CS1632: Software Quality Assurance Spring 2020

Professor Wonsun Ahn

Who Am I?

- Wonsun Ahn
  - First name is pronounced *one-sun*
  - You can just call me Dr. Ahn (rhymes with *naan*)
- PhD in Computer Science, in Compilers and CPU Design
  - University of Illinois at Urbana Champaign
- Industry Experience
  - I've been a software engineer, field engineer, technical lead, manager
  - I've worked at a 70-person startup company
  - I've also worked at IBM Research (thousands of people)
- When I refer to my experiences with software quality assurance, that is the background from which I will be speaking

Structure of the Course

- (20% of grade) Two Midterms
- (75% of grade) Five projects, mostly done in groups of two students
  - Manual Testing and Traceability Matrix
  - Unit Testing
  - Systems Testing a Web Application
  - Performance Testing
  - Comprehensive development & testing of a large application
- (5% of grade) Participation
- Class resources:
  - Courseweb: submission, grading, announcements, lecture discussion board
  - GitHub: syllabus, textbook PDF, lectures, exercises, deliverables
  - Tophat: attendance, (ungraded) online quizzes, exam preparation
  - GradeScope: source code submission, automated grading and feedback

What is Software Quality Assurance? Your thoughts...?

What is Software Quality Assurance? What it's not...

- It's not something you've never done
- It's not optional
- It's not something you do after you built something
  - It's involved in the entire software development lifecycle: requirement development, software design, writing code, integrating code, verification
- It's not finding every bug
  - It's about managing business risk from exposure to bugs
- It's not just testing
  - It's also about creating processes to help correct problems
  - It's also about providing an independent view of the product

Well, then, what is it?

All activities that ensure quality during software development

QA includes....

Unit testing, systems testing, acceptance testing, automated testing, requirements analysis, equivalence classes, white/grey/black box testing, verification, validation, combinatorial testing, performance testing, usability testing, model checking, static analysis, linting, traceability matrices, defect reporting, test planning, TDD, fuzz testing, KPIs, software profiling, resource analysis, usability analysis, regression testing, smoke testing, security analysis, penetration testing....

It’s an entire field of study!

Case Study: Boeing 737 MAX Crashes

Lion Air crash: Boeing 737 plane crashes in sea off Jakarta

29 October 2018 Share

ALEX DAVIES TRANSPORTATION 03.18.2019 02:47 PM

Crashed Ethiopian Air Jet Is Same Model as Lion Air Accident

An Ethiopian Airlines Boeing 737 MAX 8 crashed shortly after takeoff Sunday, evoking comparisons to an Indonesian incident in October.

Boeing & Aerospace | Business | Nation & World | Times Watchdog

Flawed analysis, failed oversight: How Boeing, FAA certified the suspect 737 MAX flight control system

March 17, 2019 at 6:00 am | *Updated March 21, 2019 at 9:46 am*

Case Study: Boeing 737 MAX Crashes --- Background

- How was Boeing 737 MAX different from previous 737 generations?
- First Boeing 737 (737-100) was built in April, 1967
  - Had a low profile to ease loading/unloading of plane
    (They didn't have belt-loading baggage vehicles at that time)
- Boeing 737 MAX was built in December, 2018
  - Reused old design with larger engine for heavier load (to cut costs)
  - Engine did not fit under wing so had to bring it upwards and forwards

Case Study: Boeing 737 MAX Crashes --- Background

- New engine placement on 737 MAX led to worse aerodynamics
- Boeing did not want to retrain pilots (to cut costs)
- Boeing chose instead to write software to emulate an old 737
  - Make it “feel” like pilot was flying an old 737 instead of 737 MAX
  - Called *Maneuvering Characteristics Augmentation System* (MCAS)

- MCAS was the culprit that forced the planes into a nosedive
- MCAS had issues with software quality assurance at multiple levels

Case Study: Boeing 737 MAX Crashes --- Issue #1: Requirements Analysis

- When Boeing was certifying 737 MAX with FAA (Federal Aviation Administration), MCAS functional requirements were still in flux
  - Boeing was in a hurry to compress production schedule
- MCAS was initially meant to operate only in extreme stall situations
- After certification, MCAS operating window was greatly expanded to include normal situations (e.g. the moments after take-off)
- Boeing did not need to get re-certified with FAA because the changes in requirements did not apply to extreme situations
  - Ironically, the change turned a normal situation to an extreme situation (That is, a nosedive situation right after take-off)

Case Study: Boeing 737 MAX Crashes --- Issue #2: Robustness Testing

- MCAS relied on a single sensor to make the critical decision
  - There was an identical sensor of the same type but it wasn't read
  - There were other related sensors but they weren't read
  - This created a single point of failure
- MCAS relied on a single CPU to make decisions
  - Bits in CPU represented status flags for MCAS
  - Bits can be flipped arbitrarily due to cosmic rays
    (Yes, this does happen especially at high-elevation thin atmospheres)
- Rigorous robustness testing would have caught both problems

Case Study: Boeing 737 MAX Crashes --- Issue #3: Defect Reporting

- Boeing were aware of some defects but characterized them as "hazardous" rather than "catastrophic" in its report to the FAA
  - FAA applied less rigorous standards to defect and let is pass (If pilot can recover from defect within 3 seconds, it's okay)
  - Unfortunately, 3 seconds was enough to pull plane into nosedive
- Boeing did not report the defects to the pilots flying the plane
  - In fact, 1600-page manual mentioned MCAS only once in the glossary
  - Rationale: MCAS was supposed to be invisible to the pilot
  - Had pilot known defect in advance, may have taken corrective action

Case Study: Boeing 737 MAX Crashes --- Issue #4: Corporate Culture

- Not to say that there weren't smart people at Boeing
- Some software quality assurance engineers tried to report problems
- But their independence was compromised by upper management
  - Pressured to not report defects even when found
  - Pressured to downgrade seriousness of defect even when catastrophic
- Larger picture: intense competition from Airbus
  - Competition from Airbus (est. 2000) started to eat into profit margins
  - Management has been increasingly focused on cost-cutting since
- Sometimes you must fight with management and other stakeholders