ECE 322

Assignment 1

Arun Woosaree
September 17, 2019

Arun Woosaree XXXXXXX

1

After reading the two papers, the two most essential factors which make software testing difficult in my opinion is:

1. Human nature
   In my opinion, people are generally lazy. In terms of software testing, this means that in some cases, people will tend to write enough tests to convince themselves that their program is correct, or In some cases they might even refuse to write tests once they have some code that in their mind already works, so there is no percieved need to test the software. This is probably because most programmers experience a significant mental reward when they finally get a program 'working'. However, when it comes to writing tests for the code they just wrote, there usually is much less excitement. Sometimes, programmers may even find themselves wrestling with a testing framework, which can be a frustrating experience when (they think that) their program is already working as expected.
2. the always changing nature of software

2

- Functionality
  - Does the software allow the vehicle to work autonomously, without human input?
  - Does it do so in an acceptable manner?
  - (i.e. does it reach the destination in about the same amount of time as a good human driver would do or better?
    Does it do it as safe as, or better than a good human driver?)
- Performance and reliability
  - How reliably does the software react to its environment?
  - Does the software still control the vehicle in an acceptable manner in more difficult situations?
  - If certain conditions like heavier traffic puts a higher load on the processing unit for the system, does the softare still behave reliably?
  - It should work in different driving conditions (e.g. highways vs in-city, sunny vs slippery roads)
  - Will the software perform just as reliably a few years from now?
- Efficiency
  - Does the software utilize its resources efficiently?

  - Does the software respond quickly enough to its environment?

- Maintainability
  - Is the software well-documented?
  - How easy is it to add functionality to the software? (e.g. if new driving laws have to be followed, how easy will it be to add a patch to be in compliance)
  - How much technical debt exists in the software project
  - When faults are found, how easy is it to fix them?

- Usability
  - From the user's perspective, how easy is it to use the autonomous mode of the vehicle, and how is the user experience?
  - Also, how easy it is to switch between autonomous and manual modes of the vehicle?

- Portability
  - Can the software be used in multiple types of vehicles?
  - (e.g. if there are multiple models of cars, can the same software be used with all of them?)

Risk Category	Technical Risk	Business Risk
Car might crash while in autonomous mode		Might get sued (1)
Poor quality tests	(1)	
Poor test coverage	(2)	
Poorly defined requirements	(1)	

2 a) delet this

technical risk is the likelihood that a fault might exist
assessment: system architect, designer, senior programmers
business risk is the impact of a given ault might have on the users, customers, and other stakeholders
1 - very high 5- very low risk

3

1. Reliability (Operation factor)
2. Interoperability (Transition factor)
3. Usability (Operation factor)
4. Efficiency (Operation factor)

4

Feature Description

Boeing 737 MAX flight control system

Nature of Software Failure

Any testing efforts regarding the failure?

apparently it was "only" classified as a "major failure," meaning that it could cause physical distress to people on the plane, but not death. [1]

Virtual simulations were done, and the issue of the plane repeatedly nosediving was noticed in 2017, before the plane was certified. However, "at that time, Boeing concluded that pilots could overcome the nose-down movement by performing a procedure to shut off the motor driving the stabilizer movement."[2]

Any follow up action taken? Any plan to alleviate further problems?

5 Notes

To make their new plane more efficient, Boeing added larger engines to the 737 MAX. To compensate for ground clearance, they moved the engines further in front of the wings, which makes it so that the engines can provide lift at higher angles of attack, in addition to the wings, which makes changing the plane's pitch more difficult. Boeing's "solution" to this problem was in their software, so called "Maneuvering Characteristics Augmentation System"[3] The software's purpose is to make the plane tilt down if it "thinks" that the plane is going to stall. Unfortunateley, when the software takes this corrective action, it does not give pilots an easy way to override it if they look out the window and conclude that the plane is in fact, not stalling.

References

[1] D. Gates, *Flawed analysis, failed oversight: How Boeing, FAA certified the suspect 737 MAX flight control system*, Mar. 2019. [Online]. Available: `https://www.seattletimes.com/business/boeing-aerospace/failed-certification-faa-missed-safety-issues-in-the-737-max-system-implicated-in-the-lion-air-crash/`.

[2] *Latest 737 max fault that alarmed test pilots rooted in software*, Jul. 2019. [Online]. Available: `https://www.bloomberg.com/news/articles/2019-07-27/latest-737-max-fault-that-alarmed-test-pilots-rooted-in-software`.

[3] *How the boeing 737 max disaster looks to a software developer*, Apr. 2019. [Online]. Available: https://spectrum.ieee.org/aerospace/aviation/how-the-boeing-737-max-disaster-looks-to-a-software-developer.