CPEN 321

Recap + Quiz 2 Prep

Last Two Weeks

1. Anthony Chu, Microsoft:
   Development of Containerized Applications
2. Philippe Kruchten: Maintenance and Technical Debt
3. Zeyad Tammi and Harsha Kadiyala:
   CI, DevOps, Deployment
4. Yingying Wang: Team Work and Version Control

Overall Summary – What We Covered

- Software development processes
- Requirements, use-cases
- Architecture and design, architectural design patterns
- Microservices, REST
- Code review, code smells
- Validation and verification, testing and analysis
- Technical debt, software maintenance
- Development of containerized applications
- DevOps, Continuous Integration
- Team work, version control

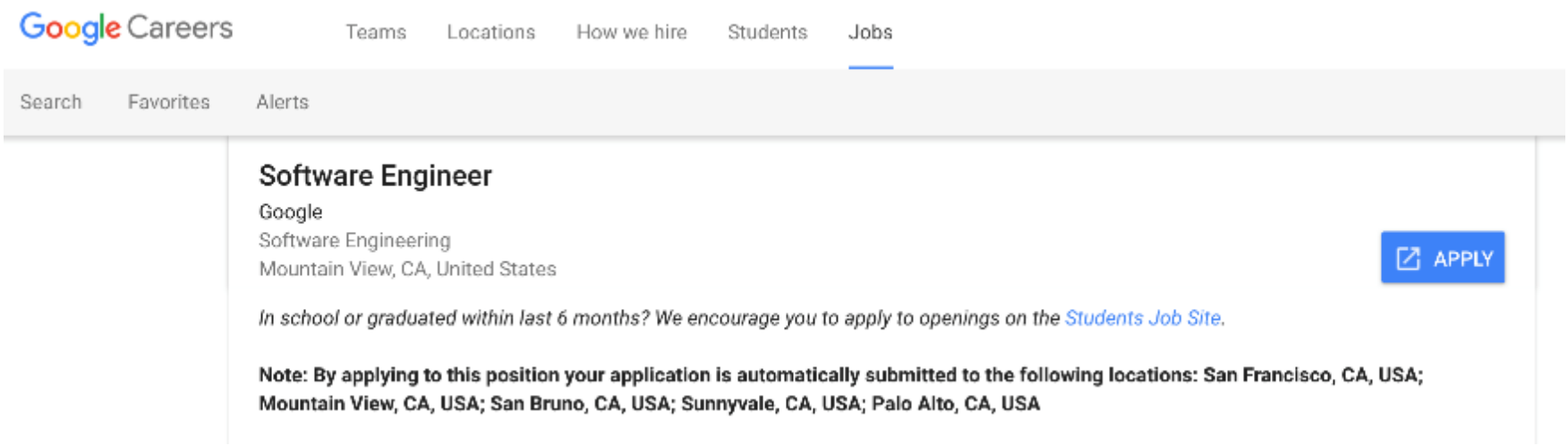
- Working with people, making decisions, meeting deadlines

Agenda for Today

- Summary
- SE Research
- Next milestones
- Mid-term 1 results
- Mid-term 2 expectations

Software Engineering – what is it all about?

https://www.levels.fyi/?compare=Facebook,Google,Microsoft&track=Software%20Engineer



Responsibilities

- Design, develop, test, deploy, maintain and improve software.
- Manage individual project priorities, deadlines and deliverables.

Minimum qualifications

- BS degree in Computer Science, similar technical field of study, or equivalent practical experience.
- Software development experience in one or more general purpose programming languages.
- Experience working with two or more from the following: web application development, Unix/Linux environments, mobile application development, distributed and parallel systems, machine learning, information retrieval, natural language processing, networking, developing large software systems, and/or security software development.
- Working proficiency and communication skills in verbal and written English.

Preferred qualifications

- Master's, PhD degree, further education or experience in engineering, computer science or other technical related field.
- Experience with one or more general purpose programming languages including but not limited to: Java, C/C++, C#, Objective C, Python, JavaScript, or Go.
- Experience developing accessible technologies.
- Interest and ability to learn other coding languages as needed.

