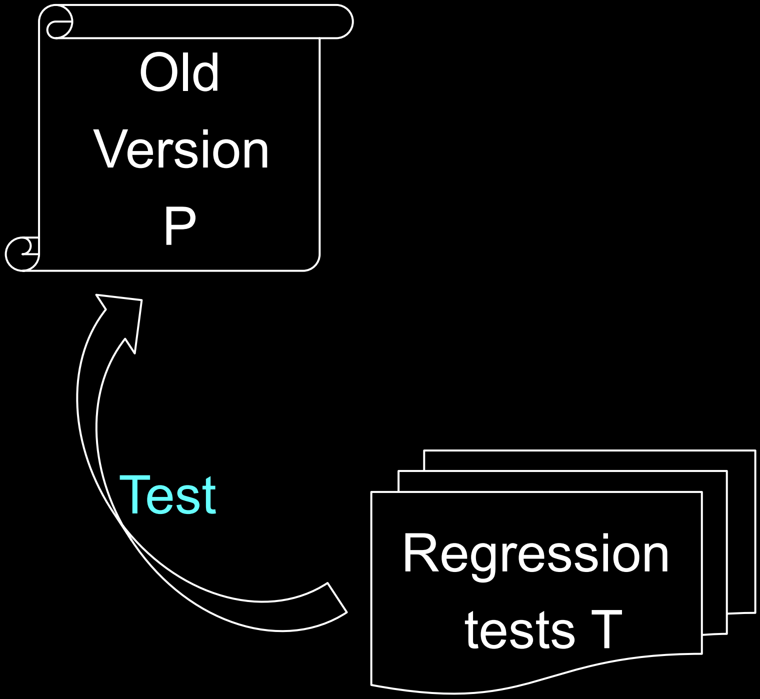
FaultTracer: A Change Impact and Regression Fault Analysis Tool for Evolving Java Programs

Lingming Zhang, Miryung Kim, Sarfraz Khurshid
University of Texas at Austin
zhanglm@utexas.edu

