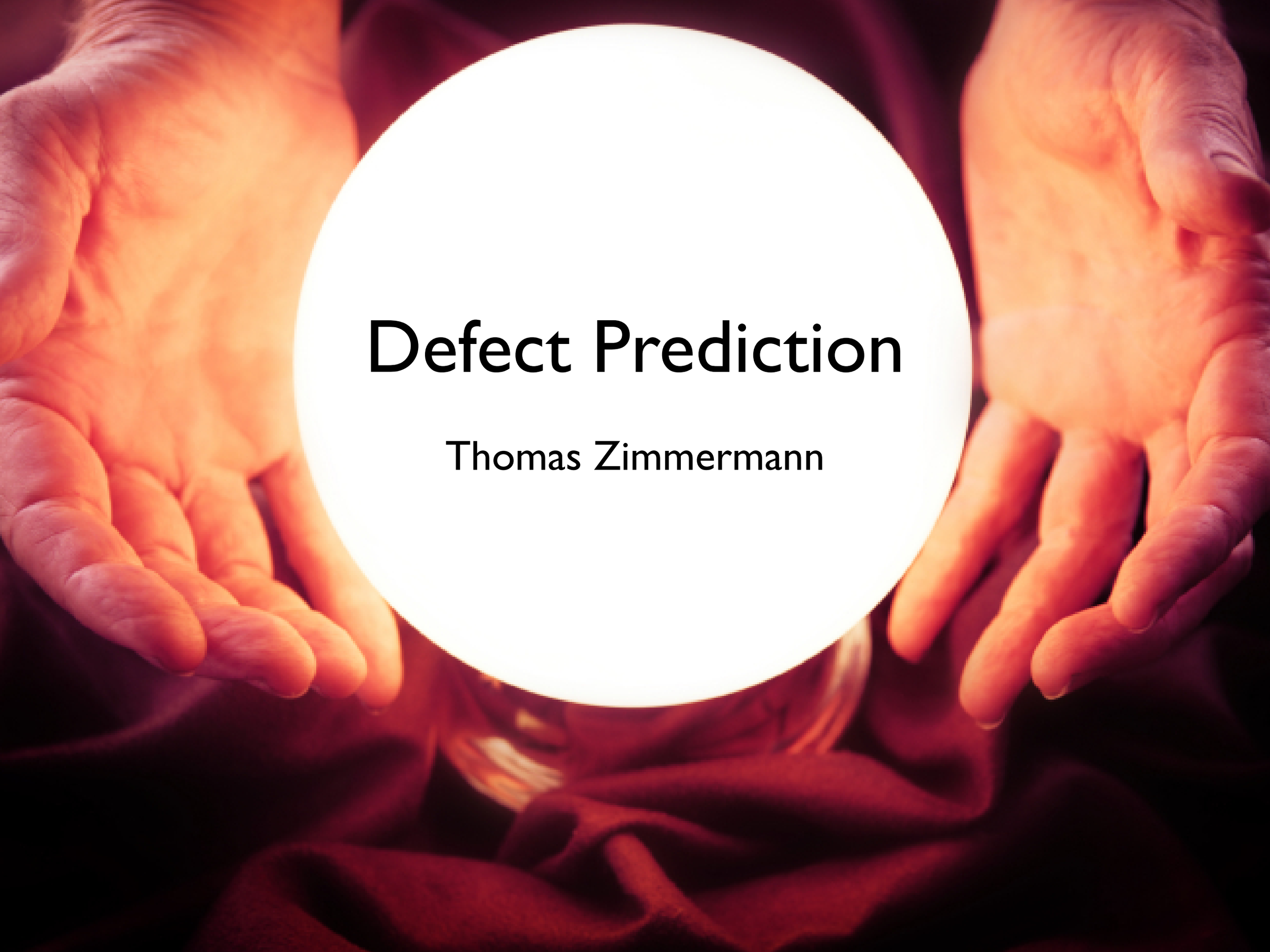


Defect Prediction

Thomas Zimmermann

Register for exam!

- Deadline for registration is TOMORROW! Tomorrow is June 22nd.
- You will need an RZ account and a TAN list. Get them in the Mensa building.
- To register log on to the following website: https://www.lsf.uni-saarland.de/

Tuesday are meetings

- 13:20 - 13:40 (TBA) Team QUALITY
- 13:40 - 14:00 (TBA) Team DUPLICATES
- 14:00 - 14:20 (Zeichensaal) Team VISUALIZE
- 14:20 - 14:20 (Zeichensaal) Team TRIAGE

Modules

Modules













Modules









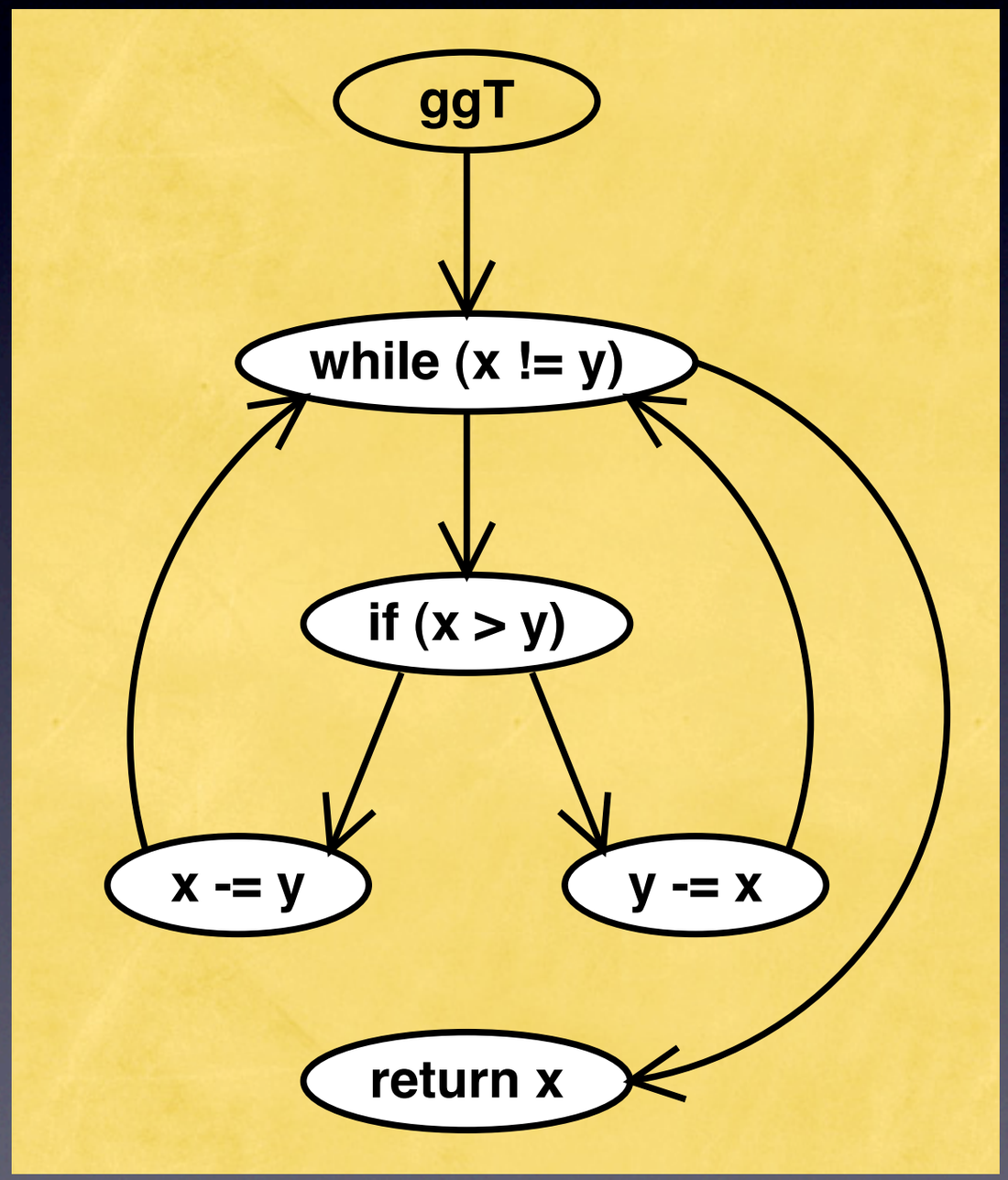

Which should you test most?

Complexity Metrics

- Measure "complexity" of the source code:
  - #Lines
  - #Classes
  - #Parameters
- Higher metric = greater complexity

McCabe Metrics

- Measure complexity of control flow
- V(G) = e – n + 2p
- e: #edges
- n: #statements
- p: #entries



Maintainability Index

Oman, P. & Hagemeister, J. "Constructing and Testing of Polynomials Predicting Software Maintainability."
Journal of Systems and Software 24, 3 (März 1994): 251–266.

Maintainability Index

Maintainability =

Oman, P. & Hagemeister, J. "Constructing and Testing of Polynomials Predicting Software Maintainability."
Journal of Systems and Software 24, 3 (März 1994): 251–266.

