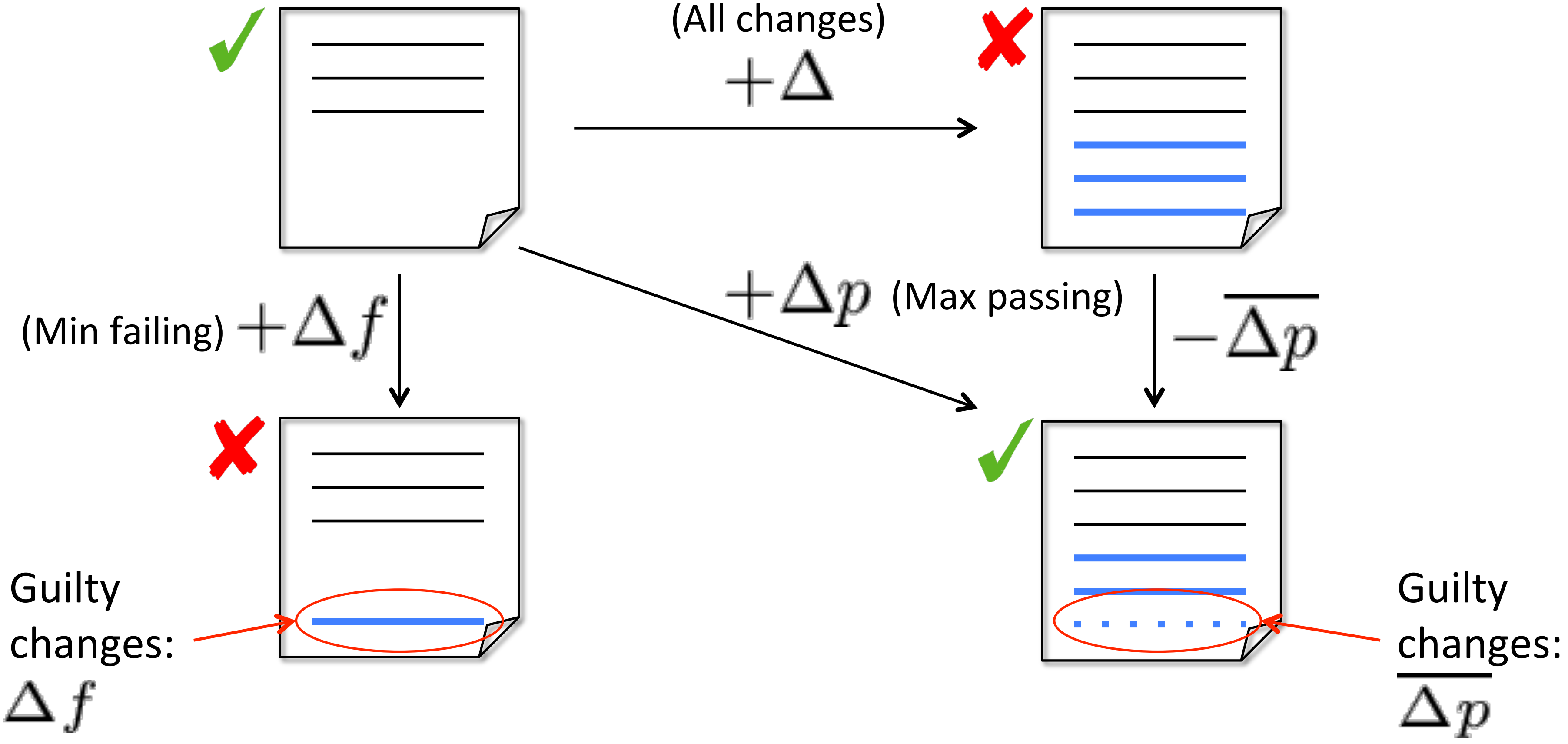
Understanding Regression Failures

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

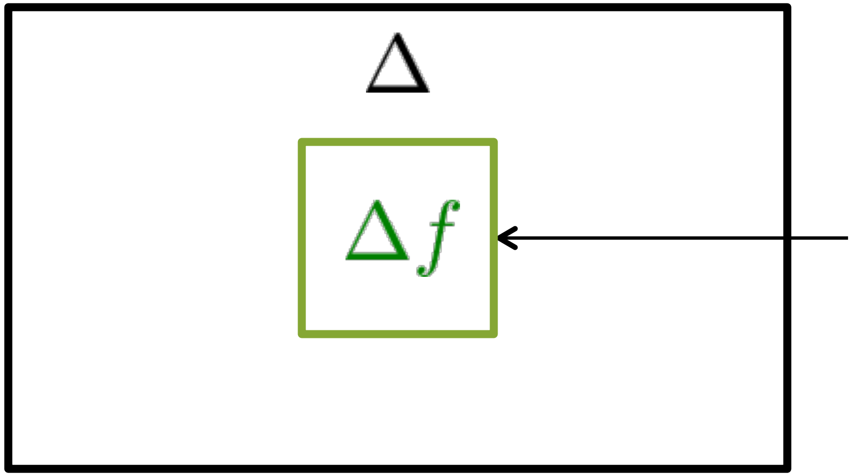
University of Washington
University of Massachusetts, Amherst

Motivation


✓
(All changes)
$+\Delta$
✗
(Min failing) $+\Delta f$
$+\Delta p$ (Max passing)
$-\overline{\Delta p}$
✗
✓
Guilty changes:
Δf
Guilty changes:
$\overline{\Delta p}$

Minimal failing change set

All changes



Δ