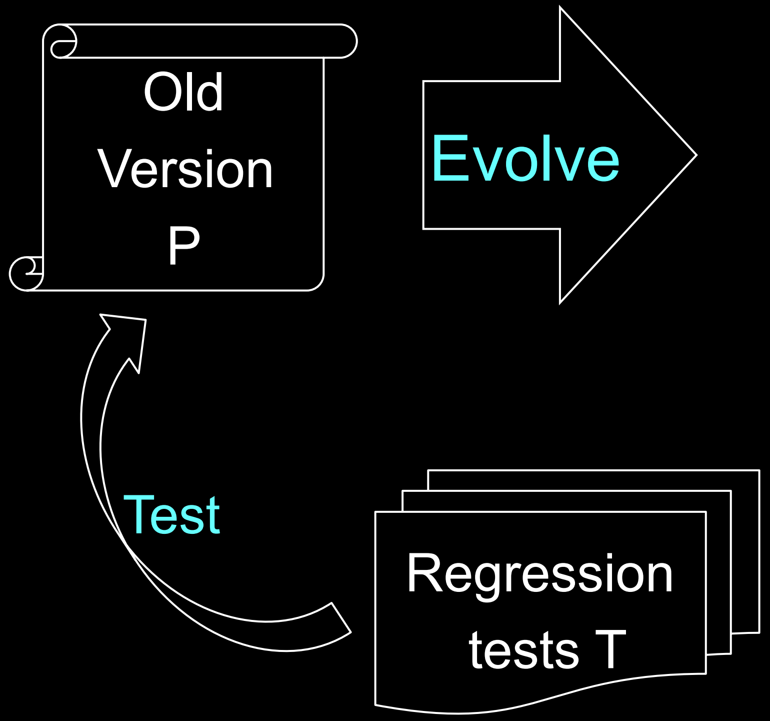
FSE Formal Research Demo Track

Nov 14, 2012

Scenario



Scenario



Old Version P

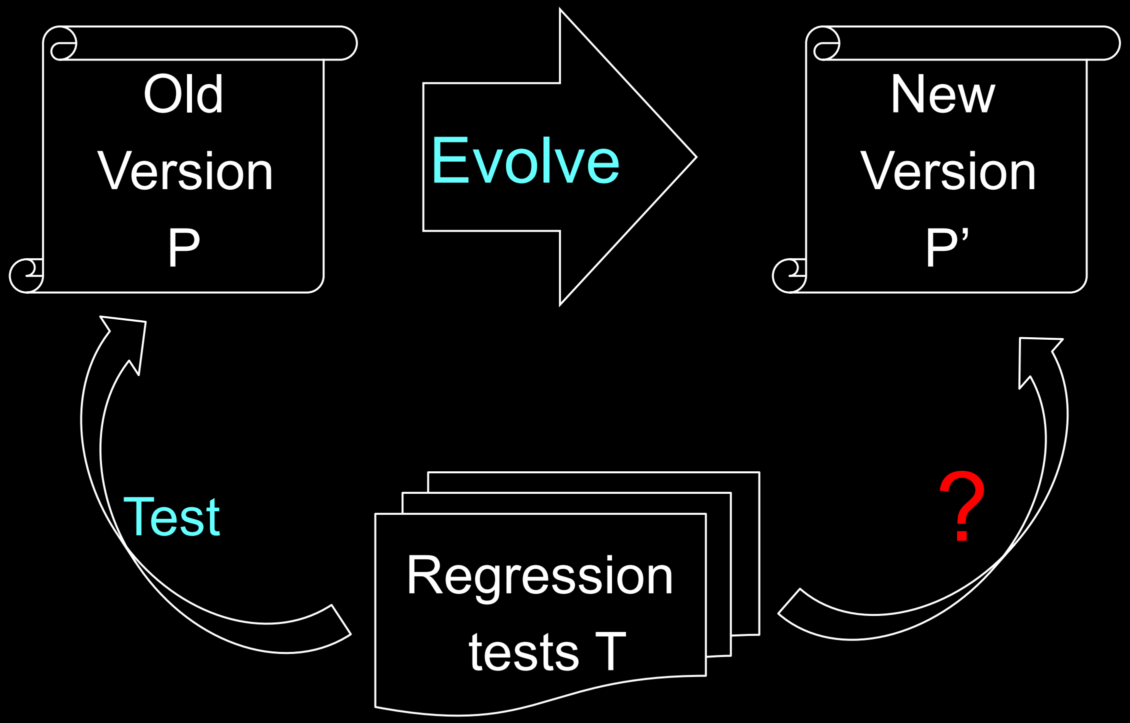
Evolve

New Version P’

Test

Regression tests T

Scenario



Old Version P

Evolve

New Version P'

Test

Regression tests T

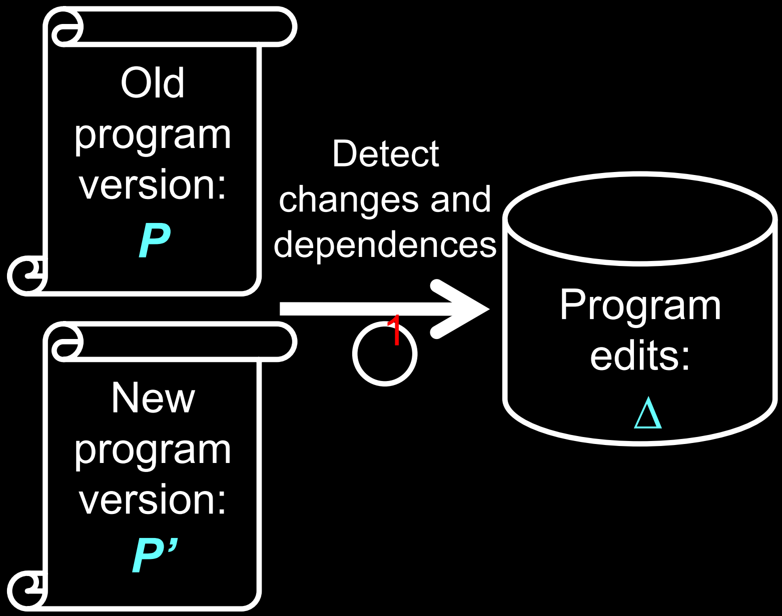
?

- Q1: How to *efficiently* run the regression tests?
  - Which tests are relevant to program edits?
- Q2: How to *effectively* localize faults when tests fail?
  - Which program edits are relevant to test failures?

Motivation

- Chianti change impact analysis is effective at finding suspicious edits but does not rank these edits. [Ren’04, Ren’06]
- Spectrum-based fault localization ranks potential faulty code fragments but does not focus on changes. [Jones’02, Abreu’07, Yu’08, Santelices’09, Parnin’11]
- Our insight is to combine change-impact analysis and spectrum-based fault localization [ICSM11].
  - Identify suspicious edits based on extended call graphs.
  - Rank suspicious edits using dynamic program spectrum information.