Maintainability Index

$$\text{Maintainability} = 171 - 5.2\ln(\overline{V}) - 0.23\overline{V(G)} - 16.2\ln(\overline{L})$$

Oman, P. & Hagemeister, J. "Constructing and Testing of Polynomials Predicting Software Maintainability."
Journal of Systems and Software 24, 3 (März 1994): 251–266.

Maintainability Index

Size of vocabulary

$$\text{Maintainability} = 171 - 5.2\ln(\overline{V}) - 0.23\overline{V(G)} - 16.2\ln(\overline{L})$$

Oman, P. & Hagemeister, J. "Constructing and Testing of Polynomials Predicting Software Maintainability."
Journal of Systems and Software 24, 3 (März 1994): 251–266.

Maintainability Index

Size of vocabulary

McCabe complexity

$$\text{Maintainability} = 171 - 5.2\ln(\overline{V}) - 0.23\overline{V(G)} - 16.2\ln(\overline{L})$$

Oman, P. & Hagemeister, J. "Constructing and Testing of Polynomials Predicting Software Maintainability." Journal of Systems and Software 24, 3 (März 1994): 251–266.

Maintainability Index

Size of vocabulary

McCabe complexity

$$\text{Maintainability} = 171 - 5.2\ln(\overline{V}) - 0.23\overline{V(G)} - 16.2\ln(\overline{L})$$

code lines

Oman, P. & Hagemeister, J. "Constructing and Testing of Polynomials Predicting Software Maintainability."
Journal of Systems and Software 24, 3 (März 1994): 251–266.

Maintainability Index

Size of vocabulary

McCabe complexity

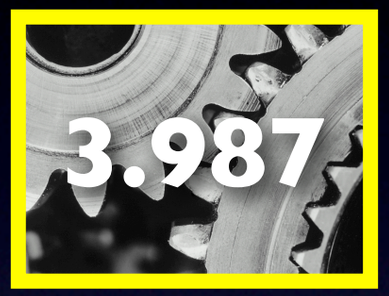
$$\text{Maintainability} = 171 - 5.2\ln(\overline{V}) - 0.23\overline{V(G)} - 16.2\ln(\overline{L}) + 50\sin\left(\sqrt{2.4\overline{C}}\right)$$

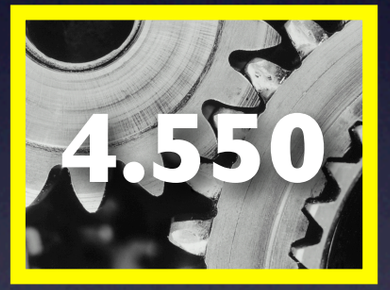
Percentage of comment lines

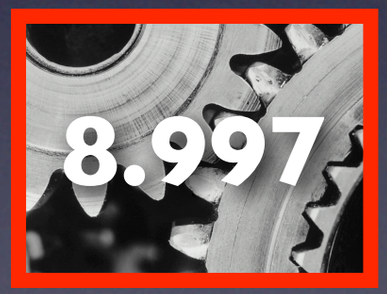
code lines

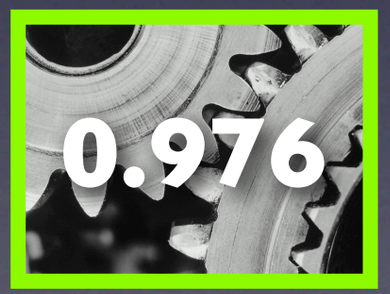
Oman, P. & Hagemeister, J. "Constructing and Testing of Polynomials Predicting Software Maintainability." Journal of Systems and Software 24, 3 (März 1994): 251–266.

Complexity Metrics









Problem Report for Keynote

Problem and system information:

```
Date/Time:      2006-03-07 23:35:25.516 +0100
OS Version:     10.4.5 (Build 8H14)
Report Version: 4

Command: Keynote
Path:    /Applications/iWork '06/Keynote.app/Contents/MacOS/Keynote
Parent:  WindowServer [79]

Version:        3.0.0 (423)
Build Version:  1
Project Name:   iWork
```

Please describe what you were doing when the problem happened:

Your report will help Apple improve this software. Your personal information is not sent with this report. You will not be contacted in response to this report. For Apple product support, visit www.apple.com/support or contact your Apple dealer.

?

Send to Apple...

Problem Report for Keynote

Problem and system information:

```
Date/Time:      2006-03-07 23:35:25.516 +0100
OS Version:     10.4.5 (Build 8H14)
Report Version: 4

Command: Keynote
Path:    /Applications/iWork '06/Keynote.app/Contents/MacOS/Keynote
Parent:  WindowServer [79]

Version:        3.0.0 (423)
Build Version:  1
Project Name:   iWork
```

Please describe what you were doing when the problem happened:

Your report will help Apple improve this software. Your personal information is not sent with this report. You will not be contacted in response to this report. For Apple product support, visit www.apple.com/support or contact your Apple dealer.

?

Send to Apple...

Problem Report for Keynote

Problem and system information:

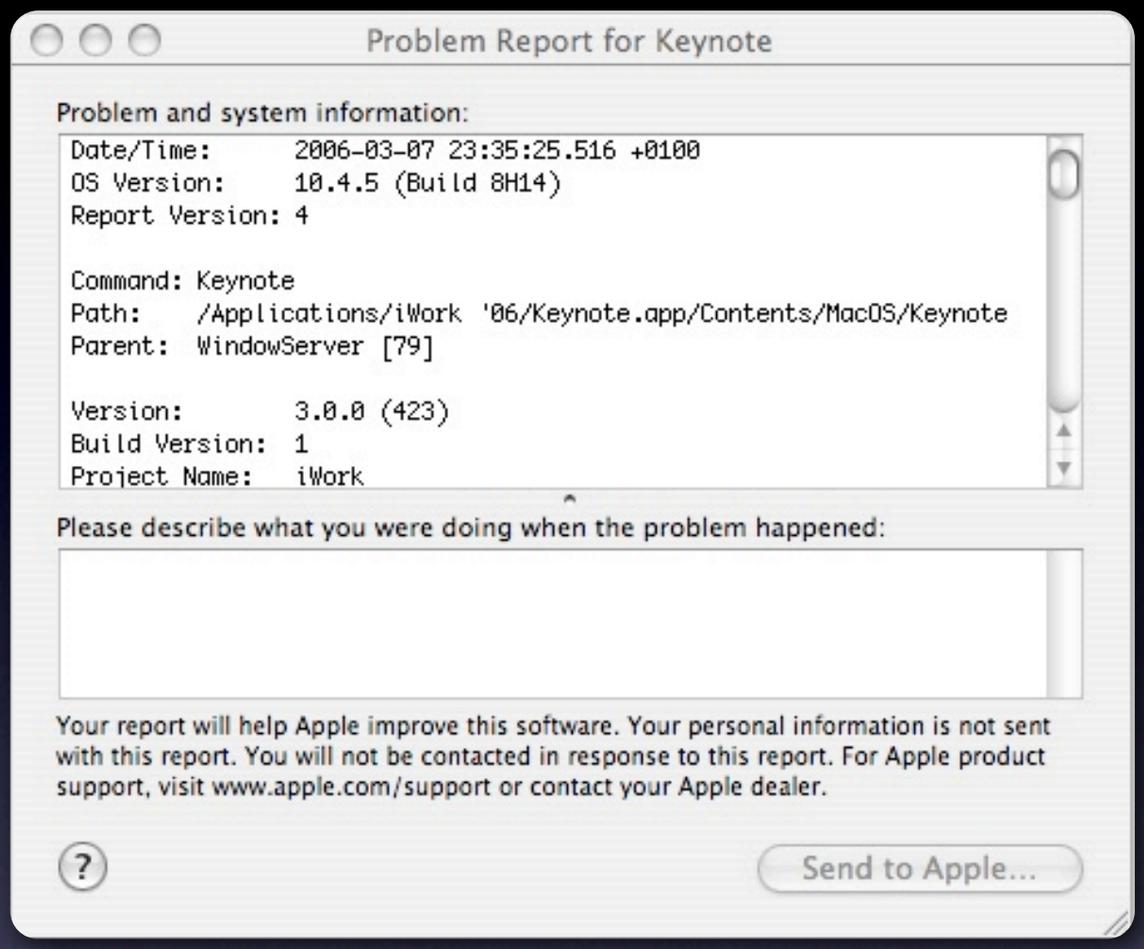
```
Date/Time:      2006-03-07 23:35:25.516 +0100
OS Version:     10.4.5 (Build 8H14)
Report Version: 4

Command: Keynote
Path:    /Applications/iWork '06/Keynote.app/Contents/MacOS/Keynote
Parent:  WindowServer [79]

Version:        3.0.0 (423)
Build Version:  1
Project Name:   iWork
```

Please describe what you were doing when the problem happened:

Your report will help Apple improve this software. Your personal information is not sent with this report. You will not be contacted in response to this report. For Apple product support, visit www.apple.com/support or contact your Apple dealer.

?

Send to Apple...



