CS510 Software Engineering

Introduction

Asst. Prof. Mathias Payer

Department of Computer Science
Purdue University

TA: Scott A. Carr
Slides inspired by Xiangyu Zhang

http://nebelwelt.net/teaching/15-CS510-SE

Spring 2015

The Nature of Software

- Easy to reproduce (copy), but *development* is hard, in other fields manufacturing is hard.
- Development is hard to automate, a human task.
- Intangible, development effort hard to understand.
- Huge demand to scale up.
- Quality hard to measure: scripting vs programming.
- Easy to modify (if source is available), but hard to understand complex system.
- Detoriation through design changes (erroneously or unanticipated).

The Nature of Software: Results

- Most software has poor design and is getting worse.
- Demand for software is high and rising.
- We are in a perpetual *software crisis*.
- We have to learn to *engineer* software.

Fundamental Facts about Software Engineering

by Robert L. Glass

`http://www.computer.org/portal/web/buildyourcareer/fa035`

Complexity: problem ↑ results in 10∗ software ↑.

People: best programmers are up to 30∗ more efficient.

Tools/techniques only allow 5-30% improvement, pays off after initial drop.

Quality includes portability, reliability, efficiency, human engineering, testability, understandability, and modifyability.

Reliability: error detection and removal results in 40% of development cost. Software will always contain residual effects, goal is to minimize.

Efficiency relies on good design, high-order language code can be as efficient as low-level code.

The Truth About Software Projects

Standish Group data of 9,236 projects completed in 2004:

- 29% completed successfully.
- 53% late, over budget, and/or with missing features.
- 18% cancelled.

Out of the incomplete/failed projects many did not meet the original definitions, there were software defects, and/or the delivery date slipped several times.

What is SE – historically

1968 NATO conference in Garmisch, Germany. Goal: solve the software crisis. Software is delivered...

- late
- over budget
- with residual faults
- with missing features

The Software Crisis

- The Software Crisis has not been solved!
- Programmers not arbitrarily scalable.
- Clear software design essential.

Definition of Software Engineering

Software Engineering is a discipline whose aims are:

Dependability: producing fault-free software.

Productivity: deliver on time, within budget.

Usability: satisfy a client's needs.

Maintainability: extensible when needs change.

Combines aspects of computer science (PL, networking, OS, databases, and many more), project management, economics, and many more.

Contents for a Graduate Level SE Course

- Testing?
- Textbooks?
- Requirement analysis, modeling?
- Agile programming?
- Results are inconclusive.

Aspects of Software Engineering

- Religion: waterfall vs XP, design patterns, architectures...
- Analytic: machine tractable, provable, model checking, program analysis, verification, testing...

CS510 contents

Algorithmic software engineering with hands on experience:

- Program analysis, model checking, software verification, testing.
- Focus on technical part first, allows you to use techniques in your projects.

Bible part is not covered: project management, design patterns, requirement analysis, software metrics, or UML[1].

[1] Look for books and online articles on these topics

CS510 contents (2)

- This may not be the course you expected.
- Many new concepts, lots of coding, x86 machine code, large analysis infrastructure.
- Get familiar with state of the art automated techniques.
- Reuse and leverage them in your own research.

Grading policy, projects, exams, and homework

- 3 small projects in first 2 months (30% of grade),
- One term project (20% of grade),
- Homework (10% of grade),
- Midterm (15% of grade),
- Final (25% of grade) (qualifier in final),
- Possible challenges for extra credits.

Submitting homework and projects

Class teaches formal aspects of software engineering, projects and homework allow practical experience:

- Use a source repository to check in solutions,
- Organize your projects and homeworks according to a design document,
- Peer review and comment the code of other students,
- Work with a large code base, develop extensions.

Most of the homework and the 3 small projects will be due in the first two months of the semester, allowing the larger project a little more than a month.

Class projects

- Use ASan to find memory errors in an existing program
- Use LLVM to construct call graph and track memory allocation sites
- Use Klee to find vulnerabilities in existing programs
- Develop an LLVM data-flow tracking pass.
- Bonus: compare GProf and FastBT basic block tracing.

Recommended Books

- Computer Systems: A Programmer's Perspective (2nd Edition); by Randal E. Bryant.
- Computer Architecture, A Quantitative Approach (5th Edition); by John L. Hennessy.
- Compilers: Principles, Techniques, and Tools (2nd Edition, the dragon book); by Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman.

Academic Integrity

All work that you submit in this course must be your own. Unauthorized group efforts are considered *academic dishonesty*.

You are allowed to discuss the problem with your peers but you may not copy or reuse any part of an existing solution or work in a team.

We will use automatic tools to compare your solution to those of other current and past students. *The risk of getting caught is too high!*

Course Organization

- Homepage:
  `http://www.nebelwelt.net/teaching/15-510-SE`
- Lecture: Tuesday and Thursday 4:30p to 5:45p in REC 313.
- Office hours: Tuesday 3:30p to 4:30p in LWSN 3154M.
- TA hours: Thursday 3:30p to 4:30p in LWSN 3133, desk 11.
- Homework/projects are due the day of the lecture, before the lecture.

Questions?

?