Lecture 22

Path Spectra
Change Impact Analysis

EE 382V Spring 2009 Software Evolution - Instructor Miryung Kim

Today’s Agenda (1)

- Recap of RTS
- Brief Discussion on Program Profiling
- Class activity on statement, branch and path coverage
- Presentation
  - Reza (advocate)
  - Xin (skeptic)

Today's Agenda (2)

- Chianti change impact analysis framework
  - First phase: affected test identification
  - Second phase: isolation of failure-inducing deltas

Recap of RTS (1)

- Software evolution may introduce regression faults.
- Regression testing intends to check preservation of desirable program behavior and to prevent undesirable program behavior (regression faults) through testing.
- Given a test suite T, two program versions, RTS selects a subset of T that have a potential to reveal regression faults.
- RTS needs three building blocks: (1) program differencing tool, (2) coverage gathering tool, and (3) test selection algorithm.

Recap of RTS (2)

- Regression testing is an exciting research area with practical impact on software evolution.
  - Test Selection
  - Test Prioritization
  - Test Minimization
  - Test Generation & Augmentation

Path Spectra [Reps et. al.1997]

- The use of program profiling for software maintenance with applications to the Y2K problem
- ESEC/FSE 1997

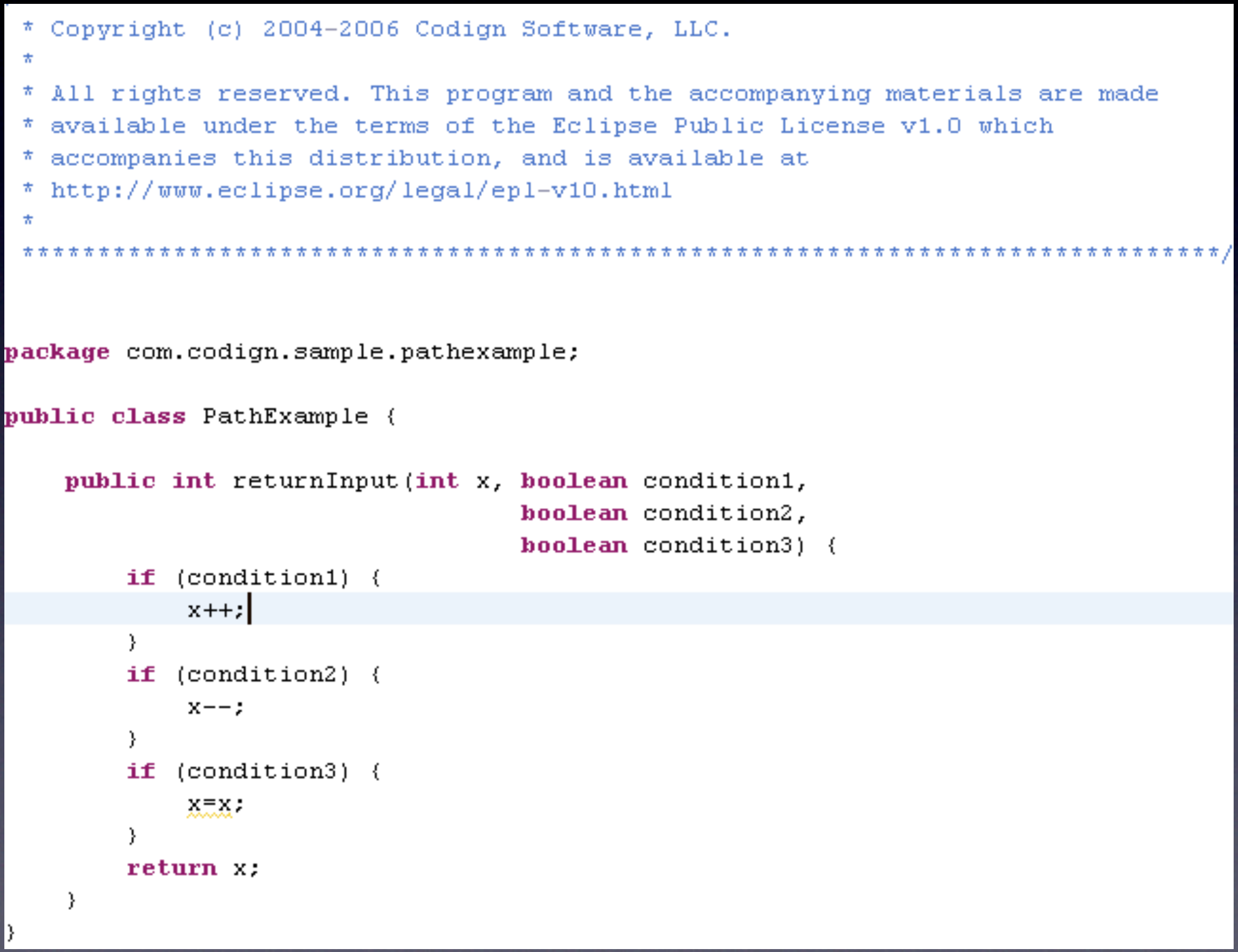
What is Program Profiling?

