear not only for myself, but for a worker outside who is trying to understand it. My work cannot be designed so that it is easy for me to discuss with a fellow software engineer. Rather, it must be transparent to anyone attempting to understand it, so that they can effectively use and utilize the technology without putting anyone in danger.

This article also taught me something about safety measures. Every program, every piece of code that is put into the market is put through safety measures. These are to ensure the capabilities of the code, and to prevent any auxiliary problems from occurring in the future. However, I have learned through this article that adding complexities into a system, whether it is a safety measure or something else, naturally makes it more difficult to understand from an outsiders point of view. Making a program more difficult to understand in turn leads to more problems with the system in the future, which could potentially be catastrophic. From this I take the lesson to make my code as simple as it possibly could be, and if there are complications that must be instituted in order for it to function, then they should be easily understandable from an outsiders point of view.

I am taking a lesson with me from learning about this article that I have believed for quite some time now. As a student in Computer Science who entered the field as a complete novice, I am constantly baffled by the amount of 'tech talk' that occurs in the lab, my classes, and even in casual conversation among CS students. Every class I have had in Computer Science, without

fail, has addressed parts of Computer Science that are completely foreign to me as a novice in ways that imply that I should have an understanding of it. Names of companies, names of languages, names of architectures are thrown around like commonspeak. In my situation, this kind of conversation is extremely confusing and intimidating. I struggle to keep up with others in my classes because I feel a sense of disconnection from them. When I code, I make it my personal mission to make my work as readable as possible, because I am not coding for others; I am coding for me. As someone who is aware of the increasing number of people in CS struggling with ‘imposter syndrome’, or a natural feeling like they may not belong with the others in their major, I strive to reach a hand to the novice. If technology as a whole is going to take over as an industry, we need to make our work understandable to everyone, not just fellow developers. That is the lesson I am taking from this article.