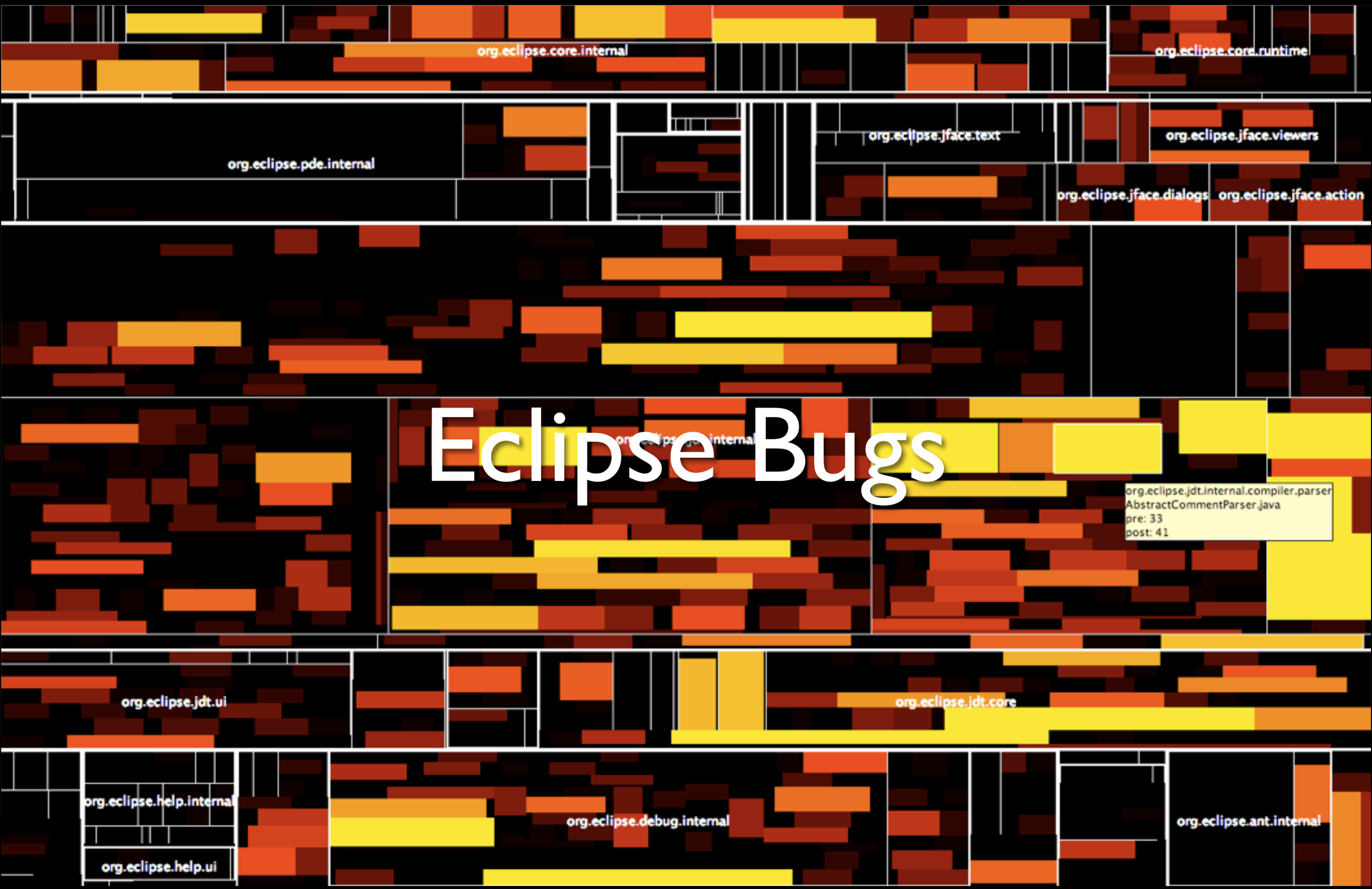
Bugs

Changes


Bugs
Changes

Map bugs to *code locations*

Eclipse Bugs



Past Defects













Past Defects







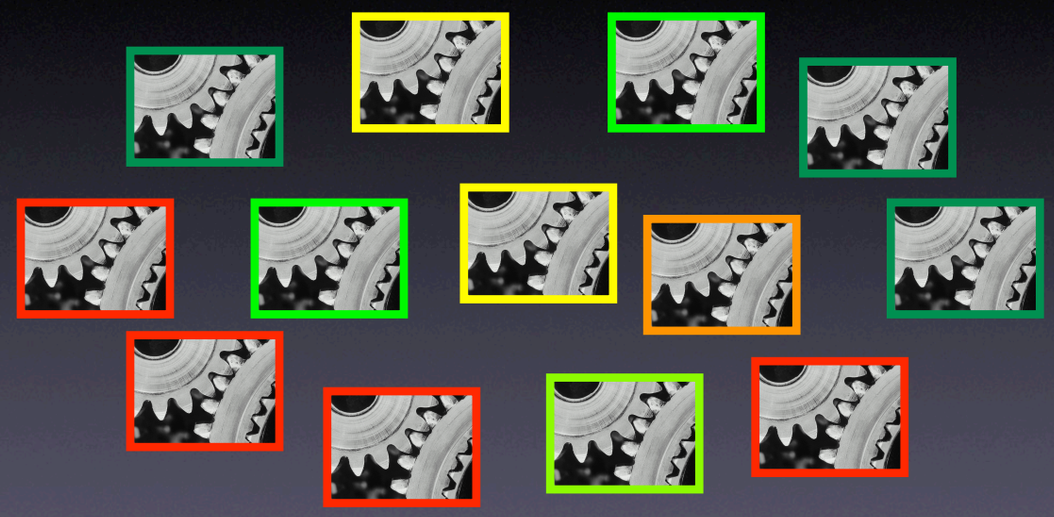






What can we use to predict defects?

What can we use to predict defects?

Past Failures	Complexity Metrics
	
if we have some if we can afford them	if we know which ones to choose

Combined approach

Combined approach

Past defects

Combined approach

Past defects



- Collect *failures* occurring in the field within 6 months after release
- Map failures back to *fixes* and thus *defects* in *modules (binaries)*
- We can tell how failure-prone a module is

Combined approach

Past defects

Metrics

Combined approach

Past defects



Metrics

#Lines	#Vars	#Classes
#Params	#Reads	#Writes
#Arcs	#Blocks	McCabe
Fan In	Fan Out	…

Combined approach

Past defects



Metrics

#Lines	#Vars	#Classes
#Params	#Reads	#Writes
#Arcs	#Blocks	McCabe
Fan In	Fan Out	…

Combined approach

Past defects

Metrics

#Lines	#Vars	#Classes
#Params	#Reads	#Writes
#Arcs	#Blocks	McCabe
Fan In	Fan Out	…

#Lines	#Vars	#Classes
#Params	#Reads	#Writes
#Arcs	#Blocks	McCabe
Fan In	Fan Out	…

Combined approach

Past defects

Metrics

#Lines	#Vars	#Classes
#Params	#Reads	#Writes
#Arcs	#Blocks	McCabe
Fan In	Fan Out	…

#Lines	#Vars	#Classes
#Params	#Reads	#Writes
#Arcs	#Blocks	McCabe
Fan In	Fan Out	…

#Lines	#Vars	#Classes
#Params	#Reads	#Writes
#Arcs	#Blocks	McCabe
Fan In	Fan Out	…

Combined approach

Past defects

Metrics

#Lines	#Vars	#Classes
#Params	#Reads	#Writes
#Arcs	#Blocks	McCabe
Fan In	Fan Out	…

#Lines	#Vars	#Classes
#Params	#Reads	#Writes
#Arcs	#Blocks	McCabe
Fan In	Fan Out	…

#Lines	#Vars	#Classes
#Params	#Reads	#Writes
#Arcs	#Blocks	McCabe
Fan In	Fan Out	…

Correlation

Combined approach

Past defects



Metrics

#Lines	#Vars	#Classes
#Params	#Reads	#Writes
#Arcs	#Blocks	McCabe
Fan In	Fan Out	…

#Lines	#Vars	#Classes
#Params	#Reads	#Writes
#Arcs	#Blocks	McCabe
Fan In	Fan Out	…

#Lines	#Vars	#Classes
#Params	#Reads	#Writes
#Arcs	#Blocks	McCabe
Fan In	Fan Out	…

Correlation

McCabe ~ Defects

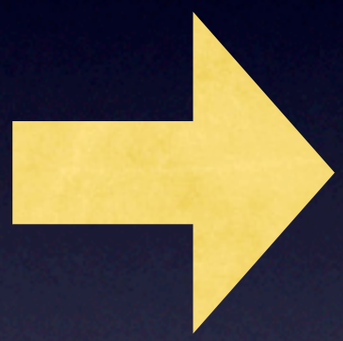
Predicting defects

Correlation McCabe ~ Defects

Predicting defects

Correlation | McCabe | ~ | Defects |

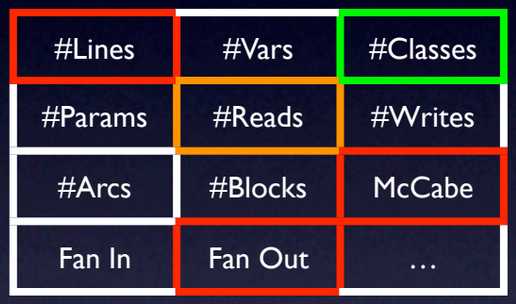
Predicting defects



#Lines	#Vars	#Classes
#Params	#Reads	#Writes
#Arcs	#Blocks	McCabe
Fan In	Fan Out	…

Correlation McCabe ~ Defects

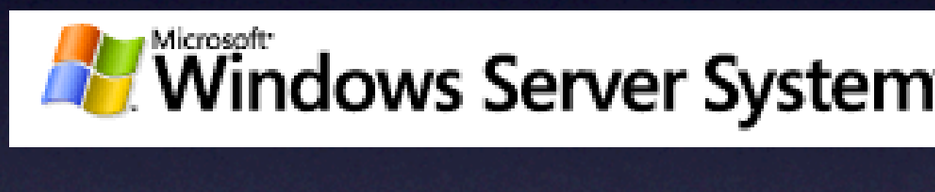
Predicting defects



Correlation McCabe ~ Defects

Projects Researched

- Internet Explorer 6
- IIS Server



- Windows Process Messaging
- DirectX

- NetMeeting

>1,000,000 Lines of Code

Projects Researched

A B C D E

Do metrics correlate with defects?