- Recording behavior of a program during execution
- What can you record about a program's execution behavior?
  - covered methods/ exercised methods
  - sequence / ordering of exercised methods (program elements)
  - running time
  - branch coverage, path coverage
  - memory usages - heap object allocation, etc
  - number of threads / thread schedule
  -

Program Profiling

- Memory usage; e.g., heap size over time. # of times a garbage collector was called.
- The depth of a stack, etc.
- Coverage
  - Function coverage: Has each function been executed?
  - Statement coverage: Has each statement been executed?
  - Branch coverage: Has each control structure evaluated both true and false?
  - Path coverage: Has every possible route been executed?

Class Activity:
Branch and Path Coverage



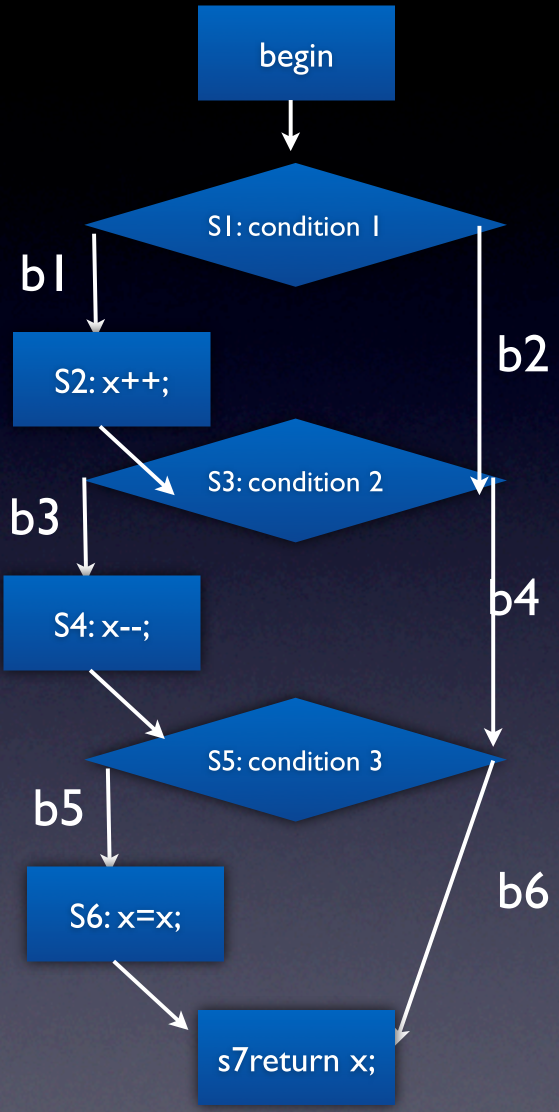
```
 * Copyright (c) 2004-2006 Codign Software, LLC.
 *
 * All rights reserved. This program and the accompanying materials are made
 * available under the terms of the Eclipse Public License v1.0 which
 * accompanies this distribution, and is available at
 * http://www.eclipse.org/legal/epl-v10.html
 *
 ********************************************************************************/


package com.codign.sample.pathexample;

public class PathExample {

    public int returnInput(int x, boolean condition1,
                                  boolean condition2,
                                  boolean condition3) {
        if (condition1) {
            x++;
        }
        if (condition2) {
            x--;
        }
        if (condition3) {
            x=x;
        }
        return x;
    }
}
```