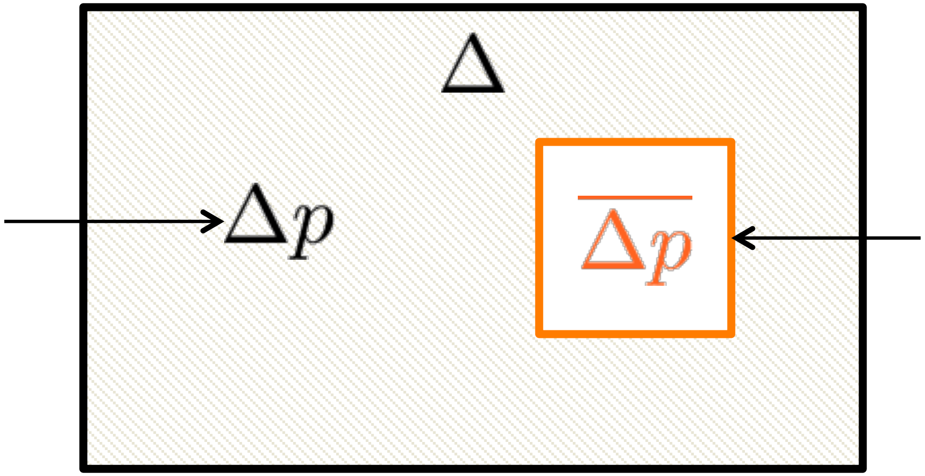
Δf

Smallest set of changes that break the test

Zeller et al. TSE 2002

Maximal passing change set

All changes



Largest set of changes that do not break the test

Smallest set of changes to undo to pass test

Question

Do Δf and $\overline{\Delta p}$ identify the same set as guilty changes?

Not always

Δf as bug indicator

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

But, after fixing this line,
the test still fails.