Do metrics correlate with defects?

Project	Metrics correlated w/ defects

Do metrics correlate with defects?

Project	Metrics correlated w/ defects
A	*#Clas*ses and 5 derived

Do metrics correlate with defects?

Project	Metrics correlated w/ defects
A	*#Clas*ses and 5 derived
B	almost all

Do metrics correlate with defects?

Project	Metrics correlated w/ defects
A	*#Classes* and 5 derived
B	almost all
C	all except *MaxInheritanceDepth*

Do metrics correlate with defects?

Project	Metrics correlated w/ defects
A	*#Classes* and 5 derived
B	almost all
C	all except *MaxInheritanceDepth*
D	only *#Lines*

Do metrics correlate with defects?

Project	Metrics correlated w/ defects
A	*#Classes* and 5 derived
B	almost all
C	all except *MaxInheritanceDepth*
D	only *#Lines*
E	*#Functions, #Arcs, Complexity*

Do metrics correlate with defects?

Project	Metrics correlated w/ defects
	#C es a d
B	ost all
C	l except *InheritanceDepth*
D	nly *#Line*
E	*#Functions, #Arcs, Complexity*

YES

Is there a set of metrics that fits all projects?

Is there a set of metrics that fits all projects?

Project	Metrics correlated w/ defects

Is there a set of metrics that fits all projects?

Project	Metrics correlated w/ defects
A	*#Clas*ses and 5 derived

Is there a set of metrics that fits all projects?

Project	Metrics correlated w/ defects
A	*#Clas*ses and 5 derived
B	almost all

Is there a set of metrics that fits all projects?

Project	Metrics correlated w/ defects
A	*#Classes* and 5 derived
B	almost all
C	all except *MaxInheritanceDepth*

Is there a set of metrics that fits all projects?

Project	Metrics correlated w/ defects
A	*#Classes* and 5 derived
B	almost all
C	all except *MaxInheritanceDepth*
D	only *#Lines*

Is there a set of metrics that fits all projects?

Project	Metrics correlated w/ defects
A	*#Classes* and 5 derived
B	almost all
C	all except *MaxInheritanceDepth*
D	only *#Lines*
E	*#Functions, #Arcs, Complexity*

Is there a set of metrics that fits all projects?

Project	Metrics correlated w/ defects
A	*#Classes*, 5 der…
B	…st a…
C	…all …, *MaxIn…anceDe…*
D	…only *#D…*
E	*#Functions, #Arcs, Complexity*

NO

Can we predict defect-prone modules?

Can we predict defect-prone modules?

- Basic idea: *Combine metrics*

Can we predict defect-prone modules?

- Basic idea: *Combine metrics*
- Give most weight to *most predictive metrics*

Can we predict defect-prone modules?

- Basic idea: *Combine metrics*
- Give most weight to *most predictive metrics*
- Problem: Metrics are *intercorrelated*

Can we predict defect-prone modules?

- Basic idea: *Combine metrics*
- Give most weight to *most predictive metrics*
- Problem: Metrics are *intercorrelated*
- Solution: Principal Component Analysis (PCA)

A Ranking

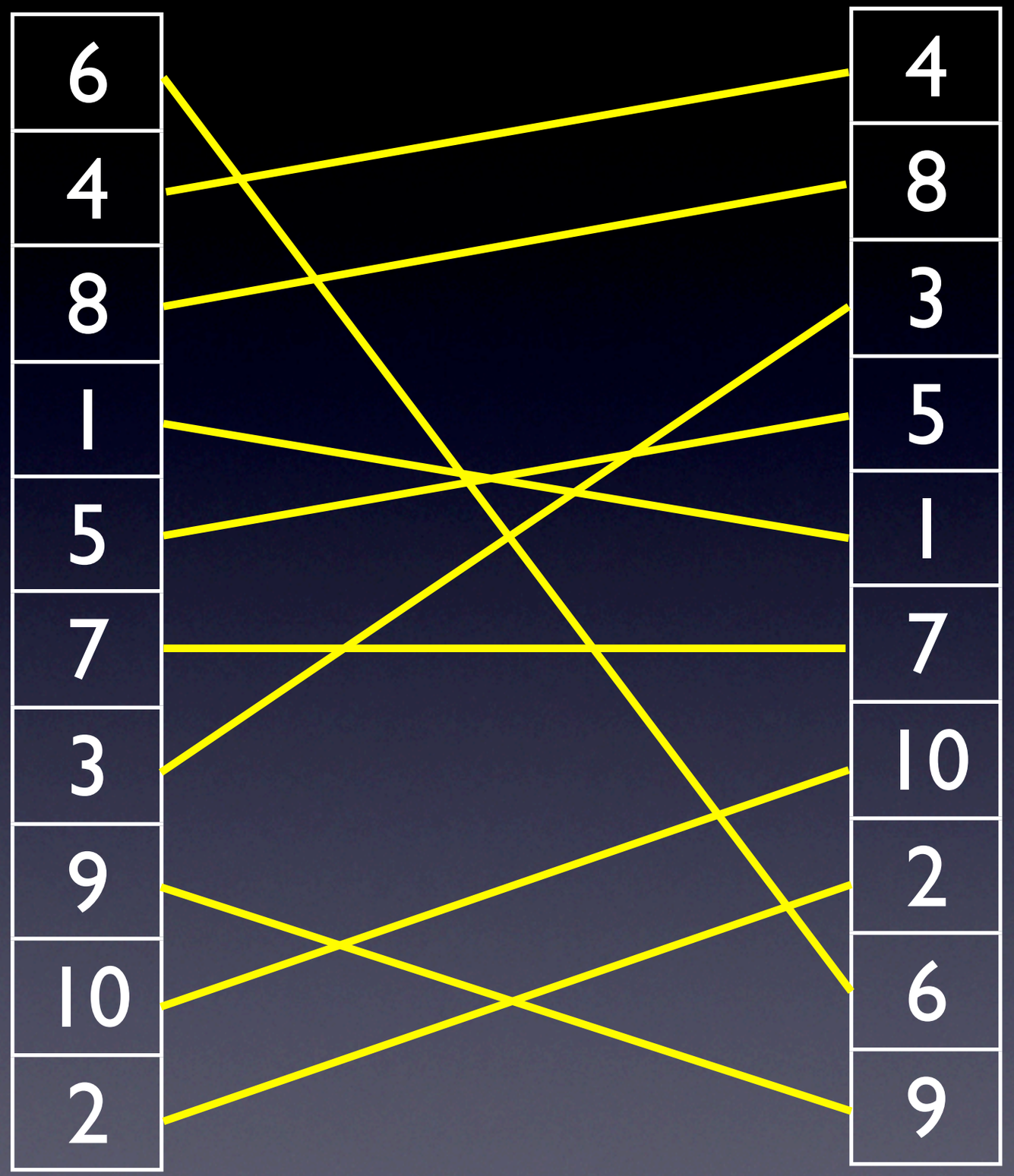
- 1/3 of the modules
- ranked according to predictor built from 2/3 of the modules
- can be *evaluated* against actual ranking