UT EID:
Name:



Fill out the following code coverage table by running the returnInput with the following input

input	covered statements	covered branches	covered paths
(cond1=true, cond2=true, cond3=true)	s1, s2, s3, s4, s5, s6, s7	b1, b3, b5	[b1, b3, b5]
coverage %	100%	50%	12.5%
(cond1=false, cond2=false, cond3=false)	s1,s3, s5,s7	b2, b4, b6	[b2,b4,b6]
coverage %	100%	100%	25%
(cond1=false, cond2=true, cond3=true)	s1,s3,s4,s5,s6, s7	b2, b3, b5	[b2,b3,b5]
coverage %	100%	100%	37.5%

Motivation of Reps et al.

- Y2K problem
  - Would my program have erroneous behavior when run on input year = 2001?
  - => Would my program exercise a different path during program execution in comparison to input year= {1900, 1901, 1902, ... 1999}?
  - => How can we concisely represent path profiles for a set of inputs (in order to do this profile comparison)?

Research Problem addressed by Reps et al.

- Given two different sets of inputs for the same program, how can we reason about path-profile differences (divergences?
- What is an appropriate representation for reasoning about program path profiles for a set of inputs?
- What is an efficient numbering scheme for loop-free paths?

Class Presentations on Chianti

- Reza
- Xin

Change Impact Analysis

- Given the differences between Po and Pn, identify code in Po that are potentially affected by the differences.
  - e.g. find all methods that are called after the changed method.
  - e.g. find all methods that are called after a changed method p and are on the call stack after p returns.

Chianti

- A change impact analysis tool
- Ren et al.
- OOPSLA 2004

Motivation

- To allow programmers to experiment with different edits (e.g. if the edits lead to failure, then use alternative edits.)
- To reduce the amount of time and efforts for running regression tests (similar to RTS)
- To reduce the amount of time spent in debugging (similar to fault localization & delta debugging)

