Universität des Saarlandes, DFG-Graduiertenkolleg, November 13th, 2003

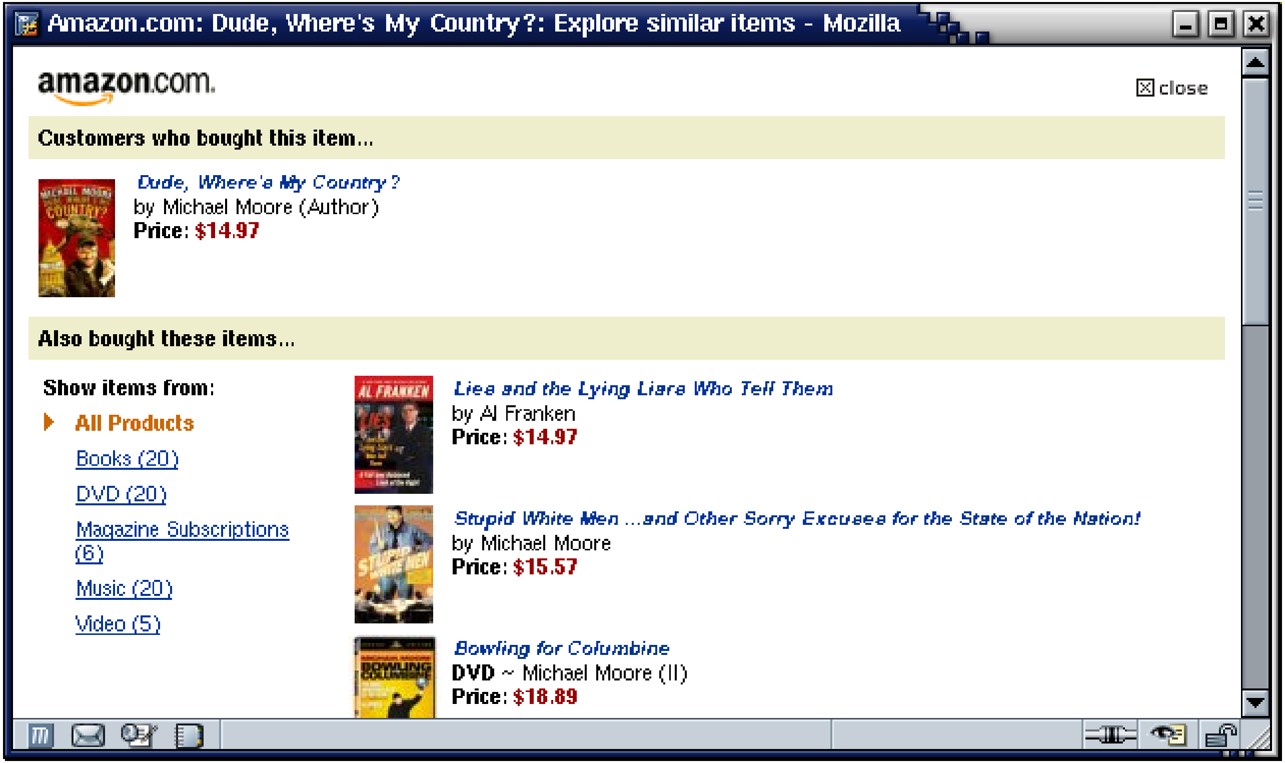
Data Mining Version Histories

Andreas Zeller

(with Thomas Zimmermann, Peter Weißgerber, and Stephan Diehl)

Lehrstuhl Softwaretechnik
Universität des Saarlandes, Saarbrücken

The Idea



Amazon.com: Dude, Where's My Country?: Explore similar items - Mozilla

amazon.com.

☒ close

Customers who bought this item...

Dude, Where's My Country?
by Michael Moore (Author)
Price: **$14.97**

Also bought these items...

Show items from:

- **All Products**
- Books (20)
- DVD (20)
- Magazine Subscriptions (6)
- Music (20)
- Video (5)

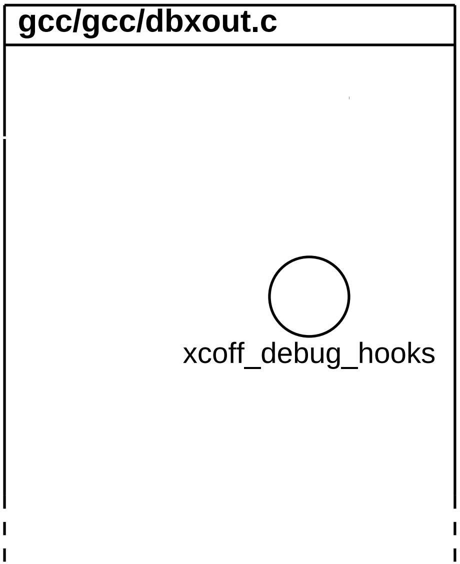
Lies and the Lying Liars Who Tell Them
by Al Franken
Price: **$14.97**

Stupid White Men ...and Other Sorry Excuses for the State of the Nation!
by Michael Moore
Price: **$15.57**

