Development Models: Extreme Programming

SimSE:
Software Development Simulation Game

Plan for today

- What's coming up next week
- Brief wrap up of 521/621 material
- Course evaluations
- SimSE: Software Development Simulation Game

Coming up

- Final project reports due:
  Friday, Dec 7, 11:59 PM EST
- Final project presentations:
  Tuesday, Dec 4 and Thursday, Dec 6, in class
  presentation order picked at random, on Dec 4

Questions?

521/621 Advanced Software Engineering Analysis and Evaluation

What have we learned?

Software Engineering before this class

You knew how to build software systems

- design
- specify
- develop
- document
- test
- maintain

Software Engineering after this class

Now you know
- how to reason about software
- what can be done automatically
- what can be proven
- what cannot be proven

- Let's consider some highlights

Dynamic Analysis techniques

- Automatic property inference
  - Daikon: run tests, extract properties over data values
- Speculative analysis
  - Crystal: learn about conflicts as soon as they happen
  - Quick Fix Scout: learn about effects of menus

What's hard about specification?

- User communication
  - Most common cause of project failure: not involving the users
- Getting on the same page
  - Without a careful approach, even designing small systems in small teams quickly leads to ambiguities and misunderstandings
- User interface can make or break a system
  - People won't use your app
  - Luggage gets lost
  - People can die

Static Analysis

- Automatic test generation
  - Can use documentation (pre- and post-conditions) to generate tests automatically
  - Or combine with Daikon to infer pre- and post-conditions
- Formally verify system correctness
  - FLAVERS: prove state reachability, safety properties

Can we compute anything?

- Undecidability
  - Most problems cannot have a program written to solve them
  - This bounds the power of static analysis: can't prove simple things (e.g., if a line can ever execute) in general
- There are
  - As many rational numbers as integers
  - Many, many more irrational numbers than integers

Debugging

- When and how to debug
  - Make errors impossible by design
  - Think before you code: make code right
  - Make errors immediately visible: don't hide
  - Last resort: form a hypothesis, test it, trace through code, find bug
- Performance
  - In some domains, as important as correctness
  - Path tracing leads in accurate runtime complexity measures

Automatic Debugging

- Minimize cause of failure: Delta Debugging
  - Find the smallest input that causes a test to fail
  - Undo a small subset of latest changes
- Automatically remove errors: Genprog
  - Use tests to guide search through program space
  - Automatically generate a patch
  - Works for small fixes, but…

Can it work for generating programs from scratch?

What I hope you walk away with

- Research skills: pushing the state-of-the-art
- How to use the latest ideas in software engineering in your work
- Where to find a solution to your software engineering problems

Evaluations

Development Models: Extreme Programming

SimSE:
Software Development Simulation Game

SimSE: Software Development Simulation Game

http://people.cs.umass.edu/~brun/class/CS521.621/SimSE