Software Engineer Skill Set

- Technical programming skill and experience
- Can manage time and tasks
- Self-starter and able to work independently
- Can manage uncertainty and take on hard (often vaguely defined) tasks
- Critical thinker, has good problem analysis and problem-solving skills
- Quick learner
- Persistent and able to finish things
- A good team player
- Efficient communicator: writing, public speaking

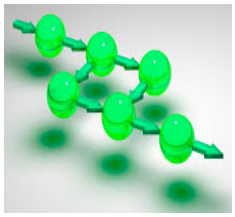
Prof. Rubin Research: Reliable, Secure and Sustainable Software Lab

Microservice-based distributed systems

- Architecture and change management
- Performance and energy efficiency
- Microservice-aware scheduling

Security, privacy, and energy efficiency in mobile

- Analysis and patching of existing application
- (Security) Testing
- New OS mechanisms



Feature management and reuse in version control systems

- Feature extraction and integration
- Change management and fault localization

Agenda for Today

- Summary
- SE Research
- **Next milestones**
- Mid-term 1 results
- Mid-term 2 expectations

Future Milestones (Updated)

- W4: Development team and the customer discuss the requirements.
- W5: M1 – Requirements (both customer and development teams).
- W6: M2 – Design (development team).
- W8: M3 – MVP (development team).
- W9: M4 – Code review (development teams).
- W10: M5 – Test plan (development team).
- W11: M6 – Refined specifications (development team).
- W12: M7 – Test results + customer acceptance testing (both customer and development teams).

W12, M7 (Nov 19) – Test Results + Customer Acceptance Testing

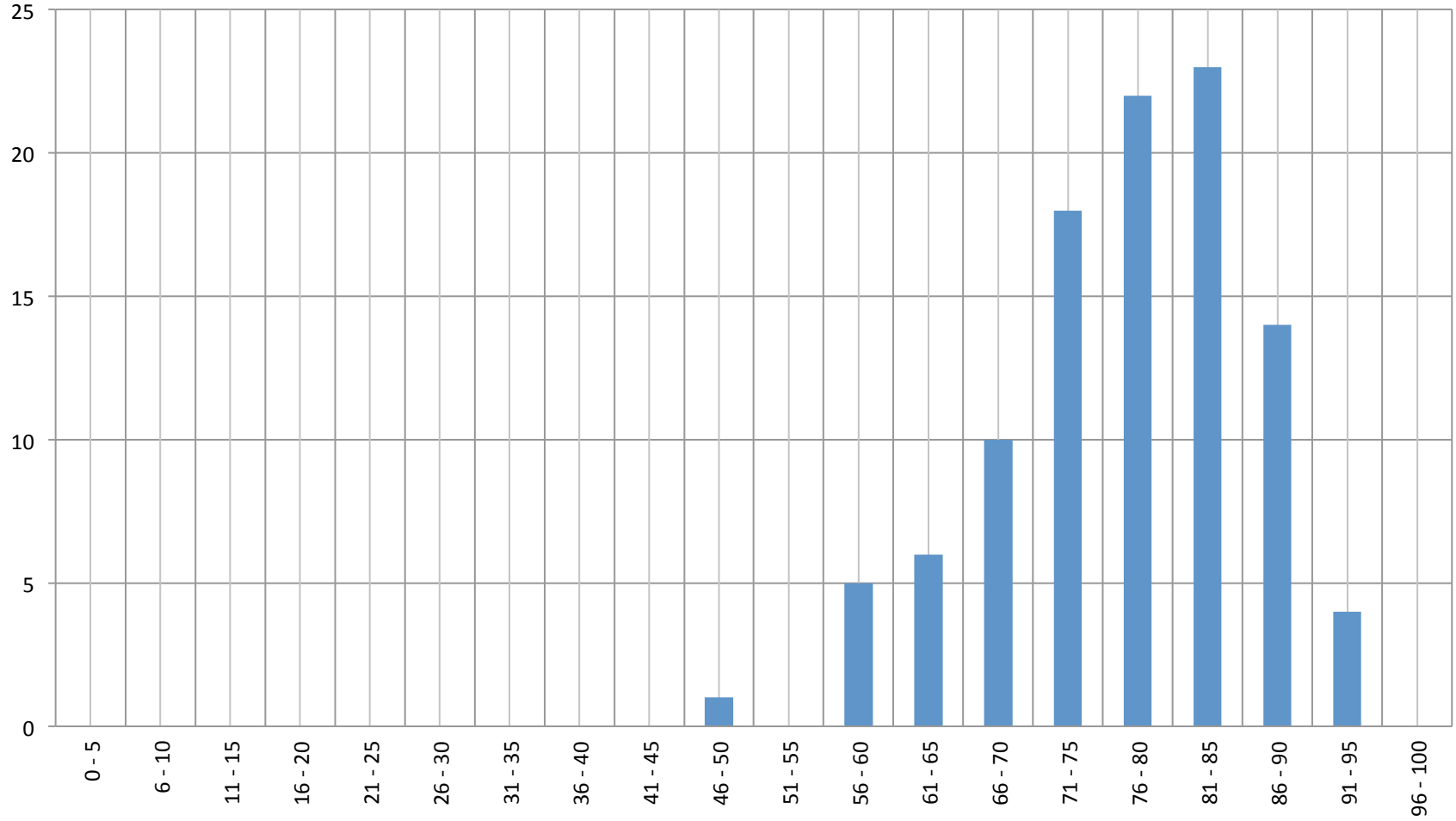
- Development team
  - Statistics + explanations about the number of test you created, per category (unit tests, system-level test, GUI tests)
  - A log of your automated test execution + status of the tests
  - A report on code coverage
- Customer team
  - A report on 1 buggy execution scenario that you found via an ad-hoc GUI testing
  - The report should focus on a major fault
  - It should contain the execution scenario (sequence of events, with screenshots) and the description of the fault