FaultTracer overview



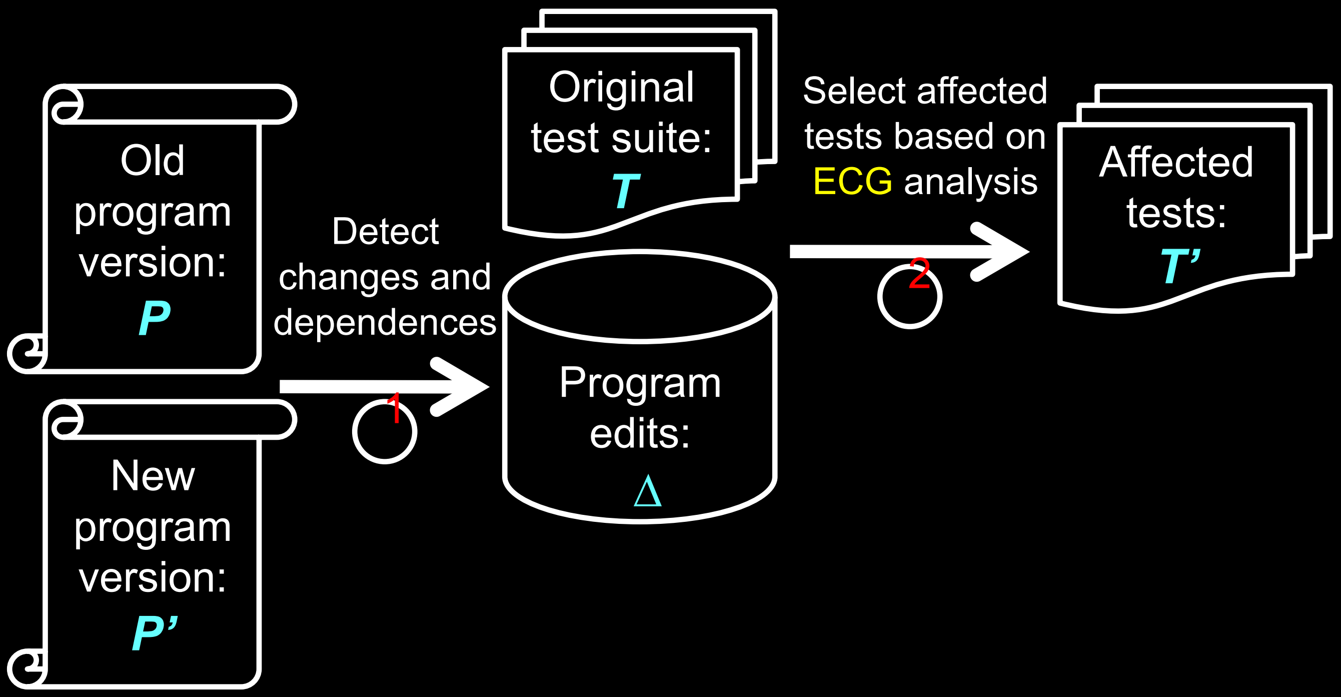
Old program version: *P*

New program version: *P'*

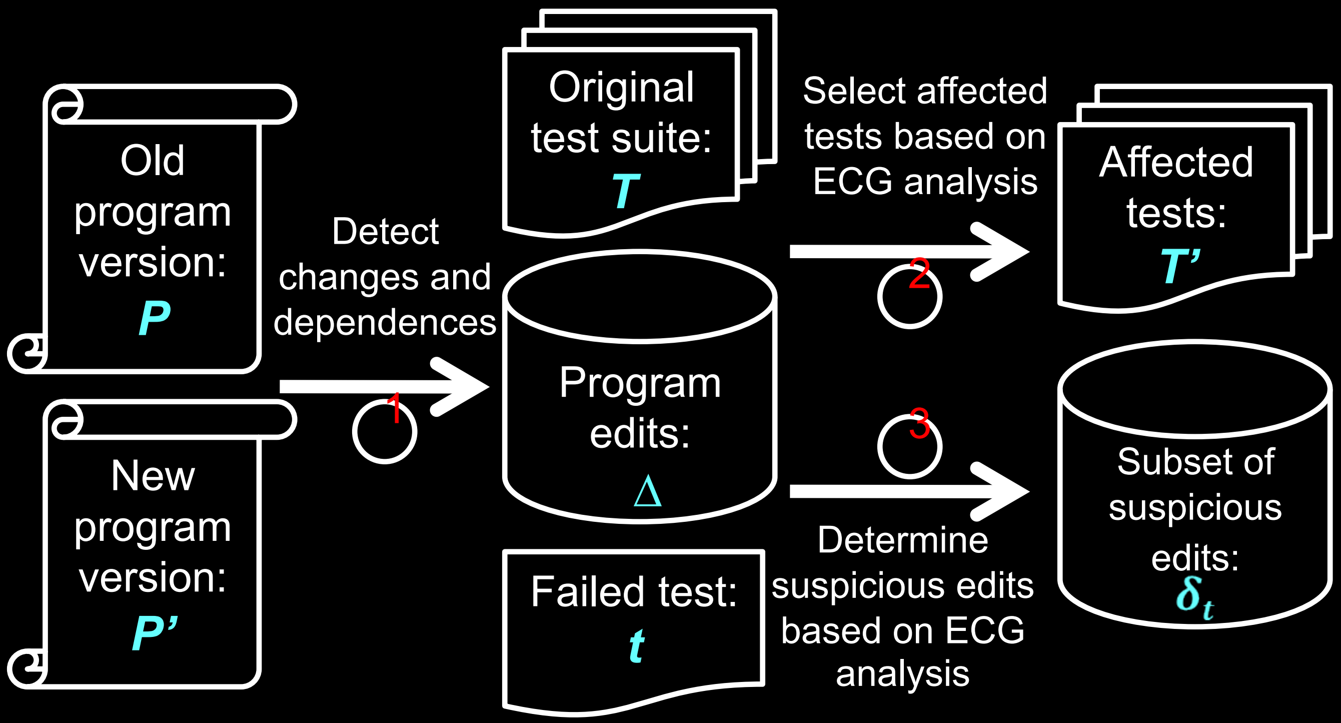
Detect changes and dependences

Program edits: Δ

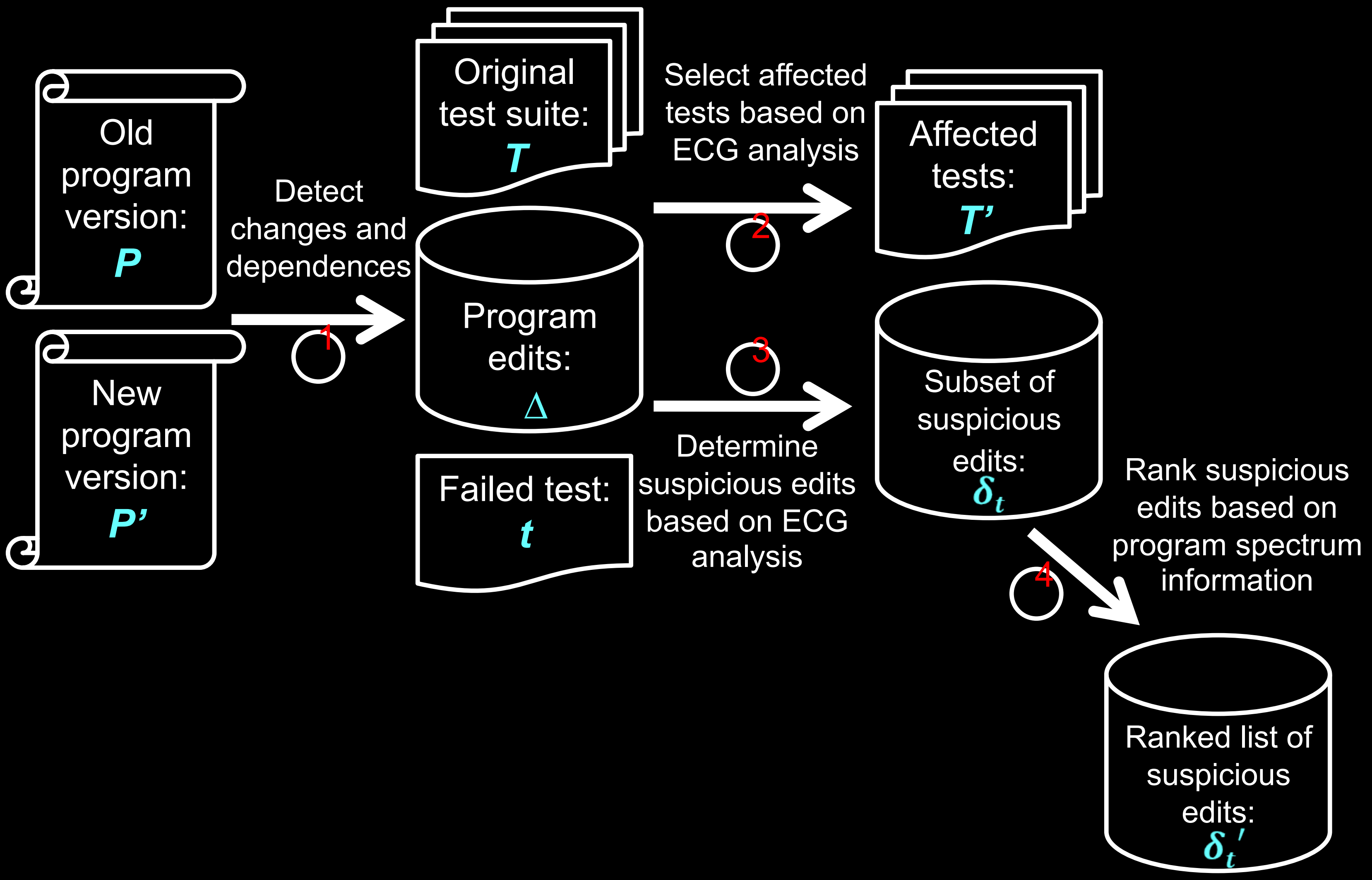
FaultTracer overview


Old program version: *P*
New program version: *P'*
Detect changes and dependences
1
Original test suite: *T*
Program edits: Δ
Select affected tests based on ECG analysis
2
Affected tests: *T'*

FaultTracer overview


Old program version: P
New program version: P'
Detect changes and dependences
1
Original test suite: T
Program edits: Δ
Failed test: t
Select affected tests based on ECG analysis
2
3
Determine suspicious edits based on ECG analysis
Affected tests: T'
Subset of suspicious edits: δ_t

FaultTracer overview



Old program version: P

New program version: P'

Detect changes and dependences

1

Original test suite: T