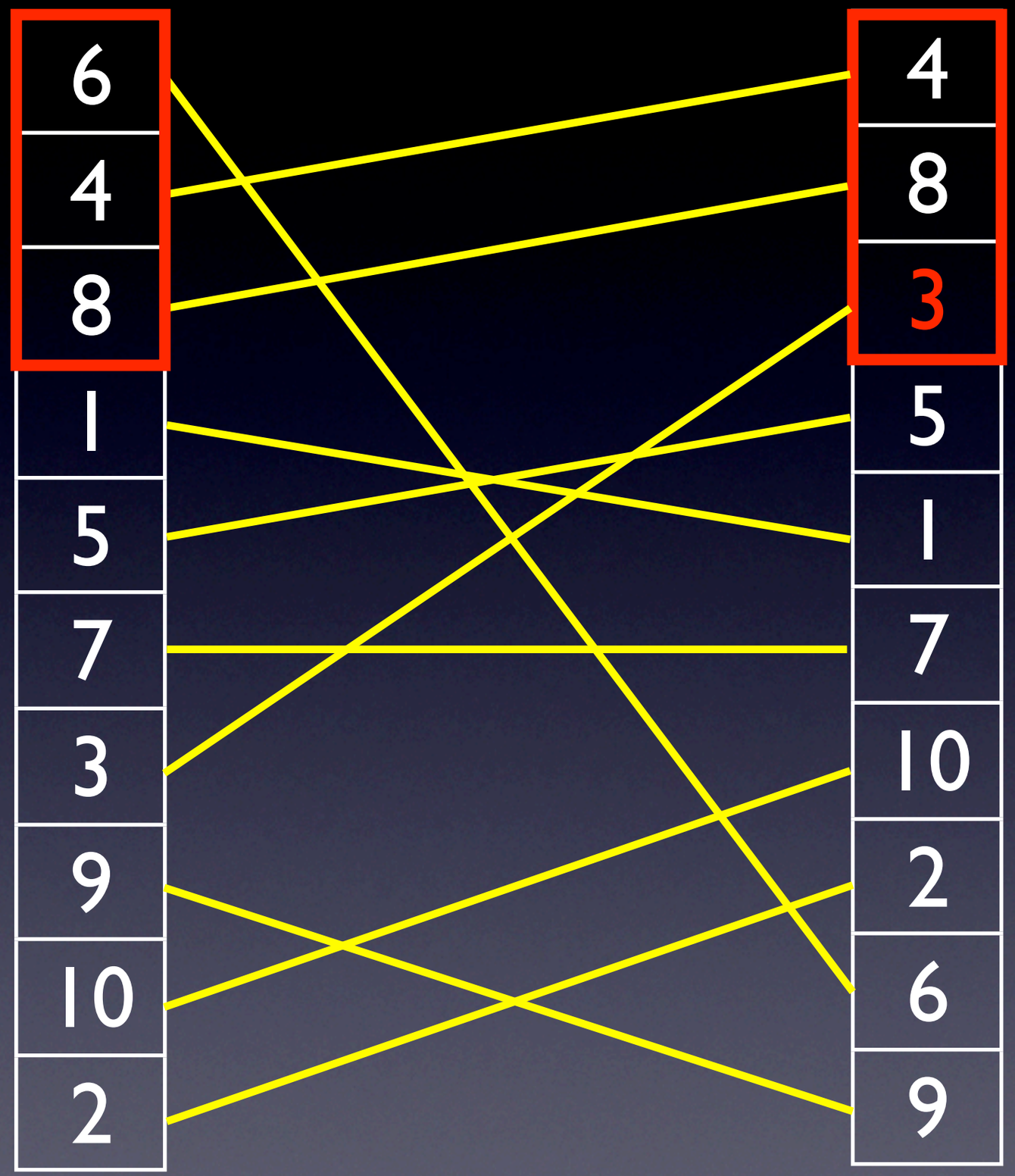
A Ranking

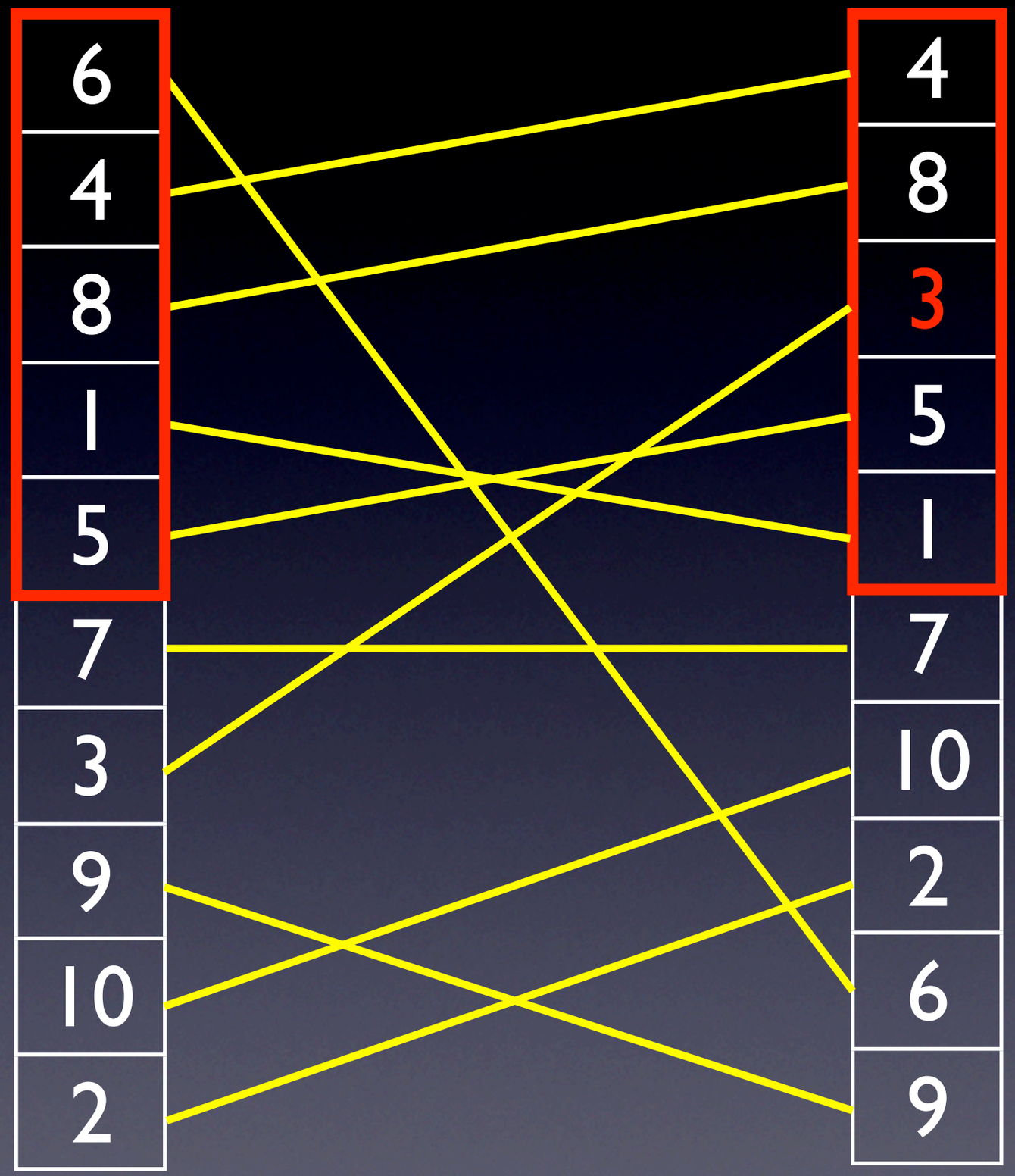
- 1/3 of the modules
- ranked according to predictor built from 2/3 of the modules
- can be *evaluated* against actual ranking

