Utkarsh Jain
September 22, 2019

Software Engineering Disasters

CS-C3150 Software Engineering

"To err is human, but to really foul things up you need a computer." (Paul Ehrlich)

The last half-century has seen a huge change in the way that complex things are made. Before that, the exemplar of a sophisticated machine was a watch. Since then it has been the computer. Today software is used in a wide range of activities, from huge industries to entertainment, from large scale computing to micro-processors. It is knitted with our lives so much that we use it without knowing. Without Software Engineering, computer systems wouldn't have any functionality. But is it always right to trust a software? Is it right to expect them to work perfectly every time? The answer is a big NO! Softwares are designed and developed by humans, thus they are prone to all kinds of error which we can make. Software errors have caused a lot of loss to the world economy. Faulty software is expensive, destructive and embarrassing. This essay highlights some of the biggest software disasters in the history of mankind and also gives an account of the huge losses we have suffered due to them.

Ariane 5

It took almost a decade and billions for the European space agency to build the rocket, Ariane 5, which was supposed to give them dominance in the commercial space business. The rocket launched and within moments it exploded mid-air. Reason? A small software bug which was trying to stuff a 64-bit number into a 16-bit space. (Gleick, 1996)

At 39 seconds after take-off, a self-destruct mechanism initiated in Ariane 5. The mechanism started because of the aerodynamic drag the boosters of the rocket were experiencing. The rocket was trying to make a course correction that was not needed. Steering was controlled by an on-board computer which thought that the rocket needed course correction because of the data it was receiving from the inertial guidance system. The data looked like an impossible and absurd flight data but were actually an error message. The guidance system had in-fact shutdown. This shutdown occurred after 36 seconds of take-off when the guidance system's

on-board computer tried to covert the 64-bit number into a 16-bit number. The number was greater than 32,767 and thus the conversion failed. After shutting down the control was passed to an identical, but a secondary, system which was there to provide backup in case of a failure. But the second unit shut down the same way the first one did. And why not? Both of them were running on the same software.

Normally when a program tries to convert data from one type to another, the conversions are usually protected by some extra lines of code. Indeed, many of the data conversions happening inside the guidance system's programming included the protections. Unfortunately, in this case, the engineers decided not to provide any such protection because they thought that the data would never be too large enough to cause any such problem.

Mariner 1

Mariner 1 was America's first spacecraft in the Mariner space program which was launched on July 22, 1962 to collect a variety of scientific data out Venus during a flyby. Just over 290 seconds after take-off, a range security offer ordered a destructive abort as the rocket veered off course. Faults in the guidance system made steering of the spacecraft impossible, which were directing the spacecraft towards a crash somewhere in the shipping lanes of North Atlantic. The abort was called six seconds before separation after which it could not be possible. (Wikipedia, 2017)

Improper functioning of the airborne beacon caused multiple periods of time without a rate signal. A missing hyphen in the coded computer instructions in the data-editing program allowed transmission of incorrect guidance signals. When the airborne beacon wasn't working, the code error caused to accept the sweep frequency of ground signals as it sought the vehicle beacon signals and combined both of them before sending them to the guidance computation. This led to an unintended course correction without proper steering commands.

Speculations after the crash reveal that the NASA built this program hastily in a space race which was supposed to keep up with the Soviets. The cost of this program was about 18 million dollars which were quite a big amount that time and this made the absence of a "hyphen" a costly mistake.(Wikipedia, 2017)

How can we prevent such disasters?

I feel that the very important and the first step while developing software is to be patient and calm. Though it may not have been the major reason behind the crash of Mariner 1, the competition to keep up with the Soviets somewhere did pollute NASA's intentions and also caused them to lose their calmness. It's equally important to understand what is required to be done. There shouldn't be any communication gap between two people who are working on the same project. Sometimes poor communication between two colleagues can cause misinterpretation of each other's goals which can lead to unwanted and fatal errors. Estimating the available resources is an important factor which can help you plan and steer the project. Planning to implement something which is out of your budget or cannot be done in the specified time will only produce poor software. Another important process is the rigorous testing of the software. Proper and thorough testing of the software is necessary to pin-point all the corner cases where the software may fail. Simulating the Ariane 5 software on extreme cases would have uncovered the conversion failure which caused a loss of a lot of money and hopes. We have already seen that the above-mentioned disasters have occurred due to the poor performance of the software on runtime. These errors need to be caught before delivery.

Learning goals from this course

I have taken this course, Software Engineering, due to the following reason:

- I'm a third-year undergraduate student and the day isn't very far when I would work with bigger firms on bigger projects. Some of these projects would be under the evolution phase or some of them may as well be under development phase. Thus a proper knowledge of Software Engineering would be very valuable and would help me contribute to them.
- To learn about the processes involved in developing a real-life project. I have worked on various academic projects, but never really on a project which is used by real-world by people. Knowing how software works, developed and maintained excites me a lot.

- I expect some kind of course-end project wherein I could work with my course mates in a team to develop or evolve a software. This would provide me with the exposure I need before I start working on real-life projects.

References

GLEICK, James, 1996. *A bug and a crash - sometime a bug is more than a nuisance* [online]. [viewed 21 September 2019]. Available from: https://around.com/ariane.html

HIGGINS, Chris, 2017. *On This Day in 1962, NASA Launched and Destroyed Mariner 1* [online]. [viewed 22 September 2019]. Available from: http://mentalfloss.com/article/502943/day-1962-nasa-launched-and-destroyed-mariner-1

WIKIPEDIA, 2017. *Mariner 1 --- Wikipedia, The Free Encyclopaedia* [online]. [viewed 21 September 2019]. Available from: https://en.wikipedia.org/wiki/Mariner_1