Program edits: Δ

Failed test: t

Select affected tests based on ECG analysis

2

Affected tests: T'

3

Determine suspicious edits based on ECG analysis

Subset of suspicious edits: δ_t

4

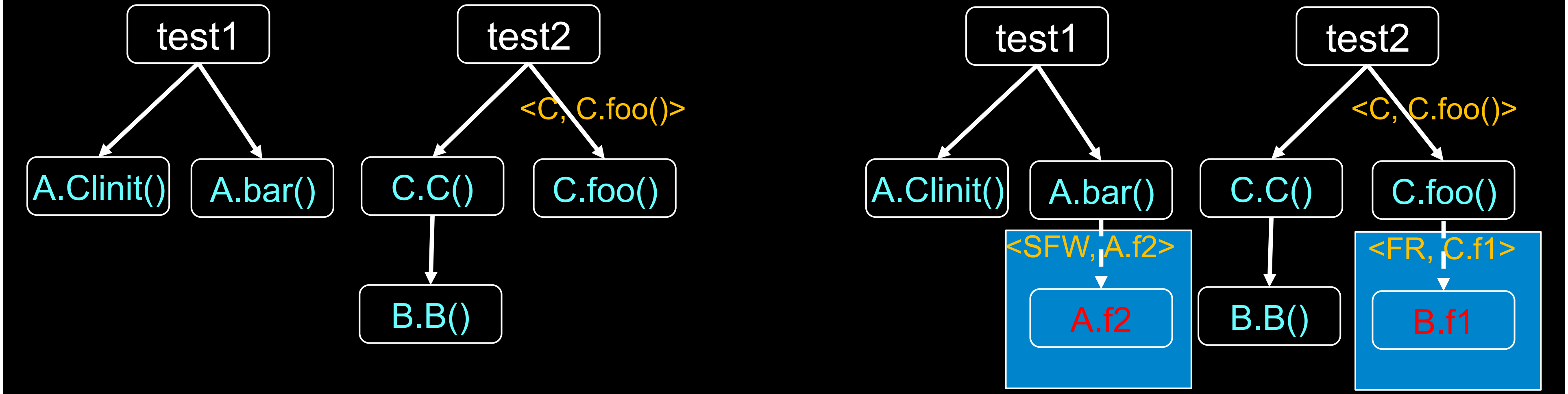
Rank suspicious edits based on program spectrum information

Ranked list of suspicious edits: δ_t'

Extended call graph (ECG)

Traditional Call Graphs used by Chianti [Ren'04]

Extended Call Graphs used by FaultTracer



Step 1: Detect atomic changes & dependences

Change types	Description
CM	Change method
AM	Add method
DM	Delete method
AF	Add field
DF	Delete field
CFI	Change instance field
CSFI	Change static field
LCm	Method look-up change
LCf	Field look-up change

- Dependence inference illustration
  - For every AM change, if a method called by the added method is new and all methods overridden by it are also new, the caller should be dependent on the added callee.