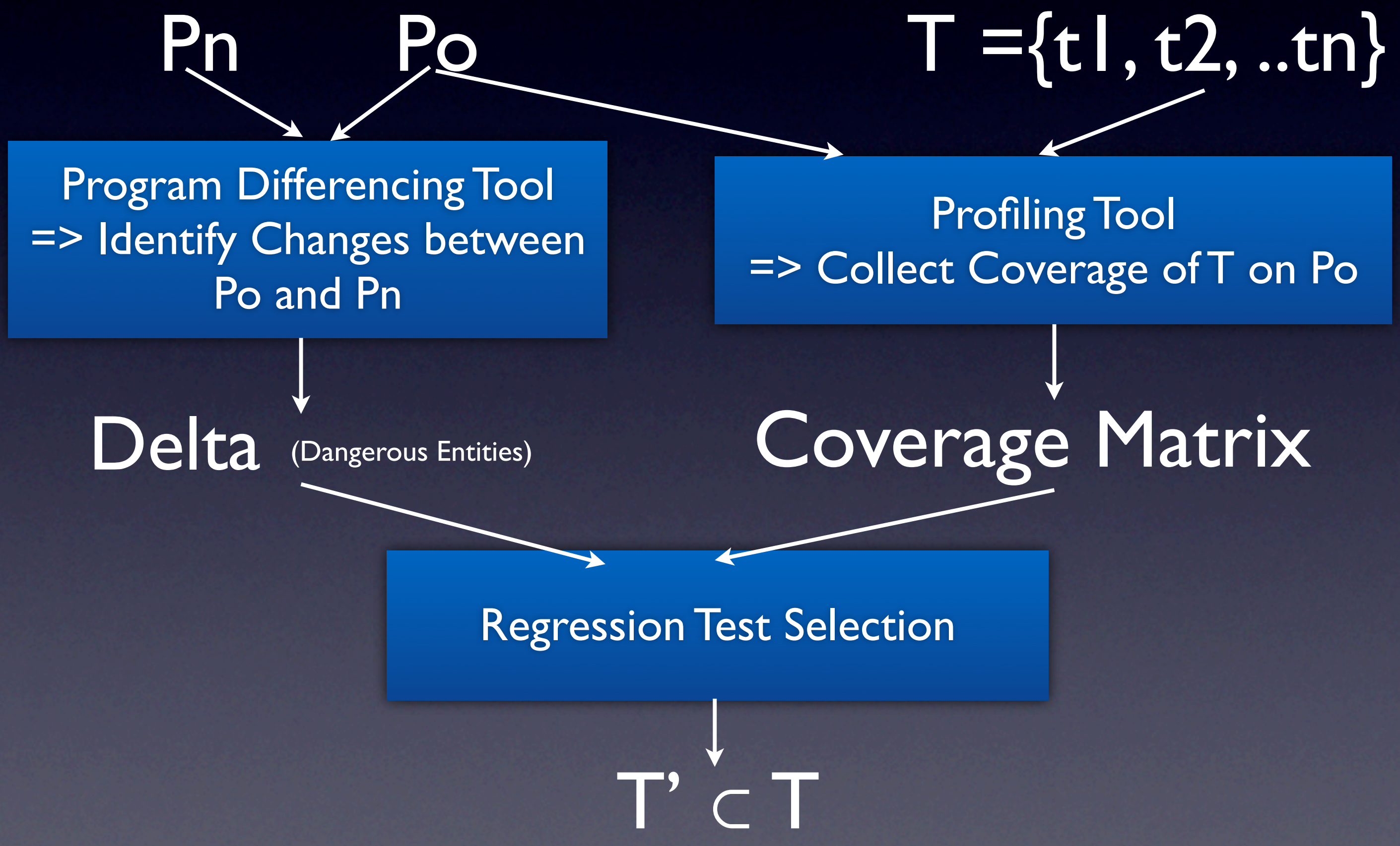
Change Impact Analysis Problem Framework

- Input
  - Po (old version)
  - Pn (new version)
  - Delta between Po and Pn
  - A test suite T for Po