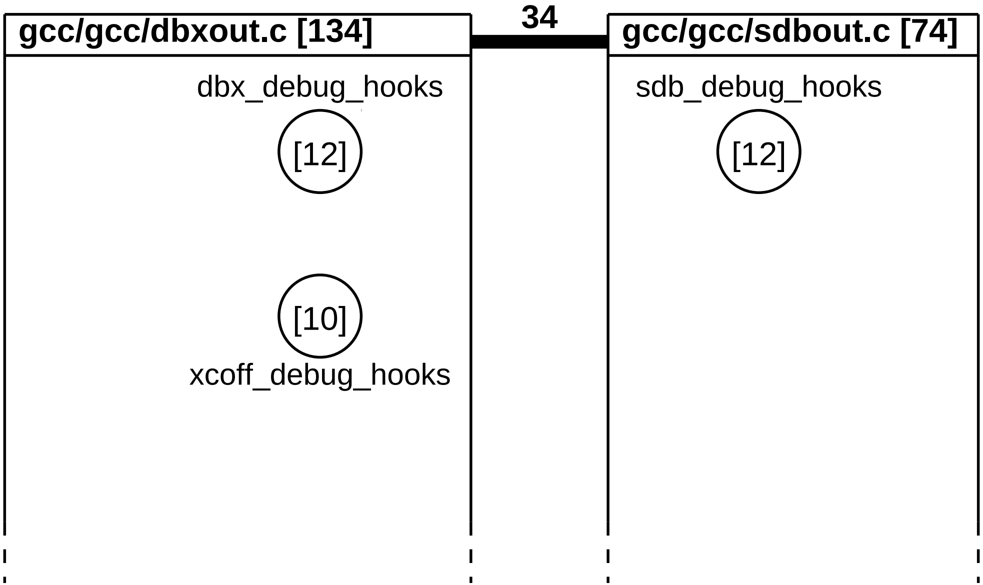
Bowling for Columbine
DVD ~ Michael Moore (II)
Price: **$18.89**

Can we make similar suggestions for *software changes?*

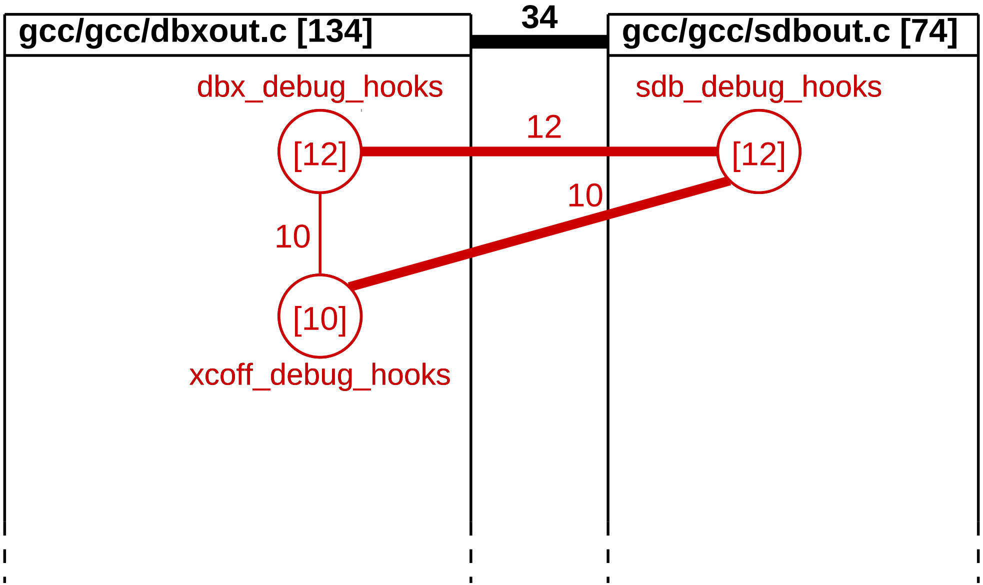
Evolutionary Coupling



Evolutionary Coupling

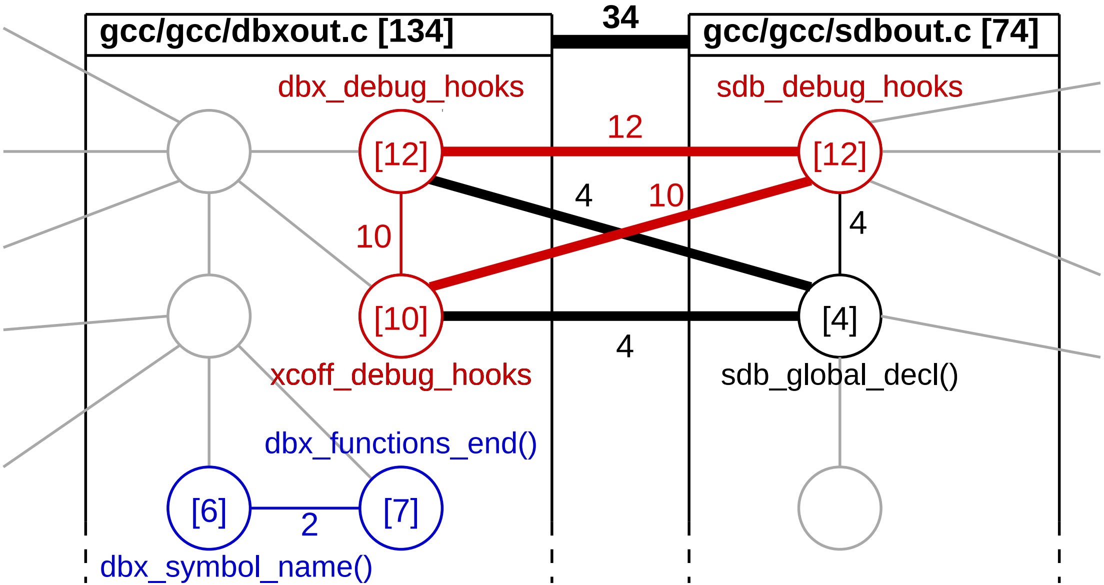


Evolutionary Coupling



Support: How much *evidence* (= simultaneous changes)?
Confidence: How *relevant* is coupling for participants?