Step 2: Select tests based on ECG analysis

- FaultTracer directly matches all non-look-up changes with ECGs of the old version to select affected tests.
    - Existing technique needs to transform field changes into constructor change first.
- FaultTracer identifies tests that are influenced by method or field look-up changes as affected tests.
    - Existing technique does not handle field look-up change.

Step 3: Identify suspicious edits based on ECG analysis

- FaultTracer directly identifies all non-look-up changes on ECGs of the new version as suspicious edits.
  - Existing technique needs to select
    - the changes covered by affected tests.
    - the changes that these covered changes transitively depend on.
- FaultTracer identifies method or field level edits that caused look-up changes on ECGs as suspicious edits.
  - Existing technique cannot find field level edits that caused field look-up changes.

Step 4: Localize failure-inducing program edits using test spectra

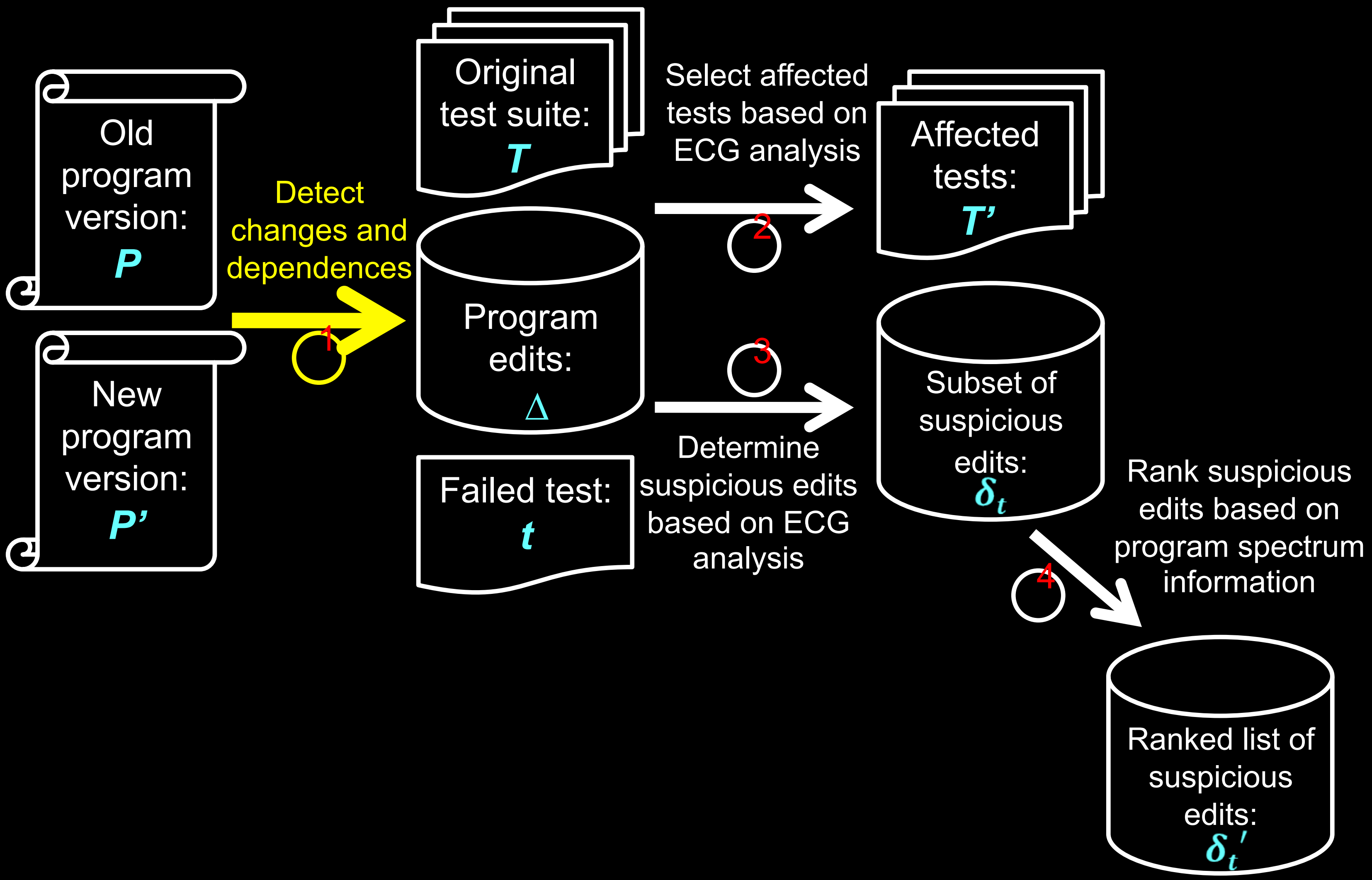
- Relation between suspicious edits and tests

Edits	Test1	Test2	test3	test4
Edit1	✘			
Edit2		✘		✘
Edit3			✘	✘
Edit4				✘
Result	Pass	Pass	Pass	Fail