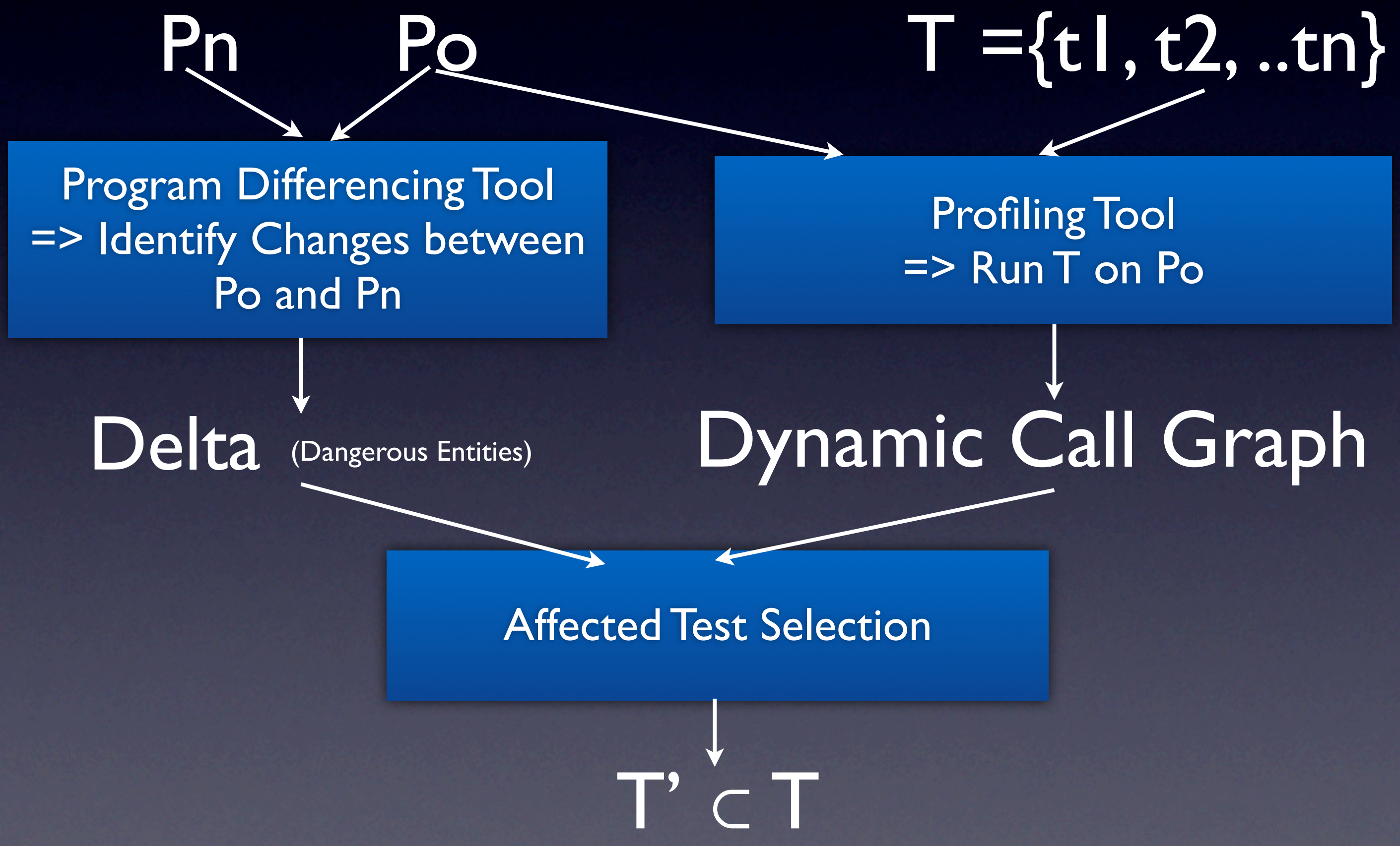
Two Research Questions in Chianti

- First phase: Which test cases do I have to rerun on Pn to identify potential regression faults? (Very similar to RTS problem)
- Second phase: For those tests that were selected & failed, which subset of the delta between Po and Pn led to behavior differences?

