AAA616: Program Analysis

Lecture 0 — Course Overview

Hakjoo Oh
2026 Fall

Basic Information

Instructor: Hakjoo Oh

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

Course Website:

- `https://prl.korea.ac.kr/courses/aaa616/2026/`
- Course materials will be available here.

Unsafe Software

- SW bugs are everywhere

Finance

Self-Driving Cars

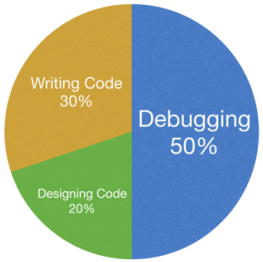
Healthcare

Blockchain

Chip Design

- Enormous costs due to SW bugs

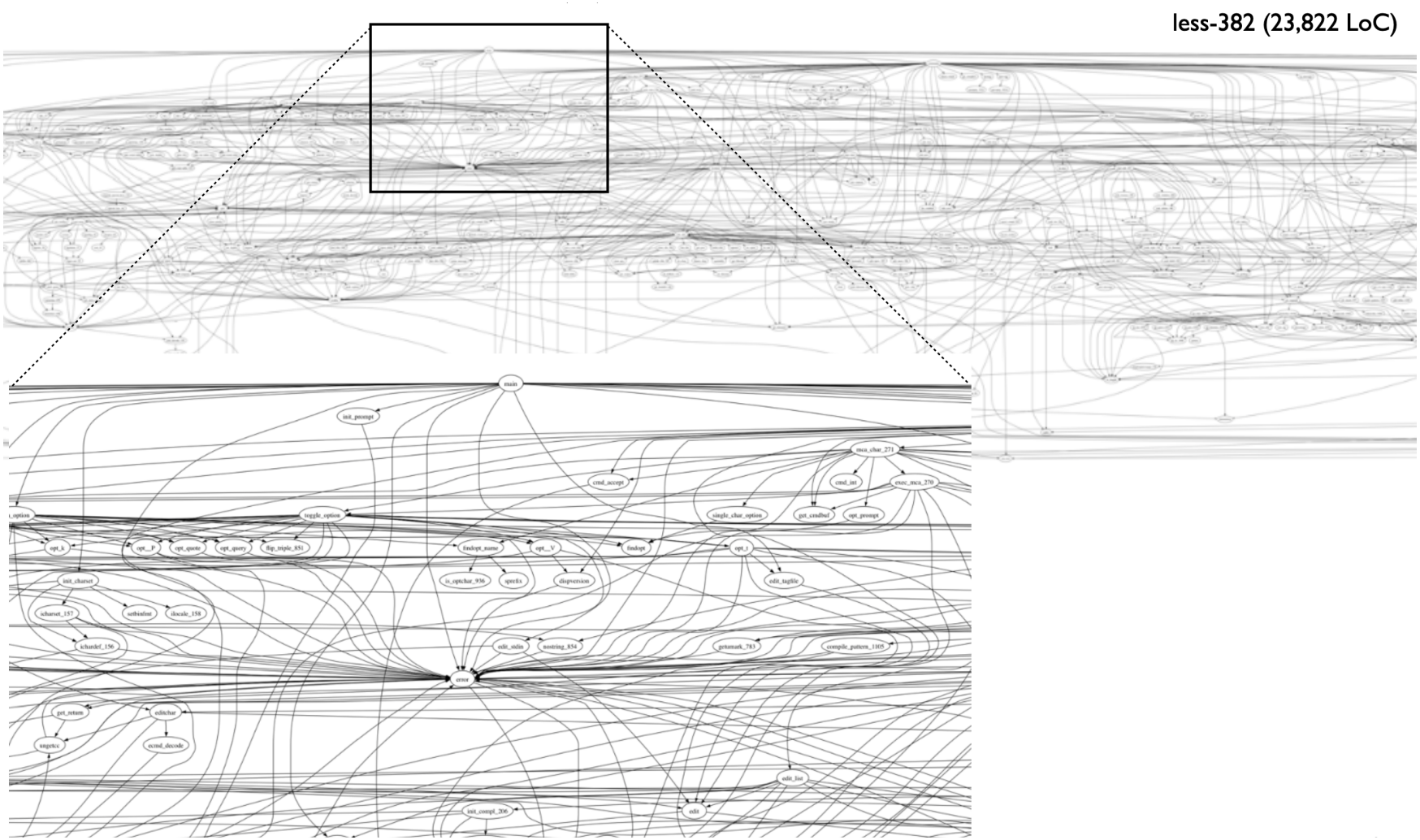
Software fail watch (5th edition). 2017



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.