predicted
6
4
8
1
5
7
3
9
10
2

actual
4
8
3
5
1
7
10
2
6
9


6
4
8
1
5
7
3
9
10
2
4
8
3
5
1
7
10
2
6
9
predicted
actual


6
4
8
1
5
7
3
9
10
2
4
8
3
5
1
7
10
2
6
9
predicted
actual


6
4
8
1
5
7
3
9
10
2
4
8
3
5
1
7
10
2
6
9
predicted
actual

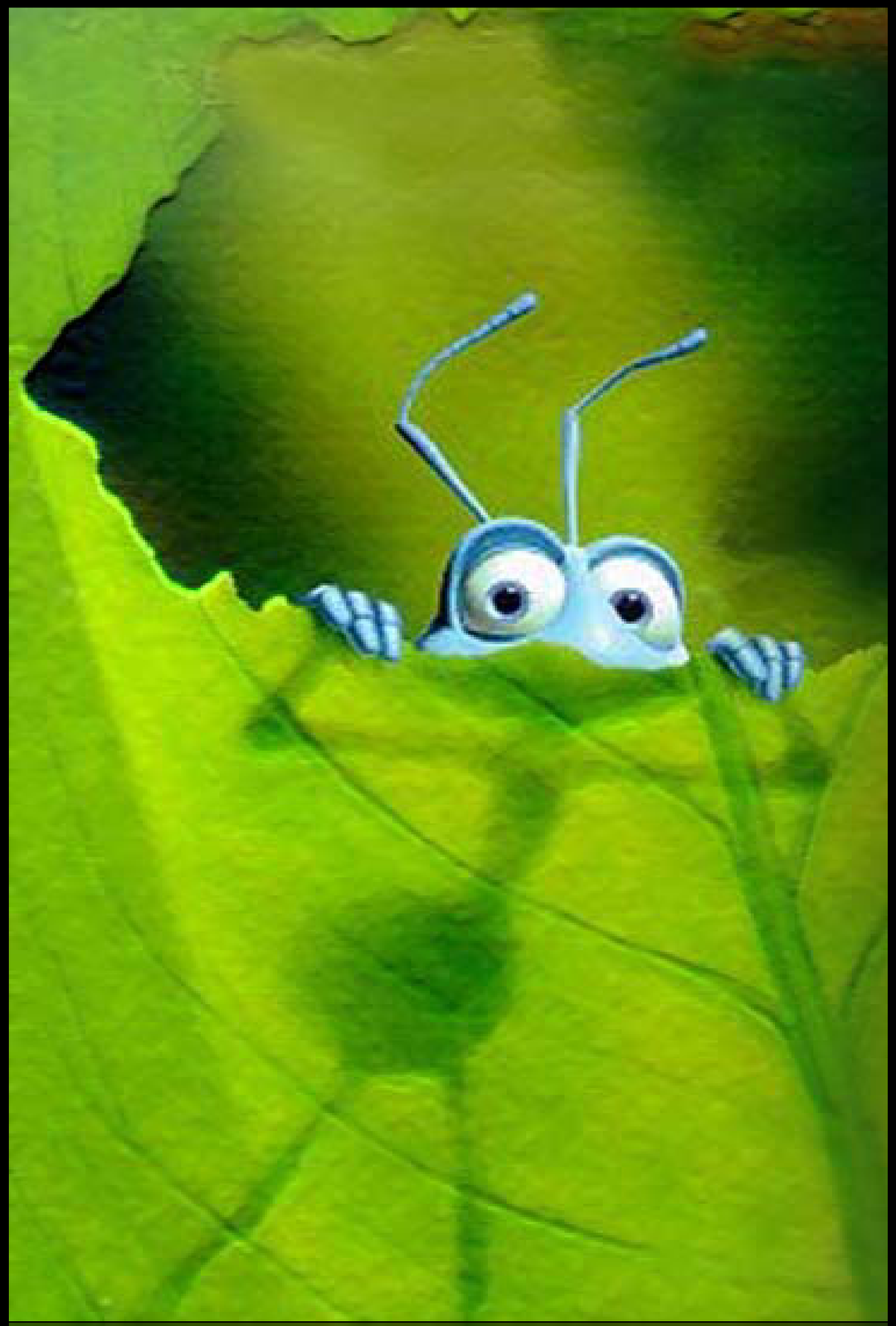
Can we predict defect-prone modules?

Project	#Components	R^2 value
A	9	0.741
B	6	0.779
C	7	0.579
D	7	0.684
E	5	0.919

Can we predict defect-prone modules?

Project	#Components	R^2 value
A	9	741
B	6	0.779
C	7	579
D	7	.684
E	5	0.919

YES



BugCache

Predicting Defects

(ASE 2006, ICSE 2007)

Sung Kim • MIT

Tom Zimmermann • Saarland University

Jim Whitehead • Univ. of California SC

Andreas Zeller • Saarland University

The Problem


How should we allocate our resources for quality assurance?

Localities

- Temporal locality
  Hypothesis: Defects occur in bursts → Cache model
- Spatial locality
  Hypothesis: Entities that are near defect-prone elements are likely to have defects as well
- Changed-entities locality
  Code churn (Nagappan et al)
  Most recently modified (Hassan et al.)
- New-entities locality
  New entities likely contain defects (Graves et al.)

