EC4219: Software Engineering

Lecture 0 — Course Overview

Sunbeom So
2024 Spring

Basic Information

- **Instructor:** Sunbeom So (소순범)
- **Position:** Assistant professor in EECS, GIST
- **Expertise:** Program Analysis
- **Email:** `sunbeomso@gist.ac.kr`
- **Office:** EECS C501
- **Office hours:** by appointment

Software and Software Engineering[1]

- Software: is a collection of computer programs and associated documentation.
  - E.g., specification documentation, installation guide, user manual.
  - In this course, we will interchangeably use the terms between program and software.
- Software Engineering: is an *engineering discipline* that is concerned with all aspects of *software production process*, with the goal of cost-effective software development.
  - engineering discipline: theories and methods to solve problems bearing in mind certain constraints (e.g., financial constraints).

[1] "Software Engineering" by Sommerville

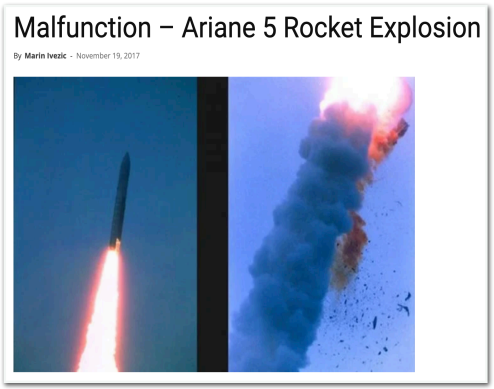
Software (Production) Process[2]

- A structured set of activities required to develop a software system.
- Common software processes include:
  - Specification: defining what the system should do.
  - Design and Implementation: defining the organization of the system and implementing the system.
  - Validation: checking that the system meets what the customer wants.
  - Evolution: changing the system in response to changing customer needs.
- In this course, **we will focus on studying the validation part for ensuring software safety, correctness, and quality.**
  - Why? The validation (SW testing and debugging) is the key; it comprises over 50% of the cost of software development.
  - Why should we invest so much in validation? Why is it so important? SW is written by humans, and humans make mistakes.

[2] "Software Engineering" by Sommerville

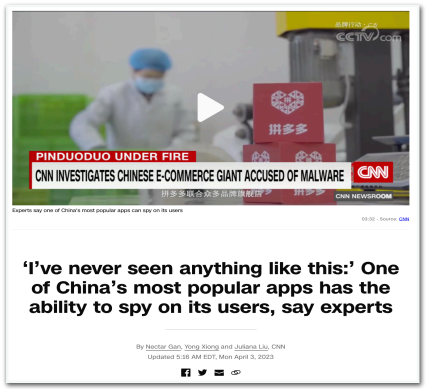
Motivation: Software Errors are Everywhere

As we get into "software-centric society", SW safety problems due to SW bugs have become increasingly prevalent.

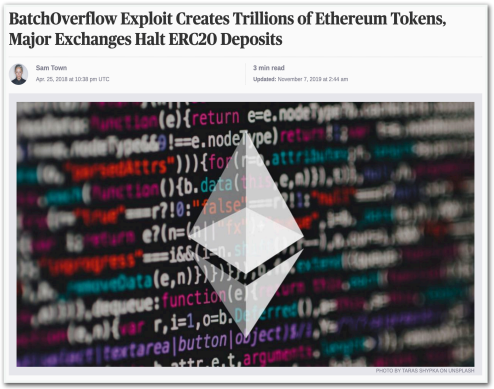


Rocket SW

Aircraft SW



Mobile App



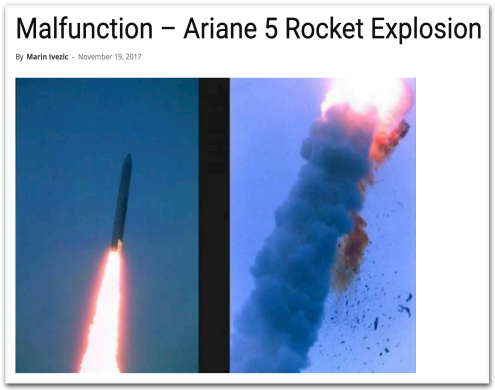
Blockchain SW

Medical SW

Self-driving SW

The Worst SW Disaster in History

- The Ariane 5 Rocket Explosion (1996)



- Cost: 10 years and $7 billion – gone 37 seconds after its launch
- The catastrophe was caused due to the integer overflow error.
  - Attempted to fit 64-bit format data into 16-bit space (during takeoff to convert one piece of data – the sideways velocity of the rocket)
  - The number was too big to fit and resulted in an overflow.
  - This error was misinterpreted by the rocket's onboard computer as a signal to change the course of the rocket.