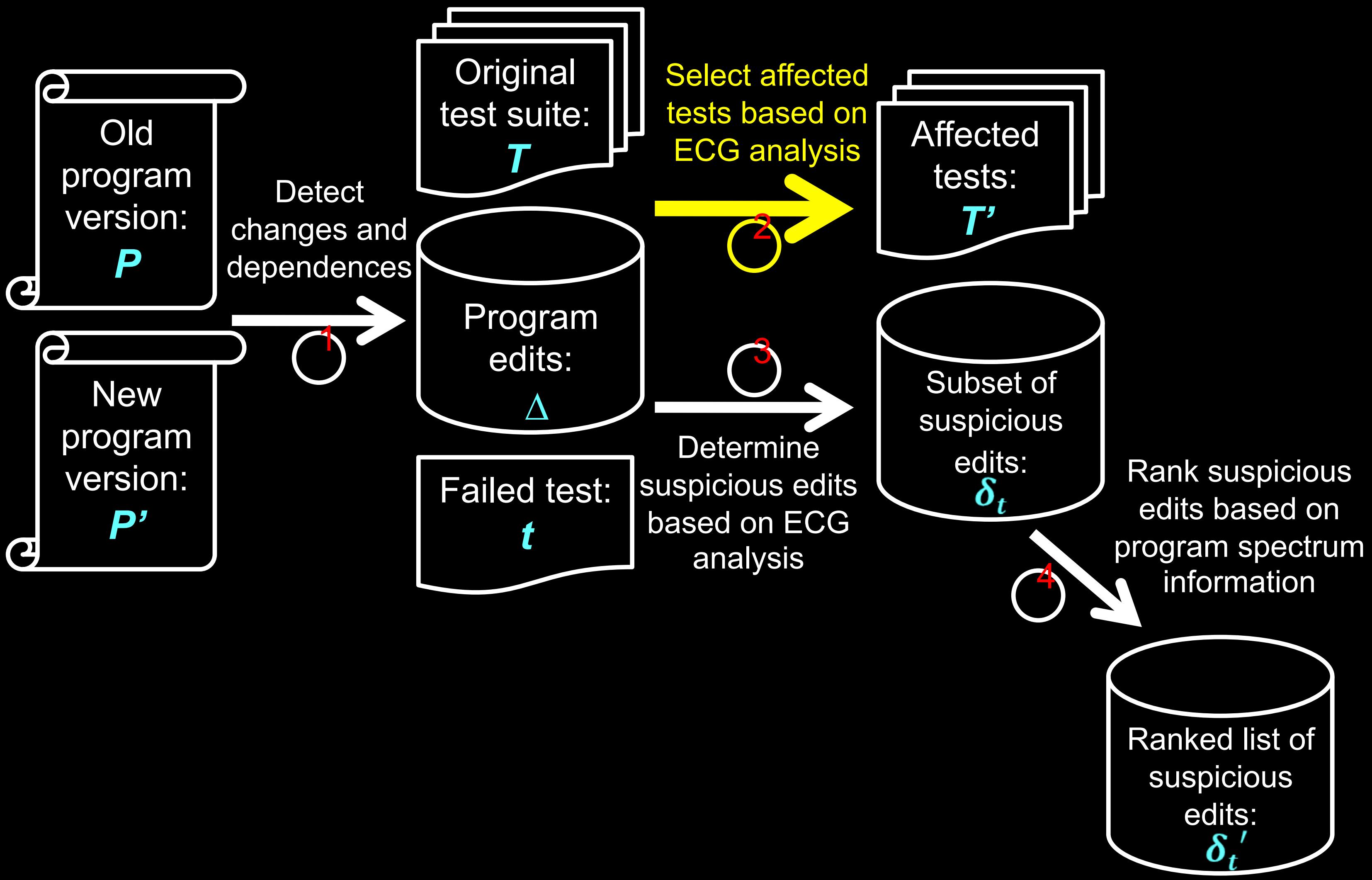
- Suspicious score calculation

Edits	Tarantula	SBI	Jaccard	Ochiai
	[Jones’02]	[Yu’08]	[Abreu’07]	[Abreu’07]
Edit1	0.00	0.00	0.00	0.00
Edit2	0.75	0.50	0.50	0.71
Edit3	0.75	0.50	0.50	0.71
Edit4	1.00	1.00	1.00	1.00

