



University of Pittsburgh

CS 1632 Introduction

Instructor Introduction

Course Introduction

Wonsun Ahn

Department of Computer Science

School of Computing and Information





Instructor Introduction



My Technical Background

■ Wonsun Ahn

- First name is pronounced *one-sun* (if you can manage)
- Or you can just call me Dr. Ahn (rhymes with *naan*)

■ PhD in CPU Design and Compilers

- University of Illinois at Urbana Champaign

■ Industry Experience

- Software engineer, field engineer, technical lead, manager
- Bluebird Corporation (70-person startup company)
 - Manufactures industrial hand-held devices from top to bottom
 - Me: Built software stack based on Windows Embedded
- IBM Research (thousands of people)
 - Does next-gen stuff like carbon nanotubes, quantum computers
 - Me: Designed supercomputers for ease of parallel programming



My World View

- Everything is connected
 - Pandemic: If my neighbors catch the virus, so will I
 - Environment: If my neighbors pollute, I will feel the effects
 - Economy: Think of how the subprime mortgage crisis spread
- Zero-sum thinking (old way of thinking)
 - “If you get a larger slice of the pie, I get a smaller slice.”
 - Therefore, if you lose, I win (and vice versa)
- Zero-sum thinking no longer works
 - If you catch the virus, do I become safer from the virus?
- Collaboration is replacing competition



Collaboration is Replacing Competition

- Is happening in all spheres of life
- Collaboration is also happening in the IT industry
 - The *open source* movement
 - Increasing importance of the software/hardware *ecosystem*
 - Increasing importance of the developer *community*
- Collaboration is also important for learning
 - During my undergrad years, what do I remember best?
 - Stuff that my classmates taught me
 - Stuff that I explained to my classmates



Supporting Collaborative Learning

- I *do not* grade on a curve
 - You will not be competing against your classmates
 - You are graded on your own work on an absolute scale
- You will be working with a partner (on GitHub)
- You are a member of the CS 1632 Team
 - I encourage you to be on Teams at most times (I will too)
 - You can install app on both laptop and cell phone
 - You can ask questions on the Team “Posts” tab
 - Either your classmate or your instructor will answer
 - You can also chat with any individual on the Team
 - “Manage Team” item in the “...” Team context menu



Supporting Collaborative Learning

- You will be a member of a Group
 - On Teams, you are part of a chat group of 10 members
 - Members are a good mix of in-person and online students
 - Your instructor is also a member of each Group
 - It is a smaller support group where you can talk more freely

- You can discuss Exercises / TopHat questions
 - You are encouraged to share each other's work and learn
 - You *cannot* view each other's work for the deliverables



Course Introduction



Structure of the Course

- (20% of grade) Two Midterms
- (70% of grade) Five projects
 - Manual Testing and Traceability Matrix
 - Unit Testing
 - Systems Testing a Web Application
 - Performance Testing
 - Comprehensive static & dynamic testing
- (10% of grade) Participation
 - TopHat lecture questions, exercise submissions, Teams participation
- Class resources:
 - Canvas: announcements, Zoom meetings, recorded lectures
 - GitHub: syllabus, textbook PDF, lectures, exercises, deliverables
 - Tophat: online lecture questions
 - GradeScope: exercise / deliverable submission, grading and feedback
 - Microsoft Teams: Online / off-line communication



For More Details

- Please refer to the course info page:
https://github.com/wonsunahn/CS1632_Spring2021/blob/master/course-info.md
- Please follow the course schedule syllabus:
https://github.com/wonsunahn/CS1632_Spring2021/blob/master/syllabus.md

TODO



11

- Submit the TopHat survey questions (due 1/27)
- Submit the Java assessment exercise (due 1/27):
 - https://github.com/wonsunahn/CS1632_Fall2020/tree/master/exercises/0
 - Submit to GradeScope
- Submit the partnership contract (due 2/5):
 - https://github.com/wonsunahn/CS1632_Fall2020/blob/master/partnership_contract.txt
 - Submit to GradeScope



Brainstorming

- Please go to your respective Teams chat groups
 - But stay in the Zoom room and use only chat on Teams
- 1. Meet & Greet (5 mins):
 - Say a fun fact about yourself
- 2. Then answer the following questions (15 mins):
 - What do you think software quality assurance is?
 - What software QA techniques have you used before?
 - What software QA techniques do you wish to learn?
- 3. Afterwards, we will share discussions with class

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 correct problems
 - It's also about providing an independent view of the SW



What it is

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, reliability 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

16

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

🕒 29 October 2018



🔗 Share

ALEX DAVIES

TRANSPORTATION 03.10.2019 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



17

- How was Boeing 737 MAX different from previous 737 models?
- 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



12

- 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 software quality issues 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 certified to operate only in extreme situations
 - More lax robustness rules compared to “always-on” components
- After certification, MCAS operating window was greatly expanded to include normal situations (e.g. the moments after take-off)
 - It makes sense to recertify MCAS with stricter requirements, right?
- Boeing did not need to get re-certified with FAA
 - Due to a loophole in FAA regulations for certain modifications

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 atmospheres)

- Rigorous robustness testing would have caught both problems

Case Study: Boeing 737 MAX Crashes

--- Issue #3: Defect Reporting



21

- 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 it 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



22

- 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