AAA616: Program Analysis

Lecture 0 — Course Overview

Hakjoo Oh
2024 Fall

Basic Information

Instructor: Hakjoo Oh

- **Position:** Professor in CS, Korea University
- **Expertise:** Programming Languages, Software Engineering
- **Office:** 616c, Science Library
- **Email:** `hakjoo_oh@korea.ac.kr`
- **Office Hours:** 1:00pm–3:00pm Mondays (by appointment)

Course Website:

- `https://prl.korea.ac.kr/courses/aaa616/2024/`
- Course materials will be available here.

Unsafe Software

- SW bugs are everywhere

Knight Capital Says Trading Glitch Cost It $440 Million

BY NATHANIEL POPPER AUGUST 2, 2012 9:07 AM 356

Runaway Trades Spread Turmoil Across Wall St.

Finance

Tesla in fatal California crash was on Autopilot

31 March 2018

Self-Driving Cars

Up to 270 women may have died after breast cancer screening IT error

Health secretary says computer errors led to 450,000 missed screenings in England since 2009

Have you been affected?

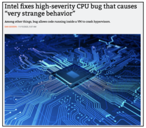
Healthcare

$25 million in BTC, ETH stolen in dForce hacking

April 22, 2020

The decentralized finance (DeFi) community has been

Blockchain

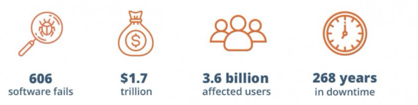

Intel fixes high-severity CPU bug that causes "very strange behavior"

Among other things, bug allows code running inside a VM to crash hypervisors.

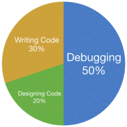
DAN GOODIN - 11/15/2023, 5:57 AM

Chip Design

- Enormous costs due to SW bugs



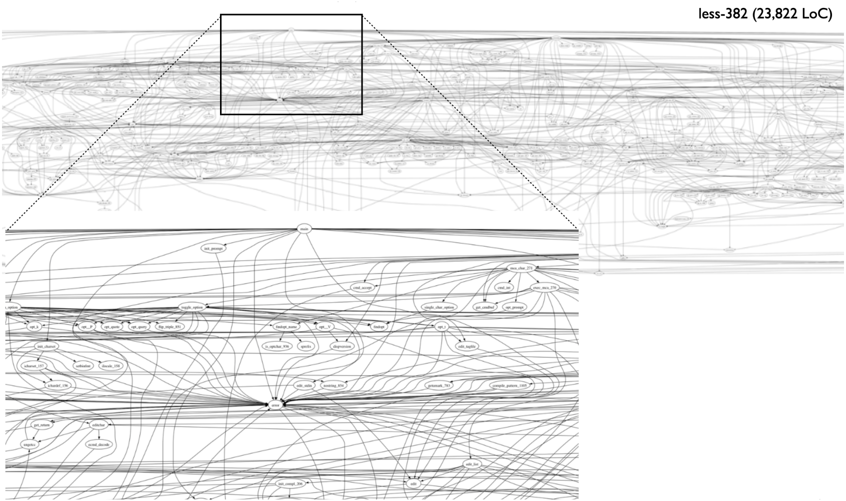
Software fail watch (5th edition). 2017



Software development cost

SW Complexity

Software is inherently complex and difficult to write, debug, and fix.



Towards Safe Software Technology

- The technology for efficient software is mature.

$$\text{Program} \rightarrow \boxed{\text{Interpreter}} \rightarrow \text{Result}$$

- However, technology for safe software is not. Current language systems put almost all the burden of writing safe programs on the programmers. This manual approach to safe software has proven extremely unsuccessful.
- Automated technology for analyzing the safety of programs:

$$\text{Program} \rightarrow \boxed{\text{Analyzer}} \rightarrow \boxed{\text{Interpreter}} \rightarrow \text{Result}$$

Static Program Analysis

- Technology for predicting SW behavior statically and automatically
  - static: before execution, before sell / embed
  - automatic: sw is analyzed by sw ("static analyzer")
- Applications
  - bug-finding: e.g., find runtime failures of programs
  - security: e.g., is this app malicious or benign?
  - verification: e.g., does the program meet its specification?
  - compiler optimization: e.g., automatic parallelization
  - program synthesis, automatic patch generation, etc

Topics

In this course, we will focus on foundational topics on program analysis:

- Programming language theories
- Abstract interpretation framework

Weeks	Topics
Week 1	Introduction
Week 2	Static Analysis Concepts
Week 3	Operational Semantics
Week 4	Denotational Semantics
Week 5	Axiomatic Semantics
Week 6	Abstract Interpretation
Week 7	Advanced Topics
Week 8	Final Exam

Prerequisites:

- Undergraduate-level programming languages, compilers, theory of computation, and discrete math

Course Materials

- Lecture slides.
- Xavier Rival and Kwangkeun Yi. Introduction to Static Analysis: An Abstract Interpretation Perspective. MIT Press
- Flemming Nielson, Hanne Riis Nielson, Chris Hankin. Principles of Program Analysis. Springer
- Others

Grading

- Quiz / Participation –50%
  - 3–4 Quizzes on random days.
- Final Exam – 50%