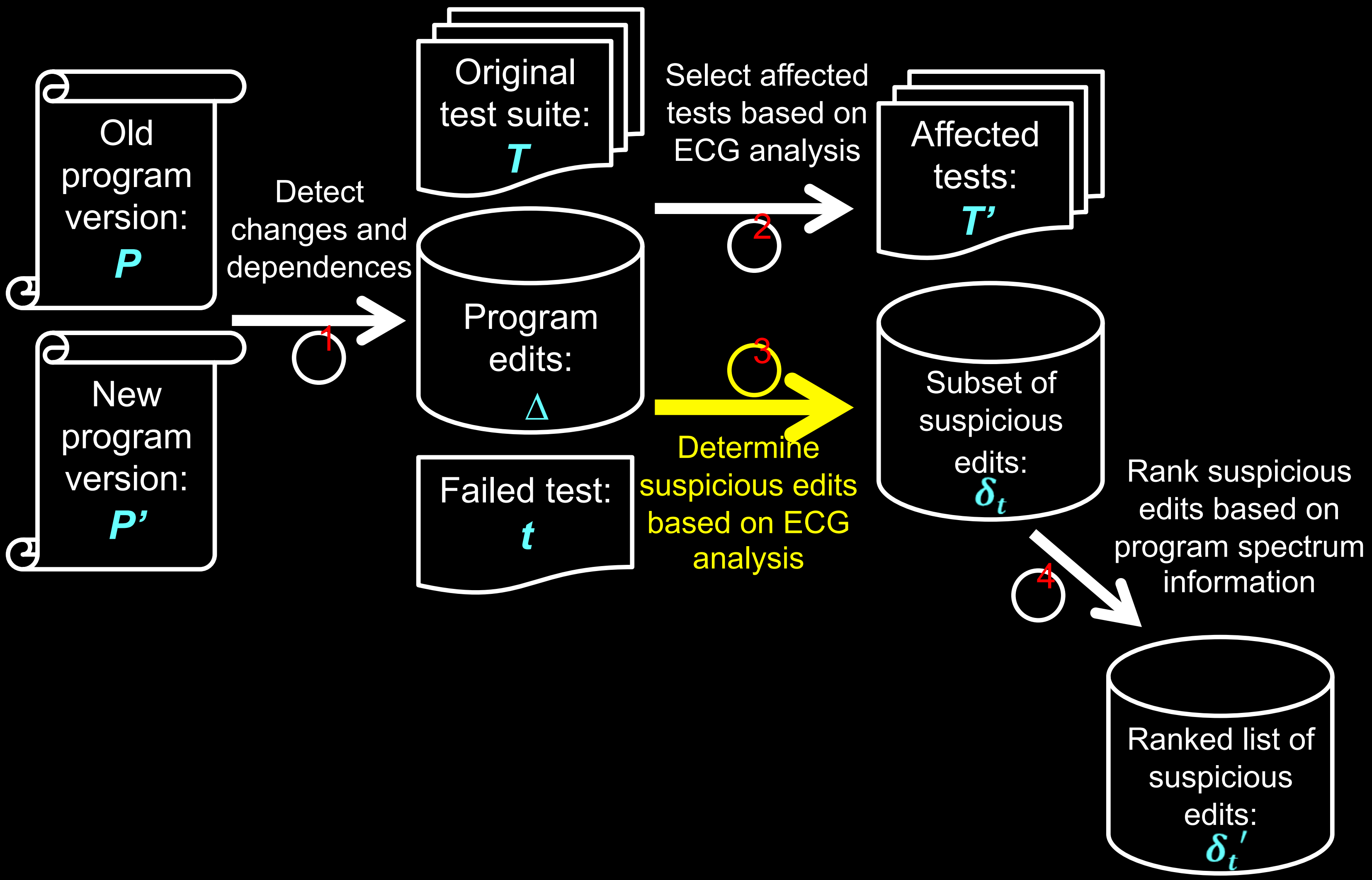
Demo: Step 1.
- Detect changes and dependences


Old program version: P
New program version: P'
Detect changes and dependences
1
Original test suite: T
Program edits: Δ
Failed test: t
Select affected tests based on ECG analysis
2
Affected tests: T'
3
Determine suspicious edits based on ECG analysis
Subset of suspicious edits: δ_t
Rank suspicious edits based on program spectrum information
4
Ranked list of suspicious edits: δ_t'

Demo: Step 2.
- Select affected tests


Old program version: P
New program version: P'
Detect changes and dependences
1
Original test suite: T
Program edits: Δ
Failed test: t
Select affected tests based on ECG analysis
2
Affected tests: T'
3
Determine suspicious edits based on ECG analysis
Subset of suspicious edits: δ_t
4
Rank suspicious edits based on program spectrum information
Ranked list of suspicious edits: δ_t'

Demo: Step 3.
- Determine suspicious edits



Old program version: *P*

New program version: *P'*

Detect changes and dependences

Original test suite: *T*

Program edits: Δ

Failed test: *t*

Select affected tests based on ECG analysis

Affected tests: *T'*

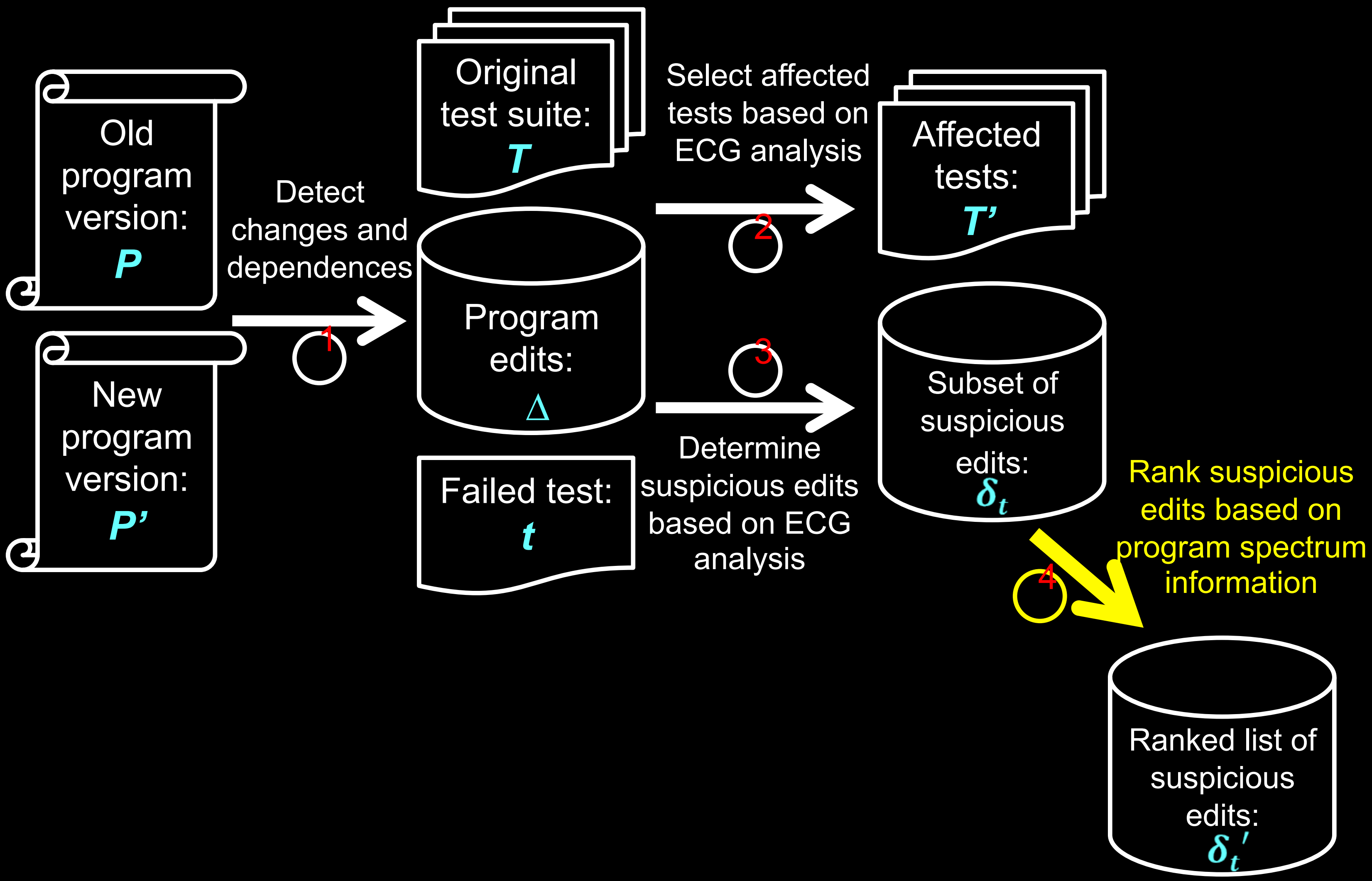
Determine suspicious edits based on ECG analysis