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

$\overline{\Delta p}$ as bug indicator

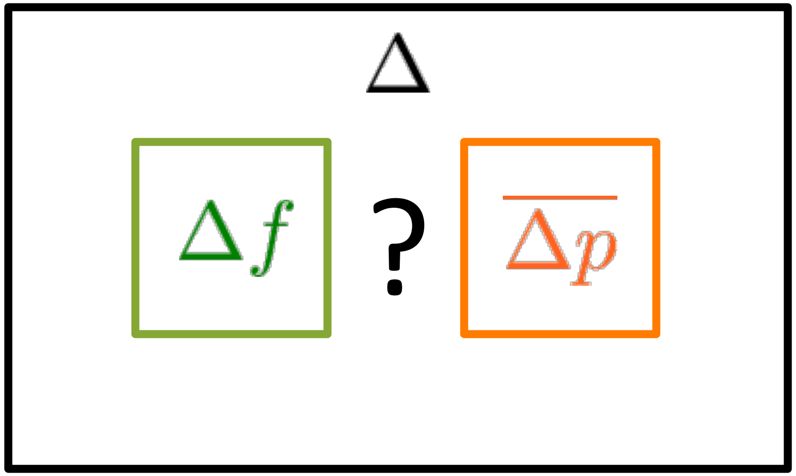
```
double getSurfaceArea(double r, double h) {
    double baseCir = 2 * Math.PI * r;
    double baseArea = Math.PI * ~~r * r;~~ pow(r, 2);          // Bug: buggy pow(int,int)
    double lateralArea = ~~2 * Math.PI * r * h;~~ baseCir + h;      // Bug: + should be *
    return 2 * baseArea + lateralArea;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

$\overline{\Delta p}$

After fixing these 2 lines, the test passes.

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

$\overline{\Delta p}$ and Δf relationships



Δ

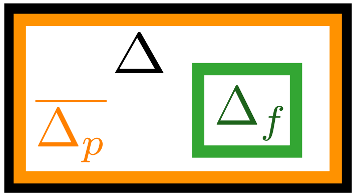
Δf ? $\overline{\Delta p}$

Total of 9 possible relationships

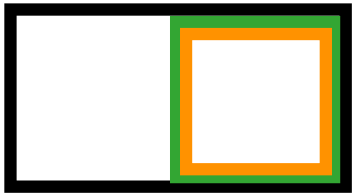
Case study: compare $\overline{\Delta p}$ and Δf

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

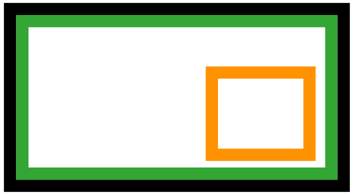
$\overline{\Delta_p}$ and Δ_f relationships



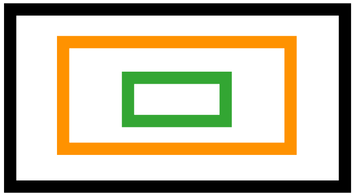
1. 65%



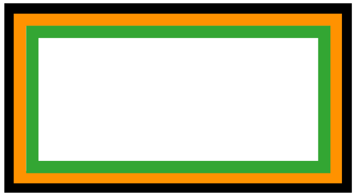
2. 10.5%



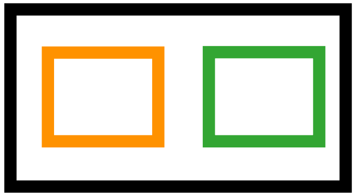
3. 9.3%



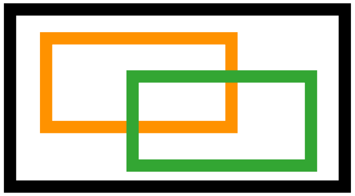
4. 8.7%



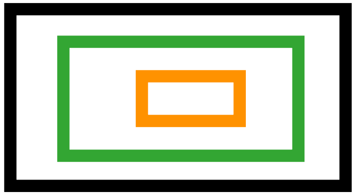
5. 2.3%



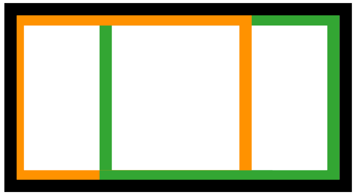
6. 2.3%



7. 1.7%

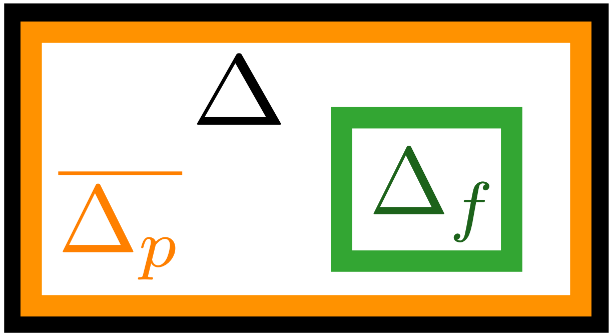


8. 0%

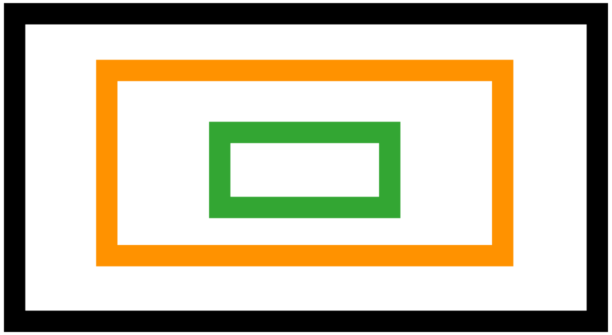


9. 0%

$\Delta f \subset \overline{\Delta p}$: multiple bugs



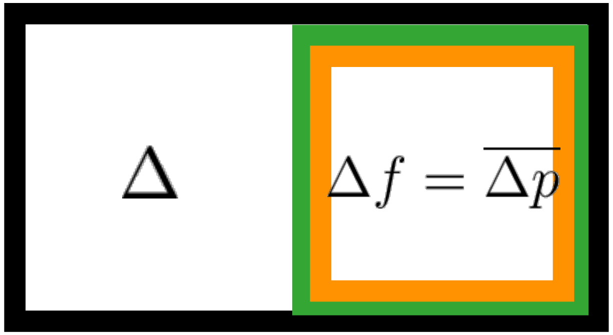
1. 65%



4. 8.7%

There might be multiple bugs. $\overline{\Delta p}$ is likely to catch more.