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

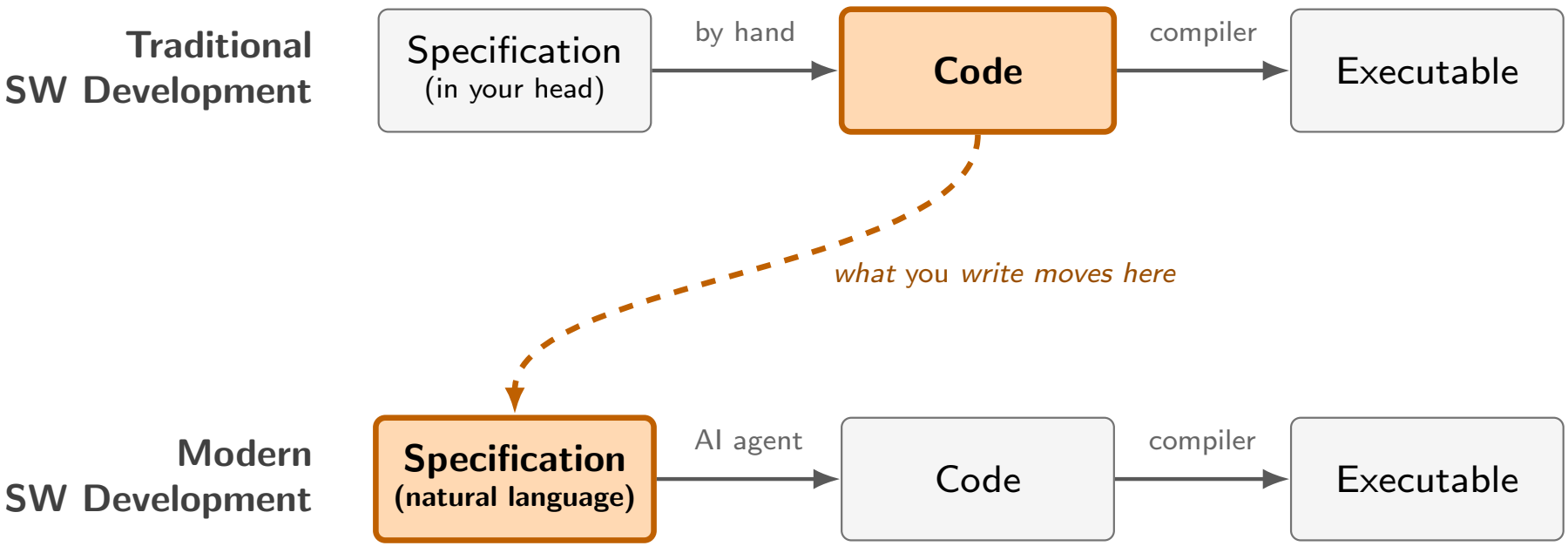
- However, technology for safe software is not. Current language systems put almost all the burden of writing safe programs on the programmers. This manual approach to safe software has proven extremely unsuccessful.
- Automated technology for analyzing the safety of programs:

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

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

SW Development in the Age of AI Agents



Topics

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

- Programming language theories
- Abstract interpretation framework
- Program verification
- Dynamic approaches

Weeks	Topics
Week 1	Introduction
Week 2	Program Analysis Concepts
Week 3	Static Analysis Overview
Week 4	Formal Semantics
Week 5	Abstract Interpretation
Week 6	Program Verification
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 (Tentative)

- Assignment / quiz –50%
  - 3–5 in class (online)
- Final Exam – 50%