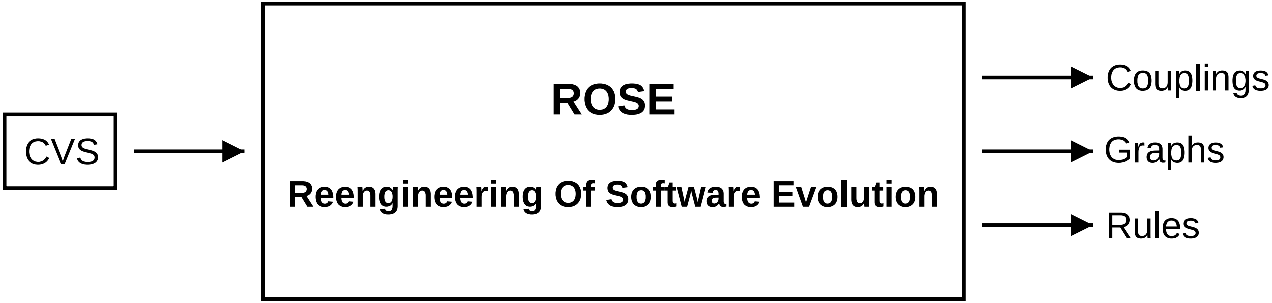
Evolutionary Coupling



Support: How much *evidence* (= simultaneous changes)?

Confidence: How *relevant* is coupling for participants?

What We Do



ROSE determines entities at different granularities:

coarse-granular entities: directories, modules, files

fine-granular entities: methods, variables, sections

Light-Weight Analysis

File: Animals.java

```
class Cat {
  public String[] COLORS = {
    ...
  }
  public Cat() {
    ...
  }
  ...
}

class Dog {
  public String[] COLORS = {
    ...
  }
  ...
}
```

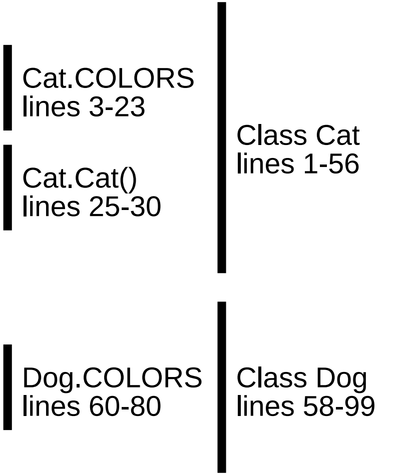
Light-Weight Analysis

File: Animals.java

```
 1  class Cat {
 3    public String[] COLORS = {
        ...
23    }
25    public Cat() {
        ...
30    }
      ...
56  }

58  class Dog {
60    public String[] COLORS = {
        ...
80    }
      ...
99  }
```