Subset of suspicious edits: δ_t

Rank suspicious edits based on program spectrum information

Ranked list of suspicious edits: δ_t'

Demo: Step 4.
- Rank suspicious edits



Old program version: P

New program version: P'

Detect changes and dependences

1

Original test suite: T

Program edits: Δ

Failed test: t

Select affected tests based on ECG analysis

2

Affected tests: T'

3

Determine suspicious edits based on ECG analysis

Subset of suspicious edits: δ_t

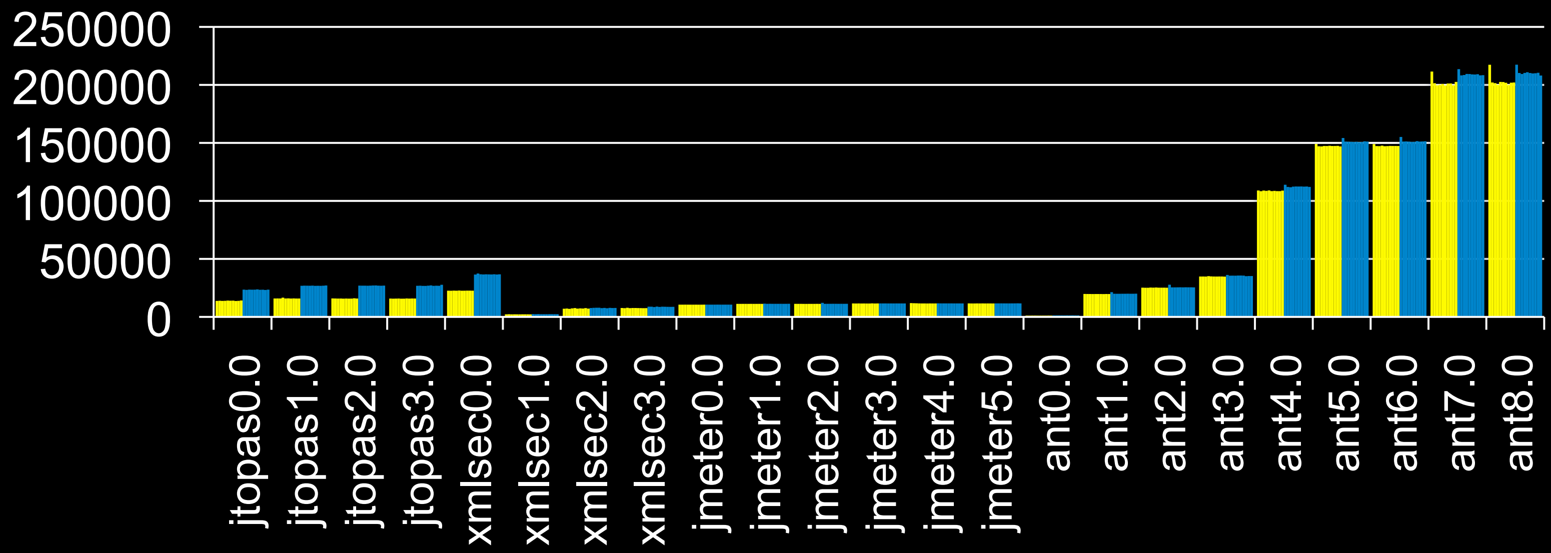
4

Rank suspicious edits based on program spectrum information

Ranked list of suspicious edits: δ_t'

Real-world Applications

- We have successfully applied FaultTracer to real-world Java programs ranging 1.83~80.44 KLoC
  - jtopas, xml-security, jmeter, and ant.
- Runtime overhead by Chianti and FaultTracer in collecting call graph information (ms).



Conclusion

- FaultTracer combines a Chianti-style change impact analysis with spectrum-based fault localization.
- FaultTracer improves a Chianti-style change impact analysis based on extended call graph analysis.
- Experimental results show that FaultTracer [ICSM11]
  - outperforms Chianti in determining affecting changes by 20%.
  - outperforms existing technique for localizing failure-inducing program edits by 50%.
- FaultTracer Eclipse plug-in is available for public download:

https://webspace.utexas.edu/lz3548/www/ftracer.html