John Goocher

Response to “The Lessons of ValuJet 592”

Synopsis

There are three types of airplane accidents discussed: procedural, engineered, and system accident. Procedural accidents are “operator error” in which a single person fails to correctly follow procedure. Engineered accidents describe failures caused by material or subsystem failures that should have been predicted by the engineers or caught by test pilots. The last type are system accidents, these come from confusion and dysfunction within the large organizations that oversee humanities most ambitious and complicated undertakings.

The plane was carrying five boxes of spare oxygen containers that were improperly stored, a practice that seemed largely accepted in the industry. In accordance with federal regulation, the cargo should have been rejected by the ramp agent. The co-pilot was made aware of the cargo, but did not act to prevent it from being loaded. Leading up to the crash, the FAA had been reviewing Valujet and had failed to fully address the concerns of its inspectors; including, a recommendation that Valujet should be required to be recertified.

Langewiesche describes the ambiguity of the instructions for removal, storage, and disposal of the MD-80 oxygen containers. One of the most preventable failures was the lack of plastic caps being supplied to cover the firing pins. This was a short cut taken by the supervisors and had seemingly become standard practice at SabreTech. Vaughan called this “the normalization of deviance,” the concept that short cuts and relaxation of rules happens overtime. Langewiesche reiterates Perrow’s claim that “what can go wrong usually goes right” and how this leads people to draw wrong conclusions. It is apparent that the failures leading up to and resulting in the crash from the highest level to the lowest level were caused by a system failure, to hold workers to a high standard and avoid complacency.

Going Further

The failures in communication, documentation, assumptions, and corner cutting can happen in any project. These obstacles are always present and the task of balancing speed, cost, and safety is always going to cause the hard choices. The hardest part of safety are accounting for unforeseen failures from design to implementation. Even in the crash, there were multiple safe guards to prevent such a tragedy; no one party could have foreseen the impact of their actions. It is unreasonable to expect someone to believe their day-to-day actions would endanger or kill someone, given this reality, it is important that the work environment emphasizes the “correct” level of responsibility and holds employees to that standard. Correct is a loaded term by which I am trying the importance of having each employee feeling they need to complete their work to an attainable and reasonable standard. Creating too low of a standard is obviously dangerous, but creating too high of a standard forces the workers to “normalize deviance”.

To try and avoid these pitfalls, it is important to avoid using outside contractors when possible, and when contractors are required it is important to hold them to a high standard and pay them a reasonable amount for the work. Having everyone under the same roof further minimizes the possibility of system failures. It is important to grow business infrastructure when growing other aspects of a project or business. As more people become involved on a project it is important that communication there are proper channels and forums for communication. In small to medium projects, the SCUM development method is great at keeping the team communicating and on the same page. As projects grow, the AGILE method allows for project to be quantified easier. This is helpful for traditional management structures.

Looking outside of the initial implementation, it is equally important that the software is maintainable. After the software leaves the shop, the usage of the product can and likely will exceed your expectations for deployment time and how it is used. Creating comprehensive documentation is important for

allowing the project to be used and updated quickly and safely. In addition to following the best practices for documentation, it is important to thoroughly test the implementation. To this end, tests should cover a wide variety use cases and performed regularly. This requires ongoing testing during the development cycle, along with enforcing current best programming practices. Having organization wide style guides and linters help people from falling victim to confirmation bias.

After completing the best work possible internally, it is important to have outside review for sensitive and potentially dangerous projects. Outside review offers a level of scrutiny that is difficult for even the best development teams to obtain. Along with this, having as many collaborators as possible minimizes the possibility for unintended behavior to make it into the final launch, being a popular open source project helps smaller teams successfully develop larger projects. As a final way to get outside help, it is beneficial to offer bounties for vulnerabilities. Tesla and Ethereum both offer cash rewards for finding vulnerabilities along with more incentives to help develop fixes for the vulnerabilities. With a mixture of these tools a project can do its absolute best to bring a fully actualized product to market.

Outside the failures within a system, it is important to try and fully understand the impact that the intended product will have. Personally, I continue to contemplate and weigh the pro and cons of the debased system. In a perfect implementation, any data stored would be irremovable, untraceable, and unmanipulable. This has meaningful and ostensibly good implications for the “good” companies and user, less data loss and theft. These are the causes that will create a better internet and world. On the reverse, any data can be saved, transmitted, and protected for entities with malicious intentions. Given the nature of the system there is no way to avoid this reality. Bitcoin has been and continues to be used by cartels and gangs. The debased system similarly, cannot discriminate in its usage. It is a reality that DMCA takedowns cannot be leveraged against copyrighted content, illegal data will be easier to disperse. Alongside this negative, is the positive that tyrannical governments cannot censor data on

debased. Ultimately, I firmly believe debased's ability to combat the erosion of freedoms in the technological age and offer asylum from surveillance states greatly outweigh the detrimental use cases.