Step A: Map to Entities



Cat.COLORS
lines 3-23

Cat.Cat()
lines 25-30

Class Cat
lines 1-56

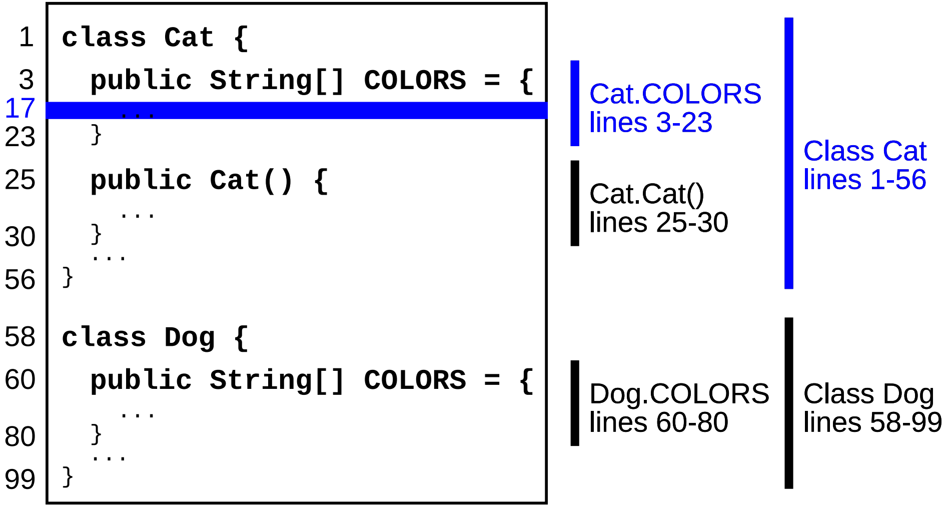
Dog.COLORS
lines 60-80

Class Dog
lines 58-99

Light-Weight Analysis

File: Animals.java

Step A: Map to Entities



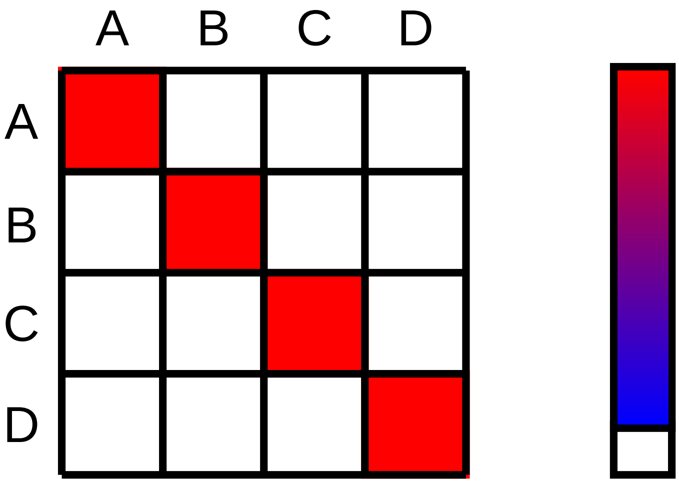
```
 1  class Cat {
 3    public String[] COLORS = {
17      ...
23    }
25    public Cat() {
        ...
30    }
      ...
56  }
58  class Dog {
60    public String[] COLORS = {
        ...
80    }
      ...
99  }
```

Step B: Filter Entities

ROSE analyzes C/C++, JAVA, PYTHON, TeX and TEXINFO files.

We get the modified *methods*, *variables* and *subsections*.

Visualizing Coupling



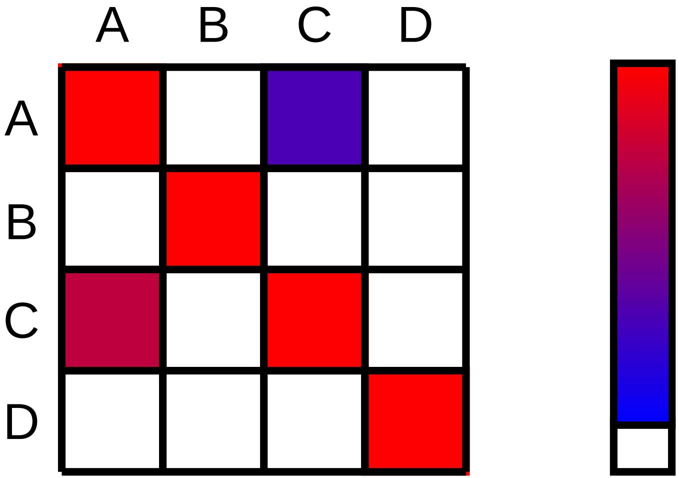
	A	B	C	D
A	🟥			
B		🟥		
C			🟥	
D				🟥

High Confidence

Low Confidence

No Coupling (No Support)

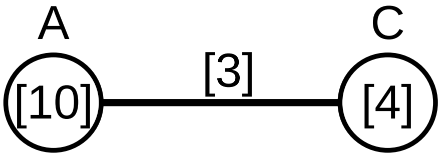
Visualizing Coupling



High Confidence

Low Confidence

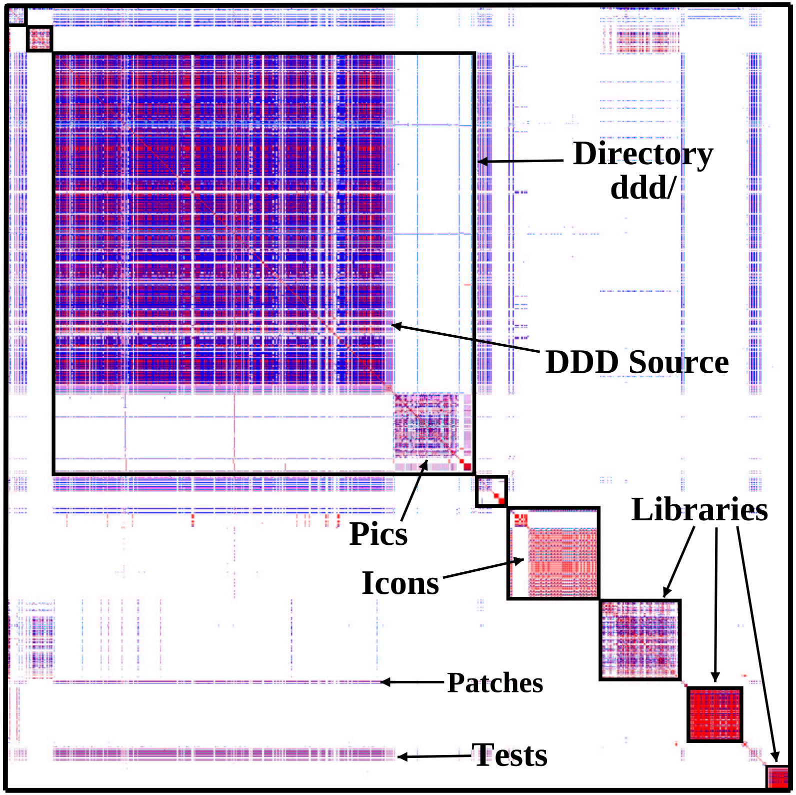
No Coupling (No Support)