W13, M8 (Nov 26) – Final project presentations

- Presentations to instructors: in class on Monday and during lab hours
- Top 5 projects presented in class on Wednesday
- Awards!

(More details next week)

Mid-Term 1 Grade Distribution



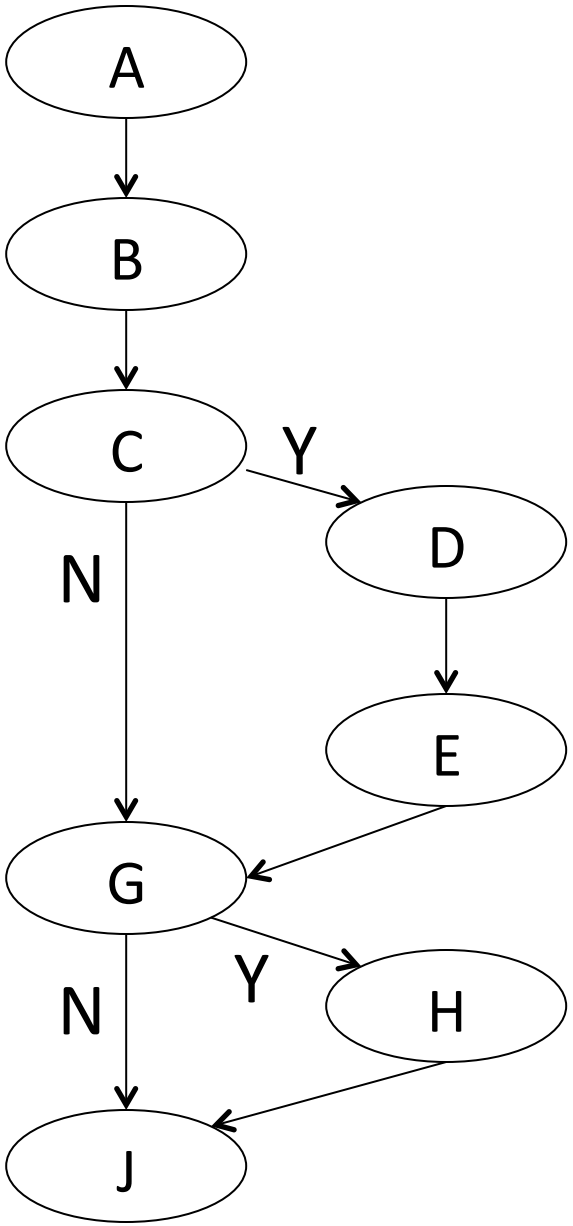
Mid-Term 2

- Software development processes
- Requirements, use-cases
- Architecture and design, architectural design patterns
- Microservices, REST
- **Code review, code smells**
- **Validation and verification, testing and analysis**
- **Technical debt, software maintenance**
- **Development of containerized applications**
- **DevOps, Continuous Integration**
- **Team work, version control**

Exercise

```
A:    void f(int x) {
B:        int y = x;
C:        if (x ≥ 10) {
D:            x = x – 10;
E:            y++;
F:        }
G:        if (x ≥ 5) {
H:            x++;
I:        }
J:        print(x,y);
K:    }
```

1. Draw the Control Flow Diagram (CFG) for this function.




```
A:    void f(int x) {
B:        int y = x;
C:        if (x ≥ 10) {
D:            x = x – 10;
E:            y++;
F:        }
G:        if (x ≥ 5) {
H:            x++;
I:        }
J:        print(x,y);
K:    }
```