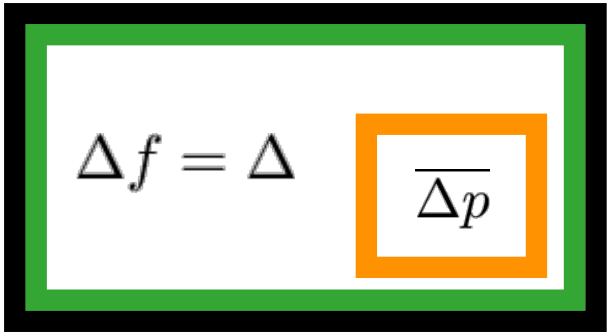
$\Delta f = \overline{\Delta p}$



2. 10.5%

No difference between inspecting $\overline{\Delta p}$ and Δf.

$\overline{\Delta p} \subset \Delta f$: interaction causes failure

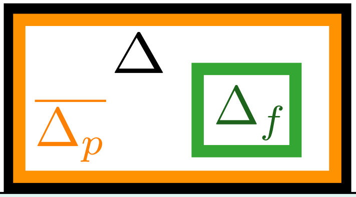


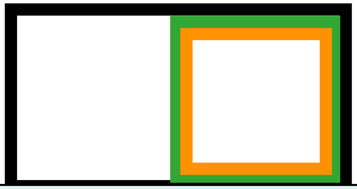
$\Delta f = \Delta$ $\overline{\Delta p}$

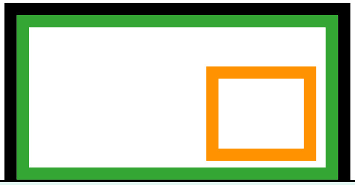
3. 9.3%

Failure cause is *interaction* between changes in Δp and $\overline{\Delta p}$.

$\overline{\Delta p}$ and Δf relationships





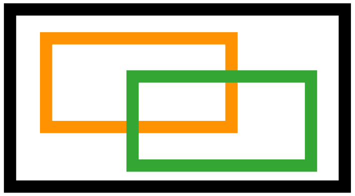


- 87% $\overline{\Delta p} \neq \Delta f$
- 78% $\overline{\Delta p}$ contains changes not in Δf

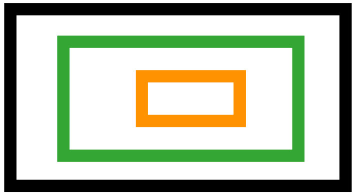
4. 8.7%

5. 2.3%

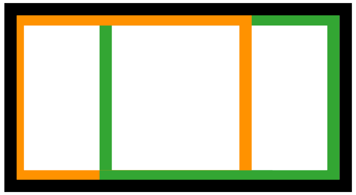
6. 2.3%



7. 1.7%



8. 0%



9. 0%

Related work

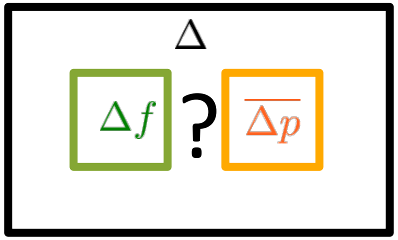
- Determine which changes should be examined:
  - most cross-cutting concerns [Eaddy TSE’08]
  - modules with highest churn [Nagappan ICSE’05]
  - modules with most dependencies [Zimmermann ESEM’09]
- Delta debugging [Zeller TSE’02]
- Safe-Commit analysis [Wloka ICSE’09]
- Change impact analysis [Ren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

Future work

- Study how often **defects** are in $\overline{\Delta p}$ and not in Δf.
- Develop a technique that leverages $\overline{\Delta p}$ and Δf to help developers debug.

Contributions

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



Recommendation: Considering $\overline{\Delta p}$ in addition to Δf may benefit debugging.