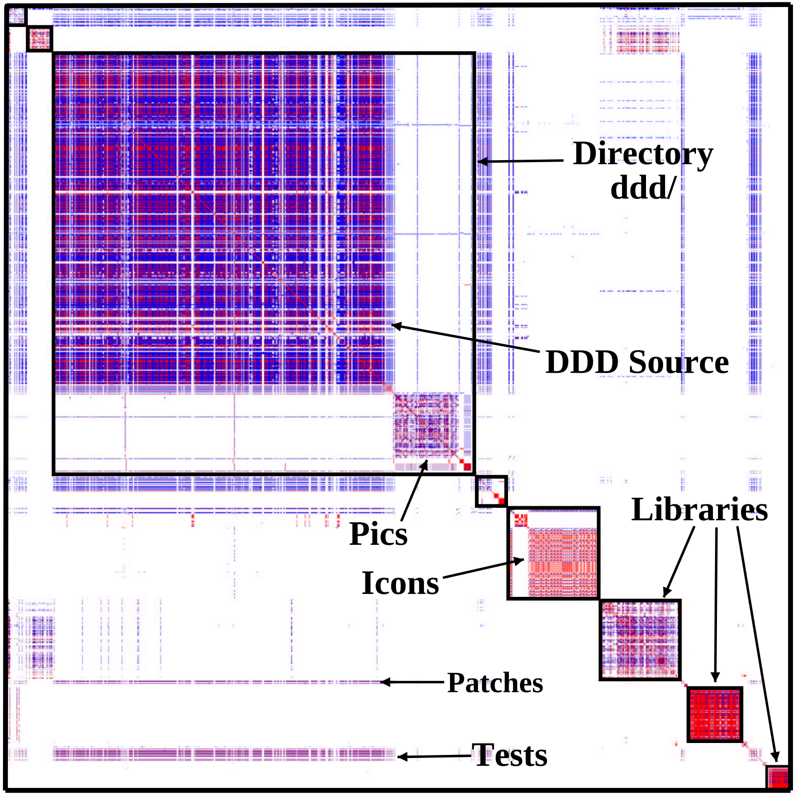
$A \Rightarrow C$: Confidence $3/10 = 30\%$

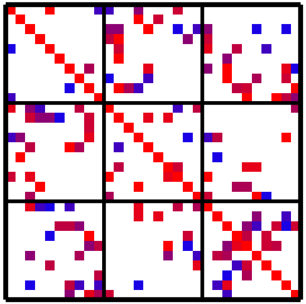
$C \Rightarrow A$: Confidence $3/4 = 75\%$