Exercise

```
A:    void f(int x) {
B:        int y = x;
C:        if (x ≥ 10) {
D:            x = x – 10;
E:            y++;
F:        }
G:        if (x ≥ 5) {
H:            x++;
I:        }
J:        print(x,y);
K:    }
```

2. How many feasible paths does this program have?

Exercise

```
A:   void f(int x) {
B:       int y = x;
C:       if (x ≥ 10) {
D:           x = x – 10;
E:           y++;
F:       }
G:       if (x ≥ 5) {
H:           x++;
I:       }
J:       print(x,y);
K:   }
```

3. Design a **minimal** test suite that has 100% coverage for each of the criteria specified below:

(a) statement coverage

(b) branch coverage

(c) path coverage

```
A:   void f(int x) {
B:       int y = x;
C:       if (x ≥ 10) {
D:           x = x – 10;
E:           y++;
F:       }
G:       if (x ≥ 5) {
H:           x++;
I:       }
J:     print(x,y);
K:   }
```

(a) statement coverage

- x=50

(b) branch coverage

- x=50
- x=0

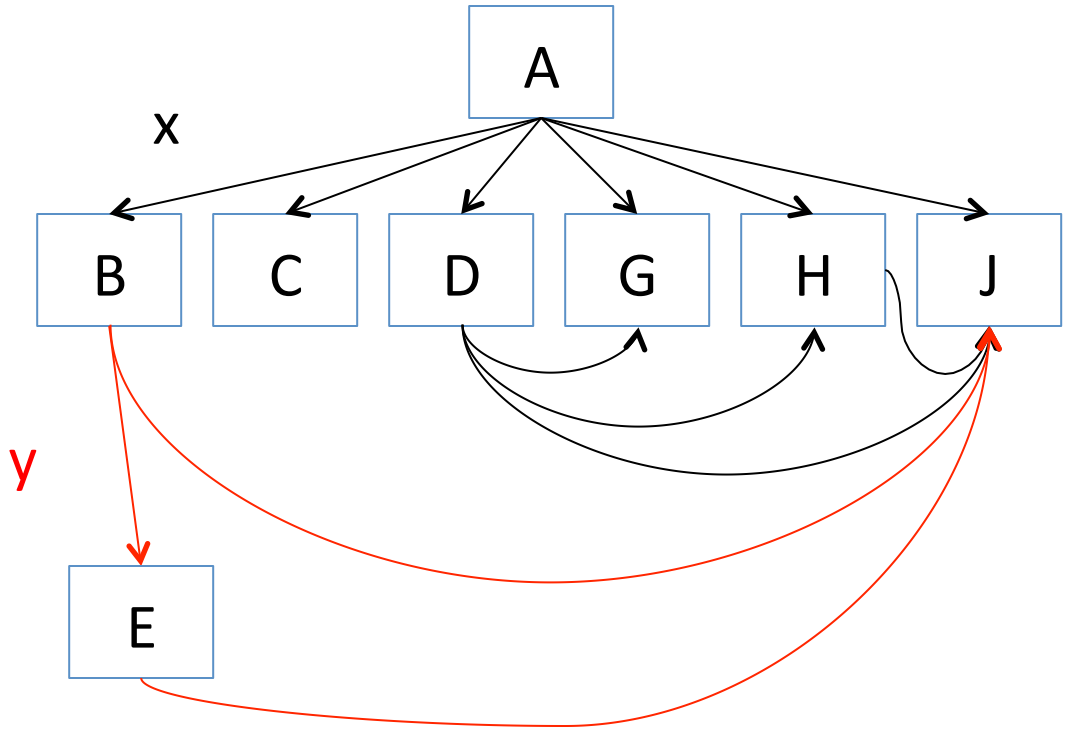
(c) path coverage

- x=50
- x=11
- x=6
- x=0

Exercise

```
A:    void f(int x) {
B:        int y = x;
C:        if (x ≥ 10) {
D:            x = x – 10;
E:            y++;
F:        }
G:        if (x ≥ 5) {
H:            x++;
I:        }
J:        print(x,y);
K:    }
```

4. Draw the data flow graph for this function. Do not include any control flow links, just the data flow links.




```
A:   void f(int x) {
B:       int y = x;
C:       if (x ≥ 10) {
D:           x = x – 10;
E:           y++;
F:       }
G:       if (x ≥ 5) {
H:           x++;
I:       }
J:     print(x,y);
K:   }
```