We will learn fundamental, modern technologies to prevent SW disasters in advance.

Program Analysis

Program analysis is the process of automatically discovering useful facts about programs. Examples include:

- Verification: is this program correct with respect to specifications?
- Bug-finding: does this program have integer-overflow bugs?
- Equivalence: are two programs semantically equivalent?
- Compiler optimization: Does the optimized program preserve the semantics of the original one?
- many others

Taxonomy of Program Analysis

Program analysis techniques can be broadly classified into three kinds.

- Dynamic analysis: the class of run-time analyses. These analyses discover information by running the program and observing its behavior.
- Static analysis: the class of compile-time analyses. These analyses discover information by inspecting the source code or binary code.
- Hybrid analysis: combines aspects of both dynamic and static analyses, by incorporating runtime and compile-time information in certain ways.

Examples of Dynamic Analysis Tools

Dynamic analyzers infer facts of the program by monitoring its runs.

- Purify: a tool for checking memory accesses, such as array bounds, in C and C++ programs.
- Valgrind: a tool for detecting memory leaks in x86 binary programs.
- Eraser: a tool for detecting data races in concurrent programs.
- Daikon: a tool for finding likely *invariants*. An invariant is a program fact that is true in every run of the program.

Examples of Static Analysis Tools

Static analyzers infer facts of the program by inspecting its source (or binary) code.

- Infer (Facebook): a tool for detecting memory leaks in Android applications.
- SLAM (Microsoft): a tool for whether C programs respect API usage rules. This tool is used by Windows developers to check whether device drivers use the API of the Windows kernel correctly.
- VeriSmart: an analyzer for finding security vulnerabilities in smart contracts.

We will learn fundamental theories behind these tools.

Course Materials & References

- Self-contained slides will be provided.
- Most slides will be based on the following materials from Software Analysis course at UPenn.

  https://software-analysis-class.org

  - Very useful materials for self-study; provides lecture videos and slides with text notes.

Welcome to Software Analysis!

In this course, we will study the theory and practice of software analysis, which lies at the heart of many software development processes such as diagnosing bugs, testing, debugging, and more.

What this class won't do is teach you basic concepts of programming. Instead, through a mix of basic and advanced exercises and examples, you will learn techniques and tools to enhance your existing programming skills and build better software.

Prerequisites & Notes

- **Prerequisites:** To succeed in this course, you should:
  - have completed of the basic CS courses (e.g., Automata Theory, Data Structures, Algorithms).
  - have extensive experience in computer programming.
- **Notes:** This is an *advanced* course. Complaints like below are hard to be reflected (sorry!).
  - "Teach me more about the syntax of OCaml language."[3]
  - "Homeworks are too hard."

[3] I will provide basic guidance, but will not explain every single detail from one to ten.

Grading Policy

- Mid-term/Final exam (100 points for each).
- 3 programming assignments in OCaml.[4]
- Your final grade will be determined as follows.
  1. Determine the letter grades based on the exam results (F – B+).
  2. If you have completed[5] 2 of 3 programming assignments, your grade will be increased by one level (e.g., B+ $\rightarrow$ A, B $\rightarrow$ B+).
     If you have completed all the assignments, your grade will be increased by two levels (e.g., B+ $\rightarrow$ A+, B $\rightarrow$ A).
- If you just need a B+ or a lower grade under this grading system, it is perfectly fine not to conduct any assignments.
- You must attend more than 2/3 of the classes according to GIST regulations (see No.25 of ER-322-07).
  - As long as you conform to this rule, attendance will not affect your letter grade. So, you do not need to send me an email before your absence.

[4] `https://ocaml.org/`

[5] passed 90% of testcases provided by the instructor (hidden at the submission time)

Academic Integrity

Read Carefully.

- All assignments (i.e., writing code) must be your own work.
  - You should not share/show your code.
  - You should not post your code on public websites (e.g., GitHub repository).
  - You should not modify other students' code.
- Cheating on assignments gets you an F.
- I will have a one-on-one talk with each suspicious student.
- If you fail to prove your integrity in the one-on-one talk,[6] I will regard that you cheated on your assignments even if you do not admit your cheating.
- Your grade is not negotiable. Do not send me an email for it after the grade notification.

[6] e.g., failing to answer my questions about the details of your submissions

Summary

- Software engineering is an engineering discipline concerned with all aspects of software production process.
- In the software process, validation is highly important due to the prevalence of SW errors.
- Program analyses are fundamental and modern technologies that can effectively help the SW validation process.
- Next class: more detailed introduction to program analysis