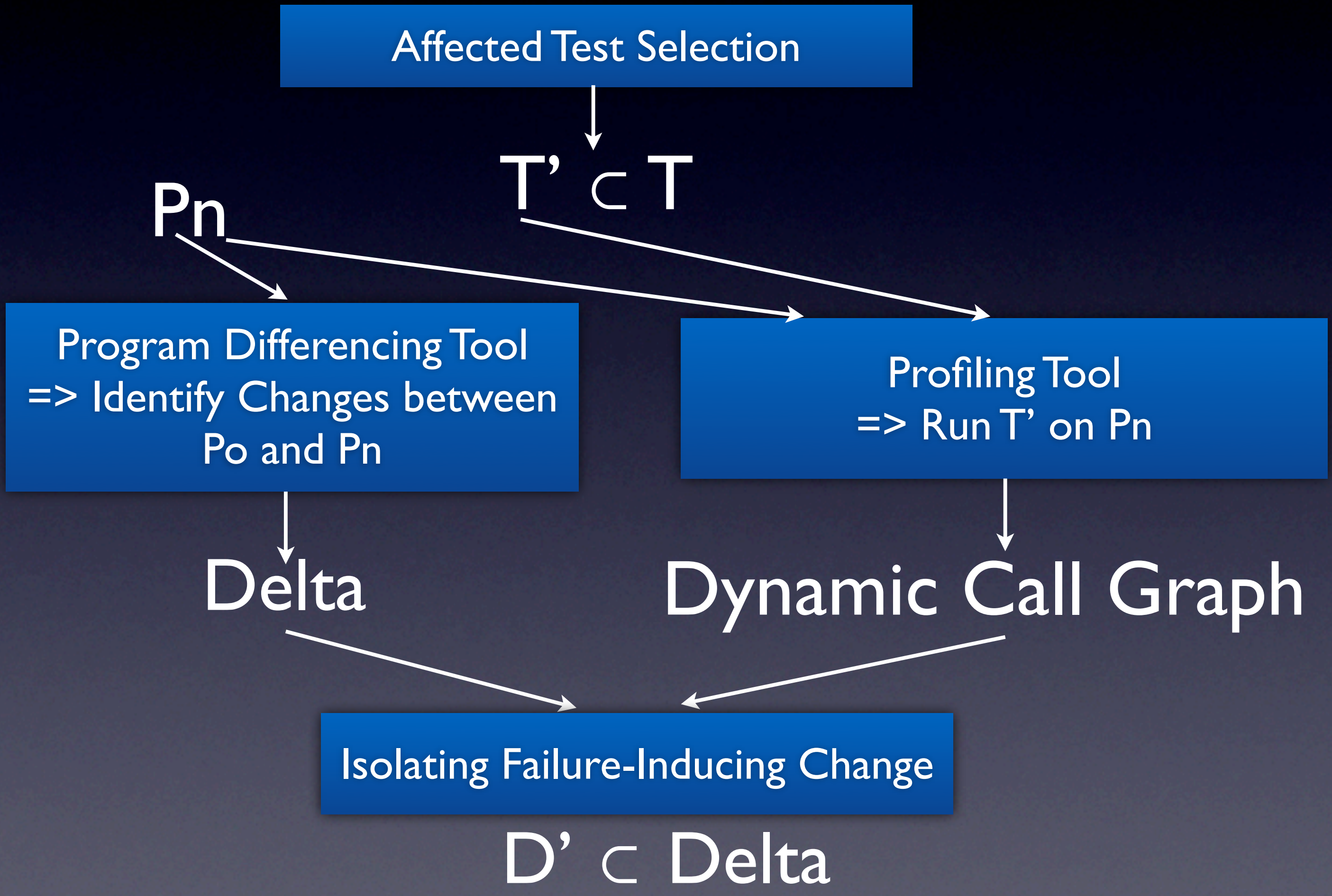
Recap: RTS Framework


Pn
Po
T ={t1, t2, ..tn}
Program Differencing Tool
=> Identify Changes between Po and Pn
Profiling Tool
=> Collect Coverage of T on Po
Delta
(Dangerous Entities)
Coverage Matrix
Regression Test Selection
T’ ⊂ T

Chianti Framework
First Phase



Chianti Framework
Second Phase



Affected Test Selection

T’ ⊂ T

Pn

Program Differencing Tool
=> Identify Changes between
Po and Pn

Profiling Tool
=> Run T’ on Pn

Delta

Dynamic Call Graph

Isolating Failure-Inducing Change

D’ ⊂ Delta

How to select affected tests T' $\subset$ T ?

- Identify a test if its dynamic call graph on the old version contains a node that corresponds to a change method (CM) or deleted method (DM)
- Or if the call graph contains an edge that corresponds to a lookup change (LC)

How to isolate changes Delta’ ⊂ Delta ?

- All atomic changes for added methods (AM) and changed methods (CM) that correspond to a node in the dynamic call graph of the new program version, Pn
- Atomic changes in the lookup change (LC) that correspond to an edge in the dynamic of the new program version.
- Their transitively prerequisite atomic changes.

Recap

- We learned how statement coverage, branch coverage and path coverage are different from one another.
- Chianti combines the regression test selection problem and fault localization problem.
- Chianti models a program delta as a set of interdependent atomic changes.

Preview for This Wed & Next Mon

- We will move on to a new topic, reverse engineering and knowledge discovery => software metrics & visualization
  - Murphy et al. Software Reflexion Model (Wed, 4/15)
  - Lanza et al. Polymetric Views (Mon, 4/20)

Announcement

- Preliminary grading guidelines for projects / literature surveys are uploaded on the blackboard.
- I am thinking about having a quiz on Chianti or Software Reflexion Model paper. If we have one, it will be this wednesday or next monday.
- There is no class lecture on Apr 29th. Use it for your project presentation & report preparation.