Comparing Architecture with Evolution

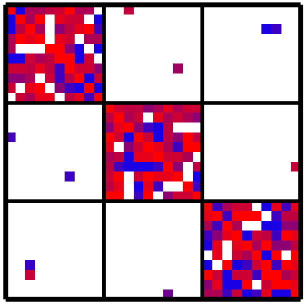


Comparing Architecture with Evolution





Bad architecture



Better architecture

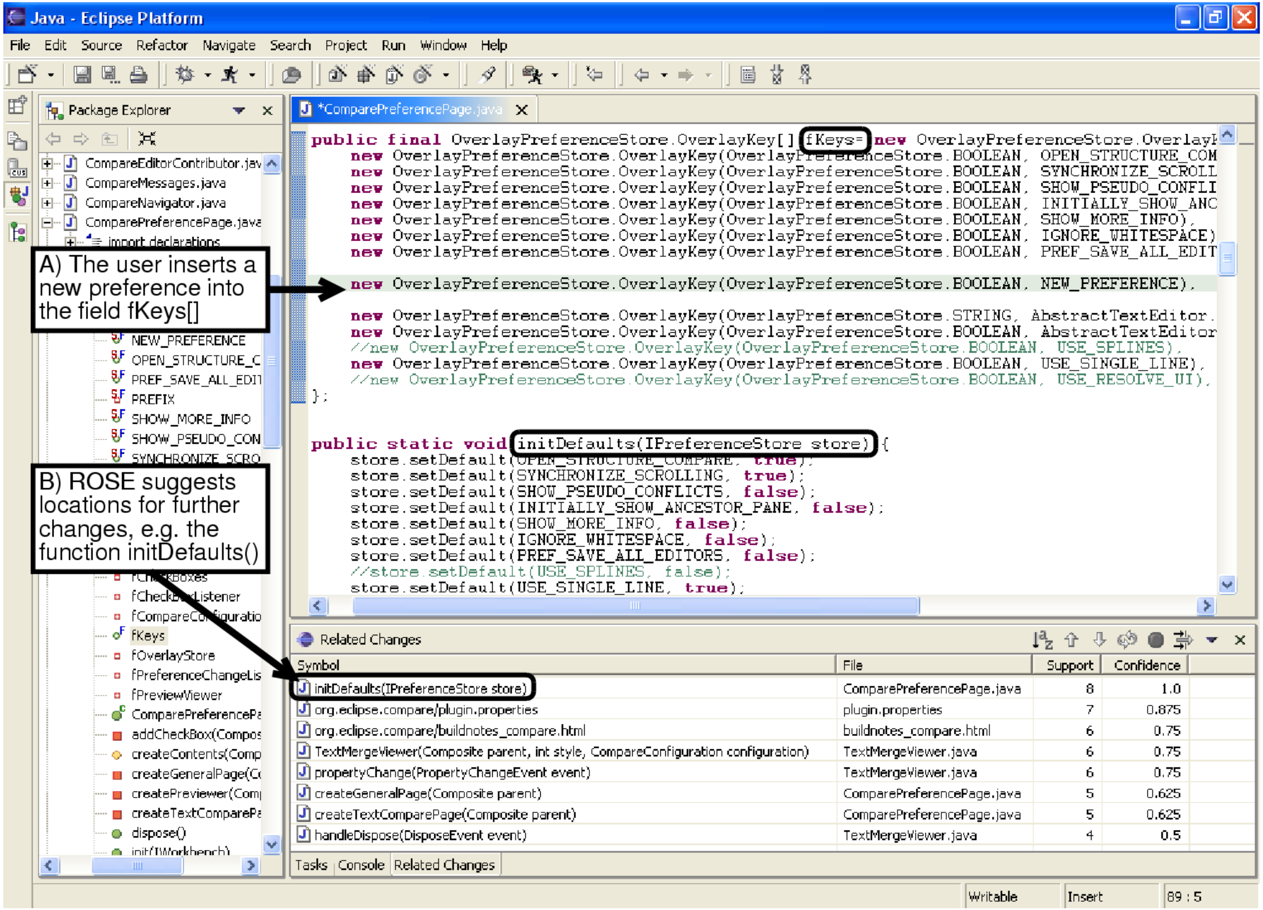
Guiding the Programmer

Understanding coupling based on evolution is neat—
but how do we put this to use?

Basic idea—*guide programmer along related changes:*

1. Programmer starts changing some location
2. ROSE suggests locations that other programmers have changed together with this location:
   “Programmers who changed this function also changed. . . ”

Guiding the Programmer in Eclipse



```
public final OverlayPreferenceStore.OverlayKey[] fKeys= new OverlayPreferenceStore.OverlayK
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, OPEN_STRUCTURE_COM
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, SYNCHRONIZE_SCROLL
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, SHOW_PSEUDO_CONFLI
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, INITIALLY_SHOW_ANC
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, SHOW_MORE_INFO),
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, IGNORE_WHITESPACE)
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, PREF_SAVE_ALL_EDIT

    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, NEW_PREFERENCE),

    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.STRING, AbstractTextEditor.
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, AbstractTextEditor
    //new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, USE_SPLINES),
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, USE_SINGLE_LINE),
    //new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, USE_RESOLVE_UI),
};


public static void initDefaults(IPreferenceStore store) {
    store.setDefault(OPEN_STRUCTURE_COMPARE, true);
    store.setDefault(SYNCHRONIZE_SCROLLING, true);
    store.setDefault(SHOW_PSEUDO_CONFLICTS, false);
    store.setDefault(INITIALLY_SHOW_ANCESTOR_PANE, false);
    store.setDefault(SHOW_MORE_INFO, false);
    store.setDefault(IGNORE_WHITESPACE, false);
    store.setDefault(PREF_SAVE_ALL_EDITORS, false);
    //store.setDefault(USE_SPLINES, false);
    store.setDefault(USE_SINGLE_LINE, true);
```

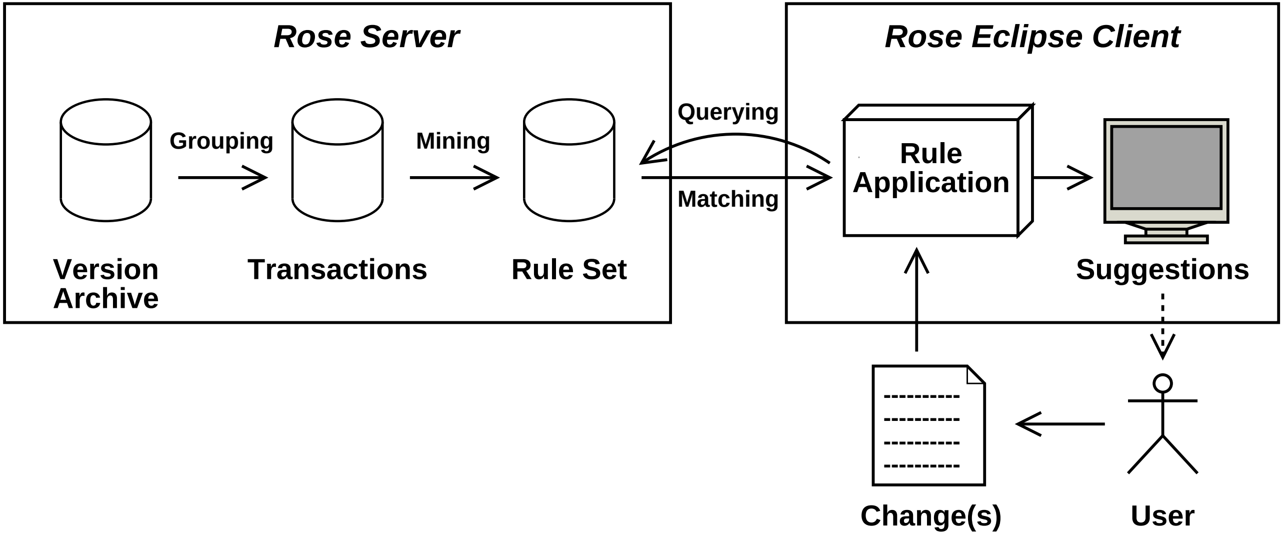
Symbol	File	Support	Confidence
initDefaults(IPreferenceStore store)	ComparePreferencePage.java	8	1.0
org.eclipse.compare/plugin.properties	plugin.properties	7	0.875
org.eclipse.compare/buildnotes_compare.html	buildnotes_compare.html	6	0.75
TextMergeViewer(Composite parent, int style, CompareConfiguration configuration)	TextMergeViewer.java	6	0.75
propertyChange(PropertyChangeEvent event)	TextMergeViewer.java	6	0.75
createGeneralPage(Composite parent)	ComparePreferencePage.java	5	0.625
createTextComparePage(Composite parent)	ComparePreferencePage.java	5	0.625
handleDispose(DisposeEvent event)	TextMergeViewer.java	4	0.5

A) The user inserts a new preference into the field fKeys[]

B) ROSE suggests locations for further changes, e.g. the function initDefaults()

ROSE Server and Client

The *ROSE server* determines coupling and rules;
The *ROSE client* guides the programmer along related changes.