One Solution

List with elements that
(will) have defects

List is adaptive, i.e.,
it changes over time

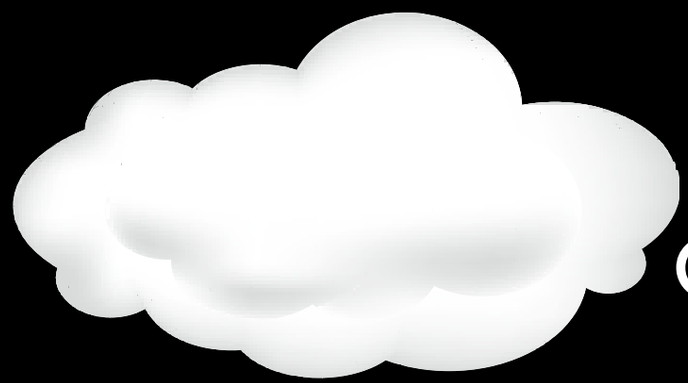
One Solution

List with elements that
(will) have defects

Cache

List is adaptive, i.e.,
it changes over time

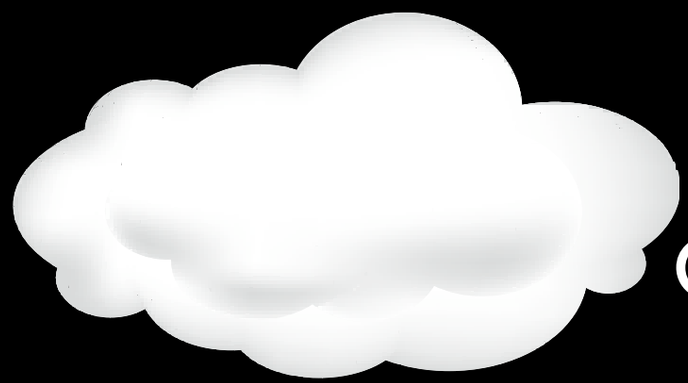
The BugCache Model



Cache size: 2

Hypothesis: Temporal locality between defects

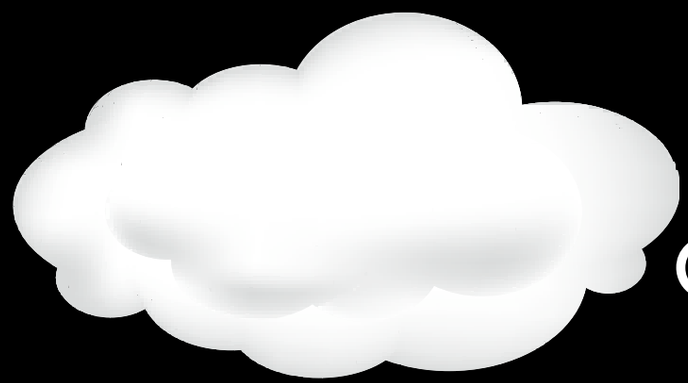
The BugCache Model



Cache size: 2

Hypothesis: Temporal locality between defects

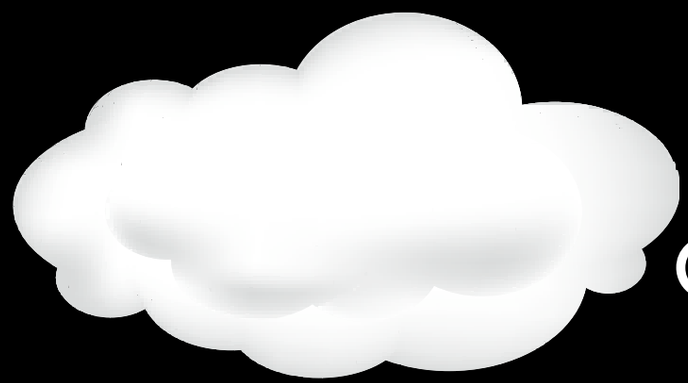
The BugCache Model



Cache size: 2

Hypothesis: Temporal locality between defects

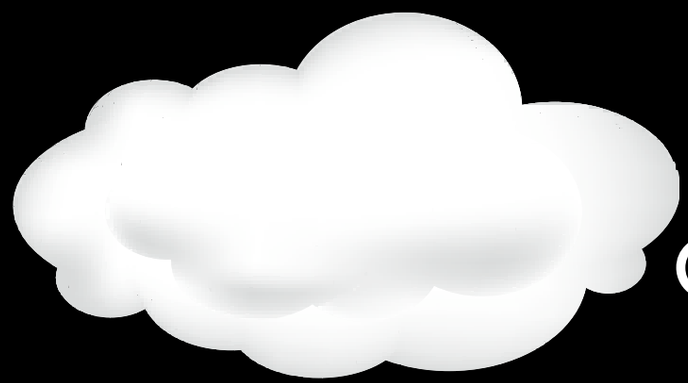
The BugCache Model



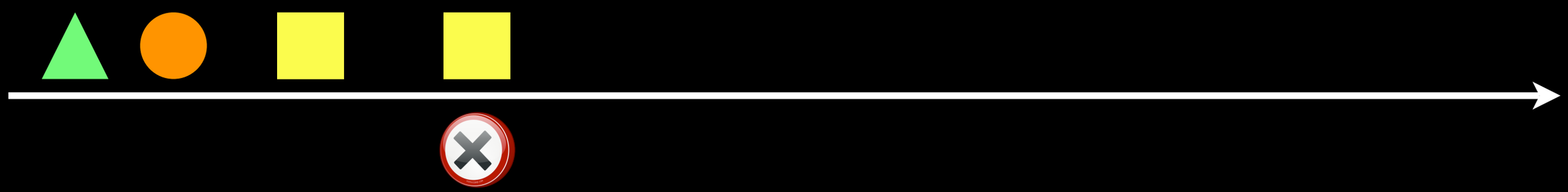
Cache size: 2

Hypothesis: Temporal locality between defects

The BugCache Model

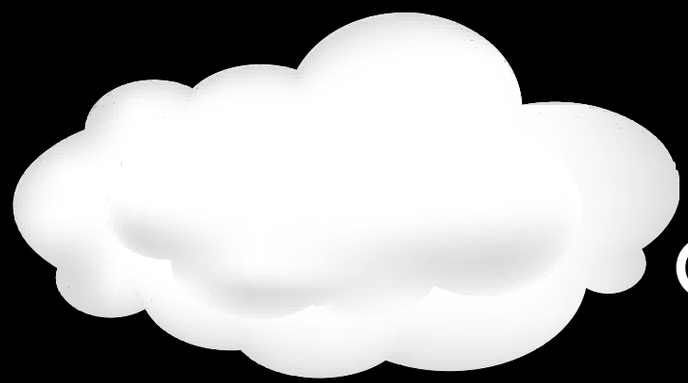


Cache size: 2



Hypothesis: Temporal locality between defects

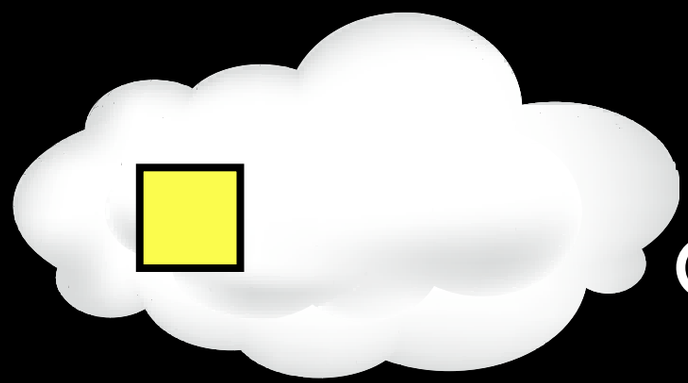
The BugCache Model



Cache size: 2

Hypothesis: Temporal locality between defects

The BugCache Model



Hypothesis: Temporal locality between defects

Bug-introducing Changes

BUG-INTRODUCING

```
...
if (foo==null) {
    foo.bar();
...
```

Bug-introducing Changes

BUG-INTRODUCING

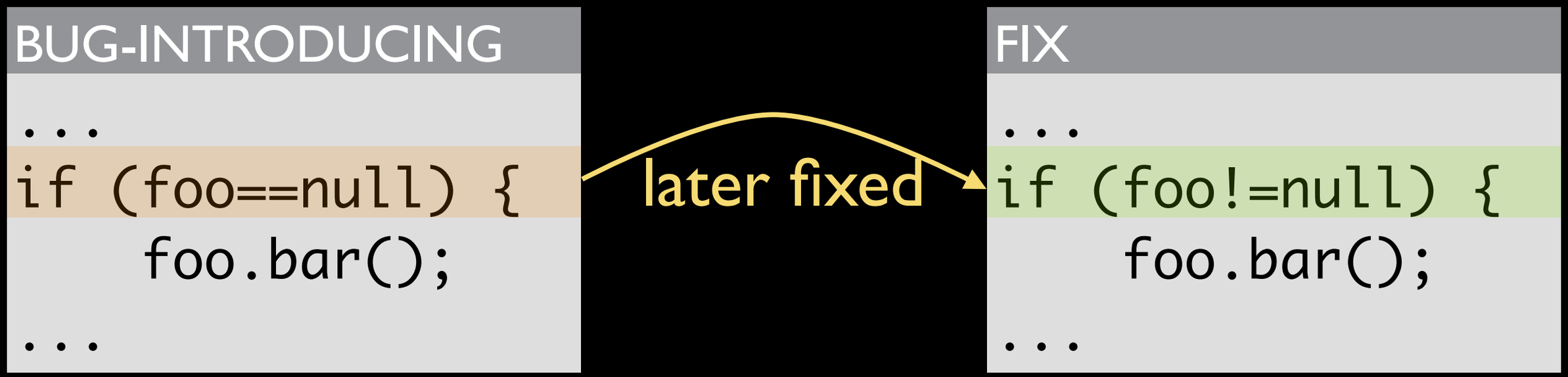
```
...
if (foo==null) {
    foo.bar();
...
```

later fixed

FIX

```
...
if (foo!=null) {
    foo.bar();
...
```

Bug-introducing Changes



Changes that lead to problems
as indicated by later fixes.

Cache Update

Load missed and nearby elements (spatial locality)

Cache Update

Load missed and nearby elements (spatial locality)

Entity	Number of common changes with ■
▲	1
●	4
◆	0

Cache Update

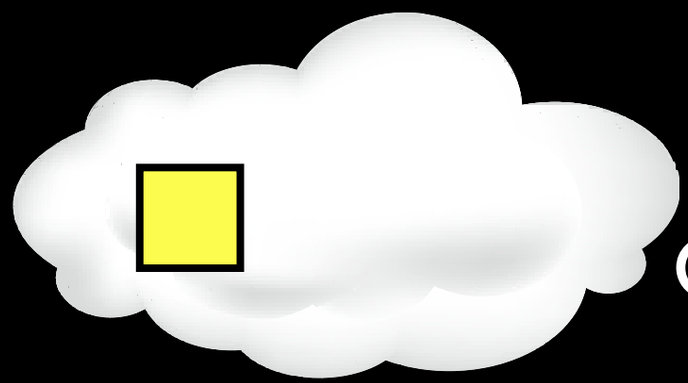
Load missed and nearby elements (spatial locality)

