Reliability, the Key to Software

Bradley Harper

Cookeville, TN

[brharper42@tntech.edu](mailto:brharper42@tntech.edu)

In the modern age, technology and software has taken over the world and spread like wildfire. While this isn’t necessarily a bad thing, the influence of both software and technology has a very vast impact on its countless users. Software is involved in various controversial topics such as Boeing’s airlines, state voting machines, social media applications, databases storing confidential information, and many others. Software is so abundantly used that some jobs and individuals rely on it for their day-to-day business and lives. This leads to the issue of reliability. Being able to rely on something is a key element for using such a type of software. This brings up the following ethical question: Who is responsible when there is malfunction of software, the creator/distributor or the user/client? Obviously no reasonable person would blame an account holder for loss of information if a large data leak or breach occurred. This leaves one party to carry the blame: the creator. Occasionally while planning, it is unclear who all will use the product being made; other times, the vision of what the product may be used for is blurred. Either way, it is clear that no one person can view all angles of possibility. Reliability can be overlooked in many ways such as for profit, by mistake, or from ignorance. According to [1], Rule Utilitarianism considers the long-term consequences or “whether adopting those moral rules could lead to the greatest increase in total happiness over all affected parties.” Rule Utilitarianism can play a pretty significant part in identifying the issue of reliability.

Boeing is the leading aerospace manufacturer in the world. Countless Boeing airlines breach the skyline each day carrying even more people. Due to the massive use of airlines supplied by Boeing, one would expect top of the line safety measures. In the case of the Boeing 737 Max, Boeing did not take the maximum precautions for handling safety features. According to [2], “A senior Boeing engineer filed an internal ethics complaint this year saying that during the development of the 737 Max jet the company had rejected a safety system to minimize costs, equipment that he felt could have reduced risks that contributed to two fatal crashes.” In order to lower how much the company spends on the 737 Max, they cut corners which could potentially endanger the passengers aboard the jet. A few more testimonies from other employees, former and current, at Boeing saying that “managers dismissed engineers’ recommendations or put a priority on profits,” says [2]. Rule Utilitarianism goes directly against the current that Boeing displays in this case. By cutting corners, Boeing completely ignores happiness in the long run and for the majority of people. Further in the article, Mr. Ewbank is asked about a type of automated software in the 737 Max. In both noted crashes of the airlines, this automated software is believed to have sent false data. In turn, the software “known as MCAS, then activated erroneously, sending the planes into irrecoverable nose dives.” [2] continues by saying that A chief test pilot of the 737 would have liked to study adding a simulated airspeed, but “a Boeing executive decided not to look into the matter because of its potential cost and effect on training requirements for pilots.”

Profit is not the only reason reliability may not be upheld. In the case of Gitlab, a major code sharing resource, reliability fell upon an employee according to [3]. During a late night shift, an employee of Gitlab deleted a folder with hundreds of Gigabytes of data. The employee realized what he had done and saved only 4.5 gigabytes says [3]. The article continues to say “the last potentially viable backup was taken six hours beforehand.” While clearly, the employee is the only person who could be held responsible, one would also wonder how to put blame on someone doing something wrong unintentionally. [3] continues by saying “that they knew about possible data protection techniques, ranging from volume snapshots to replication, and backup and recovery.” The author, Jeannie Liou, also states that it is not clear that they had the proper expertise or data protection/recovery. “Gitlab may have experienced data loss for any one of the following reasons: Neglecting backup and recovery specific tools in favor of home-grown scripts, delaying the deployment of backup and recovery tools until a data loss has occurred, Not performing a thorough analysis of business requirements before choosing a solution that can meet those application uptime requirements, or not performing a thorough analysis of business requirements before choosing a solution that can meet those application uptime requirements.” By failing to upgrade or update methods of handling data loss, Rule Utilitarianism sees this as clearly not making an attempt to increase overall satisfaction in the long run. More issues tend to arise as time goes on but adaptations are not made.

One of the least common reasons for reliability to be broken is ignorance. If one does not know why software does not work, then they cannot strive to fix the issues. This generally leads to consumer issues as well as distrust in a company or creator. We can see this in the Google Nest. According to [4], “It’s still not entirely clear where the issue comes from. Some users in the forum threads have said Google’s support reps are pointing the finger at a recent update that disabled the device’s Wi-Fi chip, but Google itself isn’t saying.” For a product like the Nest, meant to control the temperature of your home from your smartphone, to not work properly is a huge reason to turn away from the product. Whether Google knows what is causing the issue or not, the software is simply not reliable enough for consumers to gain interest in the Nest. Although the Nest is still usable manually, there is no clear purpose for it if one cannot use the desired benefit.

Trusting in the software provided by the market can be a large risk. Whether you decide to use a database or take a plane ride is entirely up to the individual. Who you choose to trust is entirely up to you. Therefore you can consider not trusting a company’s software as a reliable safety method after all. In many cases, a backup plan will generally bring the most happiness overall such as extra saves in code, or parachutes on planes. In the end, it’s always best to rely on yourself.

Works Cited

[1] M. Quinn *Ethics for the information age,* 7th ed. Hoboken, NJ, USA: Pearson Education Limited, 2015, pp. 80-83.

[2] N. Kitroeff, D. Gelles, and J. Nicas *Boeing 737 Max Safety System Was Vetoed, Engineer Says.* The New York Times. <https://www.nytimes.com/2019/10/02/business/boeing-737-max-crashes.html> (accessed Nov 16, 2020).

[3] J Liou *The Perfect Storm: Gitlab Data Loss.* DZone. <https://dzone.com/articles/a-perfect-storm-mission-critical-application-human> (accessed Nov 16, 2020).

[4] N. Statt *Some Nest thermostats can no longer connect to the internet, so Google is replacing them.* The Verge. <https://www.theverge.com/2020/7/24/21337669/google-nest-thermostat-w5-wifi-issue-remote-control-disable-replacement> (accessed Nov 16, 2020).