Mining Rules

Coupling graphs turned out to be not predictive enough. So, we had ROSE use the *Apriori Algorithm* to mine rules:

1. Determine *frequent entity sets* L that are above the minimum support.
2. Create *rules* from the sets in L that are above the minimum confidence.

The generated rules have the form

$$\textit{antecedent(s)} \Rightarrow \textit{consequent(s)}$$

Whenever the user changes the antecedent(s) of a rule, ROSE suggests the consequent(s).

Rule Examples

Coupling in GCC

{ (i386.c, *var*, *i386_cost*), (i386.c, *var*, *i486_cost*),
(i386.c, *var*, *k6_cost*), (i386.c, *var*, *pentium_cost*),
(i386.c, *var*, *pentiumpro_cost*) }
⇒ { (i386.h, *type*, *processor_cost*) }

[Support 9; Confidence 0.82]

POSTGRESQL documentation

{ (createuser.sgml, *file*, createuser.sgml),
(dropuser.sgml, *file*, dropuser.sgml) }
⇒ { (createdb.sgml, *file*, createdb.sgml),
(dropdb.sgml, *file*, dropdb.sgml) }

[Support 11; Confidence 1.0]

Evaluation

How good are rules at predicting future changes?

We look at the histories of large software projects:

Training period. ROSE infers rules from the past.

Evaluation period. ROSE applies the mined rules.

In the evaluation period, we check each transaction Δ:

Navigation. Given *one change* from Δ, does ROSE point to further changes in Δ?

Error Prevention. Given *all but one change* from Δ, does ROSE point to the missing change?

Closure. Given *all changes* of Δ, does ROSE stay silent?

Precision vs. Recall

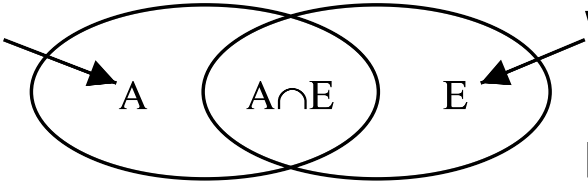
Recall: How many relevant entities are returned?

Precision: How many of the returned entities are relevant?

