Understanding Regression Failures

Roykrong Sukkerd, Ivan Beschastnikh, Jochen Wuttke, Sai Zhang, Yuriy Brun

University of Washington
University of Massachusetts, Amherst
Motivation

(All changes) +Δ

(Min failing) +Δf

Guilty changes: Δf

(Max passing) +Δp

Guilty changes: Δp

−Δp
Minimal failing change set

All changes

Δ

Δf

Smallest set of changes that break the test

Zeller et al. TSE 2002
Maximal passing change set

Largest set of changes that do not break the test

Smallest set of changes to undo to pass test

All changes

$\Delta p$

$\Delta p$
Question

Do $\Delta f$ and $\Delta p$ identify the same set as guilty changes?

Not always
double getSurfaceArea(double r, double h) {
    double baseCir = 2 * Math.PI * r;
    double baseArea = Math.PI * r * r; // Bug: buggy pow(int,int)
    double lateralArea = 2 * Math.PI * r * h; // Bug: + should be *
    return 2 * baseArea + lateralArea;
}

@Test
assert (getSurfaceArea(2, 4) == 24 * Math.PI);

But, after fixing this line, the test still fails.
As bug indicator

double getSurfaceArea(double r, double h) {
    double baseCir = 2 * Math.PI * r;
    double baseArea = Math.PI * r * r;          // Bug: buggy pow(int,int)
    double lateralArea = 2 * Math.PI * r * h;   // Bug: + should be *
    return 2 * baseArea + lateralArea;
}

@Test
assert (getSurfaceArea(2, 4) == 24 * Math.PI);

After fixing these 2 lines, the test passes.
\( \Delta p \) and \( \Delta f \) relationships

Total of 9 possible relationships
Case study: compare $\Delta p$ and $\Delta f$

- Voldemort: distributed key-value storage system
- 130K LOC
- Of 305 revisions, found 45 regression failures
- Computed and compared $\Delta p$ and $\Delta f$
\[\Delta p \, \text{and} \, \Delta f \, \text{relationships}\]

1. 65%
2. 10.5%
3. 9.3%
4. 8.7%
5. 2.3%
6. 2.3%
7. 1.7%
8. 0%
9. 0%
There might be multiple bugs. $\Delta p$ is likely to catch more.
\[ \Delta f = \Delta \bar{p} \]

No difference between inspecting \( \Delta \bar{p} \) and \( \Delta f \).
Failure cause is *interaction* between changes in $\Delta p$ and $\overline{\Delta p}$.
\[ \Delta p \text{ and } \Delta f \text{ relationships} \]

- 87% $\Delta p \neq \Delta f$
- 78% $\Delta p$ contains changes not in $\Delta f$

4. 8.7%
5. 2.3%
6. 2.3%
7. 1.7%
8. 0%
9. 0%
Related work

- Determine which changes should be examined:
  - most cross-cutting concerns [Eaddy TSE’08]
  - modules with highest churn [Nagappan ICSE’05]
  - modules with most dependencies [Zimmermann ESEM’09]
- Delta debugging [Zeller TSE’02]
- Safe-Commit analysis [Wloka ICSE’09]
- Change impact analysis [Ren TSE’06, Zhang PASTE’08]
Future work

- Study how often defects are in $\Delta p$ and not in $\Delta f$.
- Develop a technique that leverages $\Delta p$ and $\Delta f$ to help developers debug.
Contributions

- $\Delta p$: changes we need to undo to regain correct behavior
- 9 possible relationships b/w $\Delta p$ and $\Delta f$
  - 87%: $\Delta p \neq \Delta f$
  - 78%: $\Delta p$ contains changes not in $\Delta f$

Recommendation: Considering $\Delta p$ in addition to $\Delta f$ may benefit debugging.