What ROSE finds

Precision

$|A \cap E| / |A|$



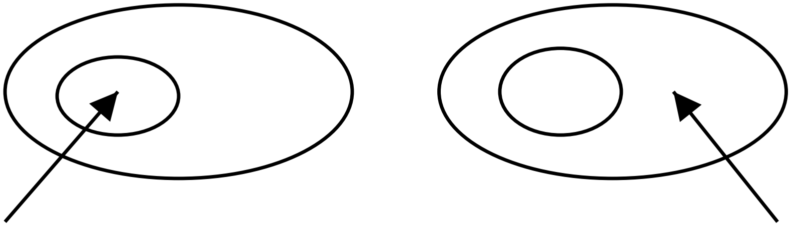
What it should find

Recall

$|A \cap E| / |E|$

High precision

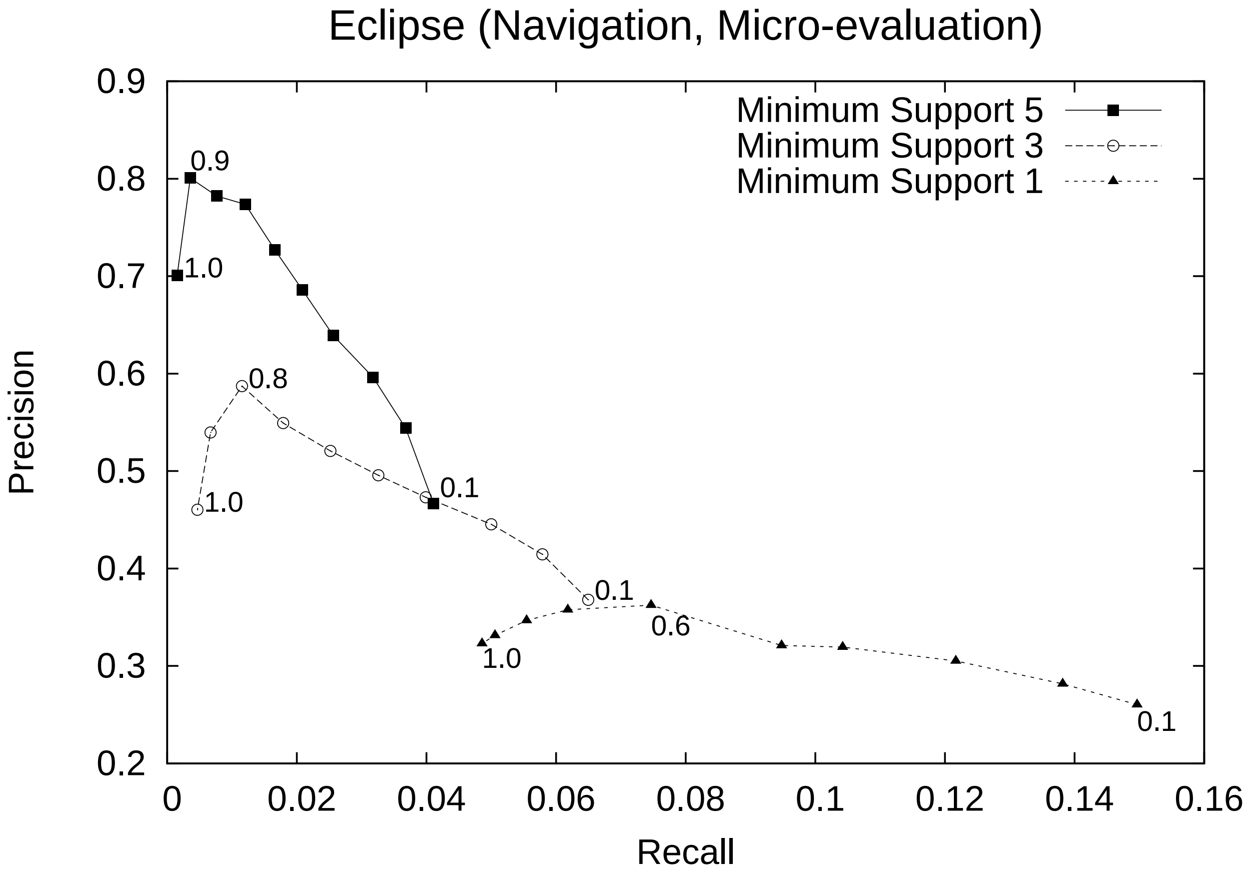
High recall



What ROSE finds

What ROSE finds

Precision vs. Recall (2)



Projects used for Evaluation

	Training			Evaluation
Project	# Txns	# Txns/Day	# Etys/Txn	# Txns
ECLIPSE	46,843	56.0	3.17	2,965
GCC	47,424	22.4	3.90	1,083
GIMP	9,796	4.1	4.54	1,305
JBOSS	10,843	9.0	3.49	1,320
JEDIT	2,024	2.9	4.54	577
KOFFICE	20,903	11.2	4.25	1,385
POSTGRES	13,477	5.4	3.27	925
PYTHON	29,588	6.2	2.62	1,201

Results: Navigation through Source Code

The programmer has changed one single entity.
Can ROSE suggest other entities that should be changed?

Granularity	Fine		Coarse	
Project	R_μ	P_μ	R_μ	P_μ
ECLIPSE	0.15	0.26	0.17	0.26
GCC	0.28	0.39	0.44	0.42
GIMP	0.12	0.25	0.27	0.26
JBOSS	0.16	0.38	0.25	0.37
JEDIT	0.07	0.16	0.25	0.22
KOFFICE	0.08	0.17	0.24	0.26
POSTGRES	0.13	0.23	0.23	0.24
PYTHON	0.14	0.24	0.24	0.36
Average	0.15	0.26	0.26	0.30

When given one initial changed entity, ROSE can predict 15% of all entities changed later in the same transaction. 26% of ROSE's suggestions actually took place.

Results: Error Prevention

The programmer has changed several entities but one.
Does ROSE find the missing one?

Granularity	Fine		Coarse	
Project	R_μ	P_μ	R_μ	P_μ
ECLIPSE	0.02	0.48	0.03	0.48
GCC	0.20	0.81	0.29	0.82
GIMP	0.03	0.71	0.08	0.74
JBOSS	0.01	0.24	0.05	0.44
JEDIT	0.004	0.59	0.01	0.44
KOFFICE	0.003	0.24	0.04	0.61
POSTGRES	0.03	0.66	0.05	0.59
PYTHON	0.01	0.50	0.03	0.67
Average	0.04	0.50	0.07	0.66

Given a transaction where one change is missing, ROSE can predict 4% of the entities that need to be changed.
On average, every second recommended entity is correct.

Results: Closure

The programmer made all necessary changes.

How often does ROSE still suggest a missing change?

Granularity	Fine		Coarse	
Project	R_M	P_M	R_M	P_M
ECLIPSE	1.0	0.979	1.0	0.980
GCC	1.0	0.953	1.0	0.946
GIMP	1.0	0.978	1.0	0.963
JBOSS	1.0	0.981	1.0	0.980
JEDIT	1.0	0.986	1.0	0.984
KOFFICE	1.0	0.990	1.0	0.971
POSTGRES	1.0	0.989	1.0	0.978
PYTHON	1.0	0.986	1.0	0.991
Average	1.0	0.980	1.0	0.973

ROSE’s warnings about missing changes should be taken seriously: Only 2% of all transactions cause a false alarm. In other words: ROSE does not stand in the way.

Challenges

Granularity. Coarser predictions are more precise.

Which is the most useful granularity?

Sequence rules. Infer over several transactions.

Programmers who changed X later changed Y.

Further data sources. Distinguish fixes from features

Access log messages, bug reports, . . .

Program analysis. Reduce noise by program analysis

What is coupling, anyway?

Best practices. Learn from earlier successes

How do we measure success?

Rationales. Present rules to programmers

How do we visualize complex rules?

Conclusion

- ROSE effectively guides users along related changes:
  - After an initial change, ROSE predicts 26% of further files and 15% of further entities
  - 30% of the suggested files and 26% of the suggested entities are correct predictions.
  - Warnings about missing changes are seldom, but reliable
- ROSE detects coupling between non-program entities (e.g. programs and documentation)
- Predictive power may increase further with log messages and bug reports being taken into account
- Research has just begun to exploit non-program artifacts

http://www.st.cs.uni-sb.de/