Entity	Number of common changes with ■
▲	1
●	4
◆	0

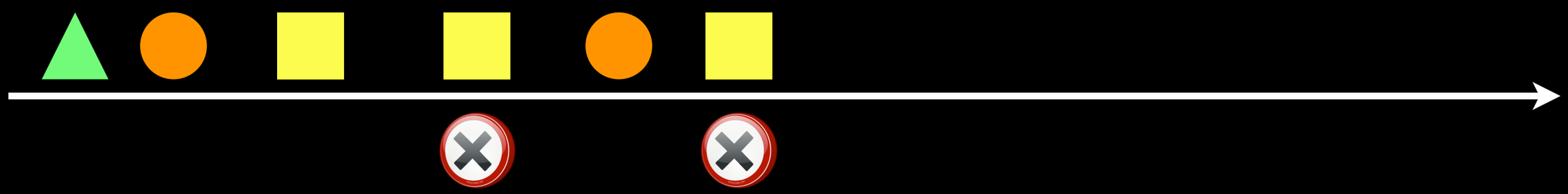
Parameter: Block size

The BugCache Model

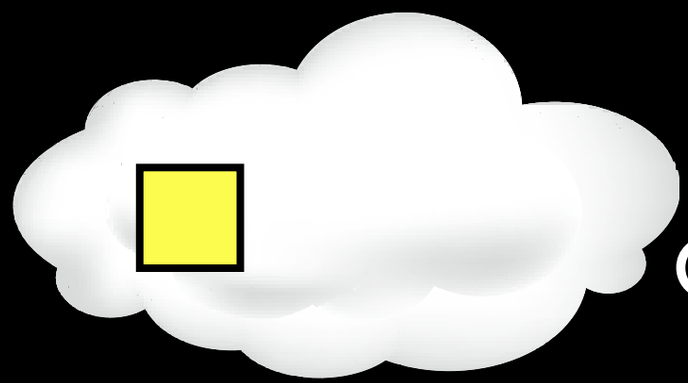
The BugCache Model



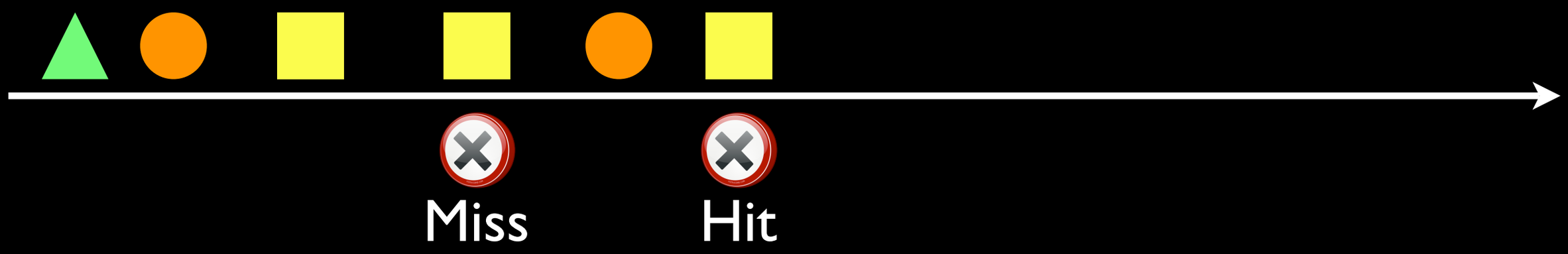
Cache size: 2



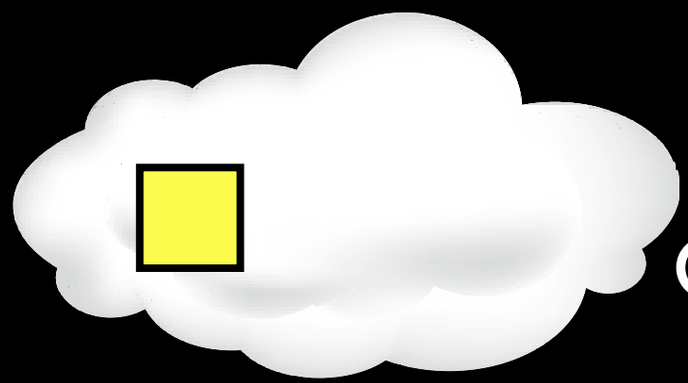
The BugCache Model



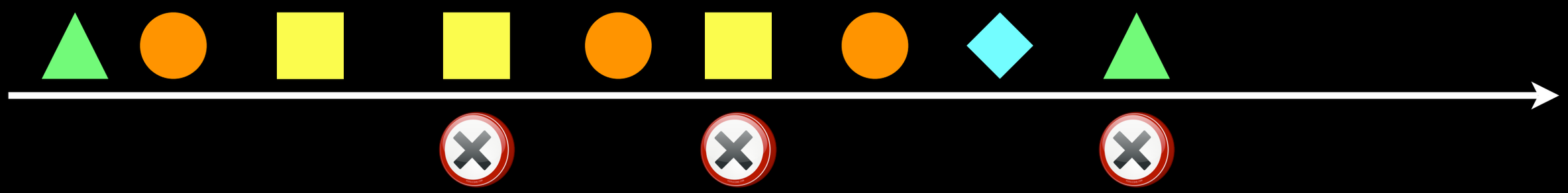
Cache size: 2



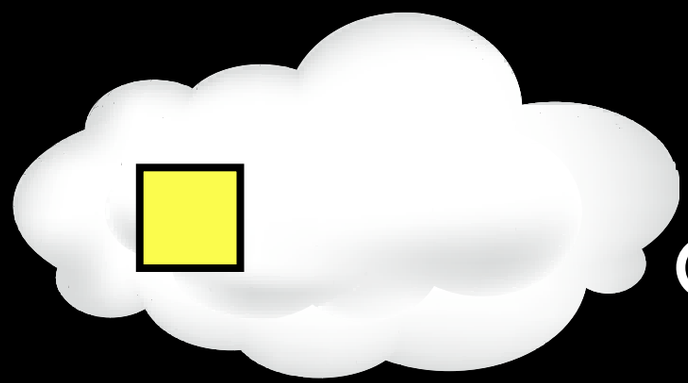
The BugCache Model



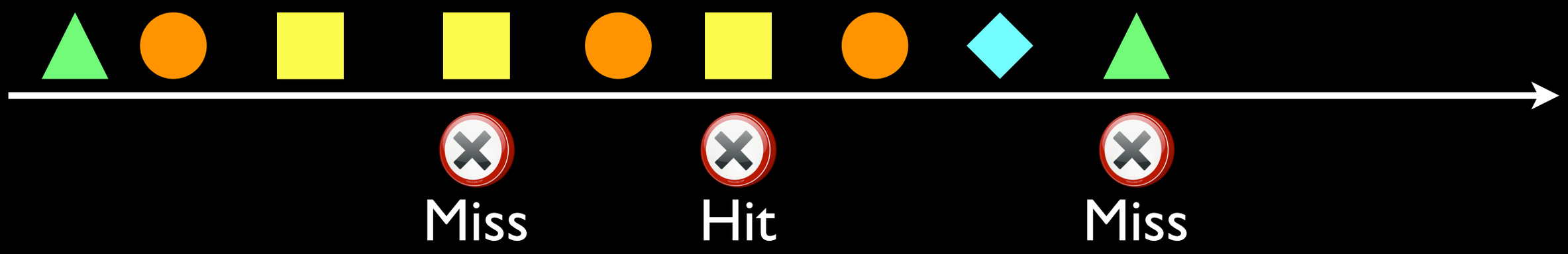
Cache size: 2



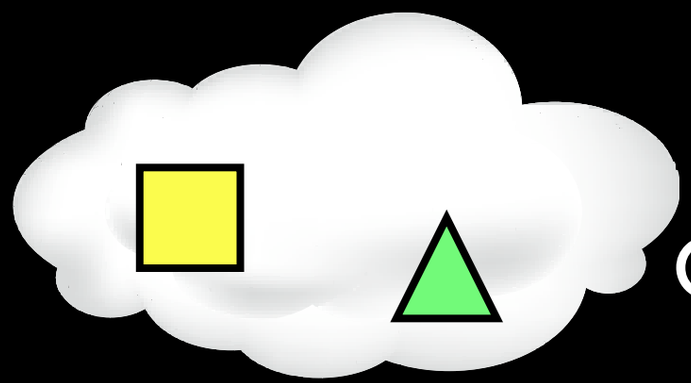
The BugCache Model



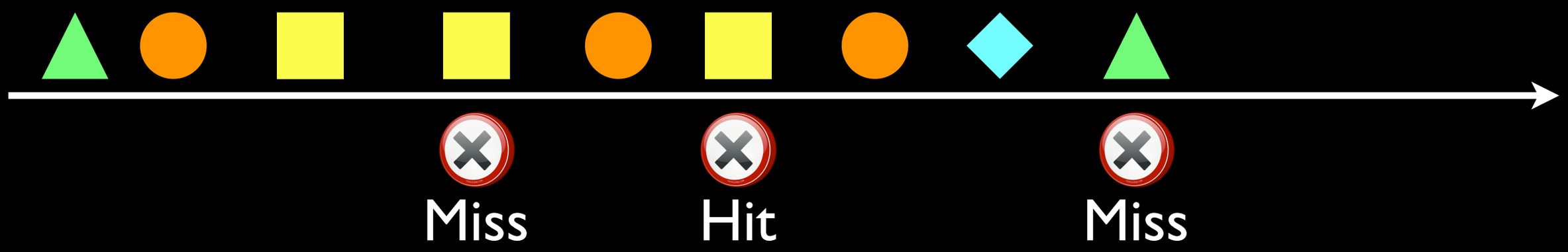
Cache size: 2



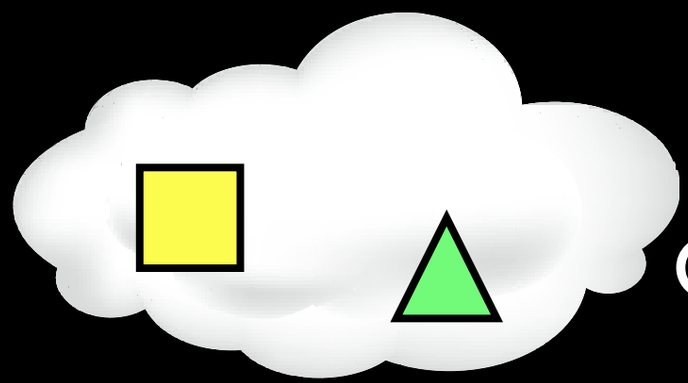
The BugCache Model



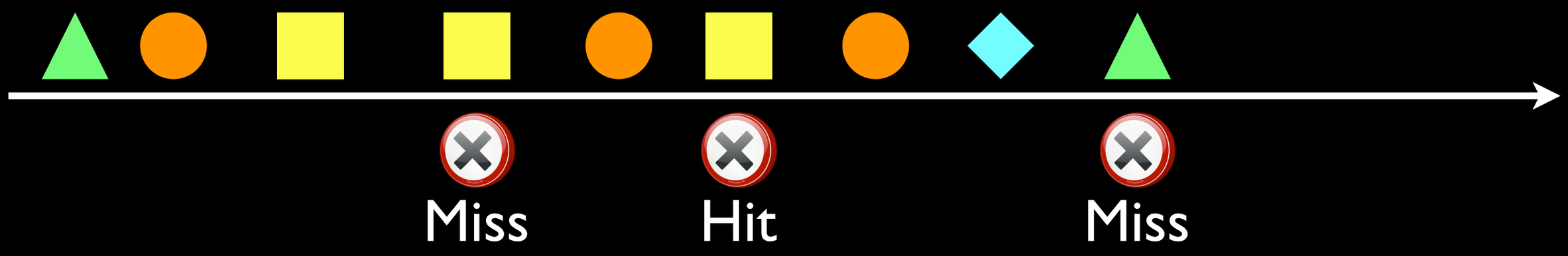
Cache size: 2



The BugCache Model



Cache size: 2



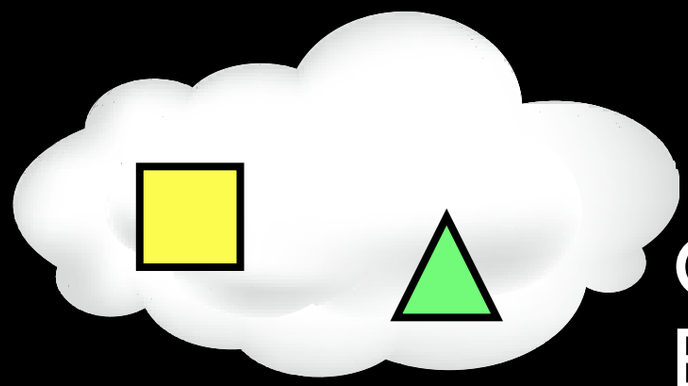
Hit rate = #Hits / #Defects = 33.3%

Replacement Policies

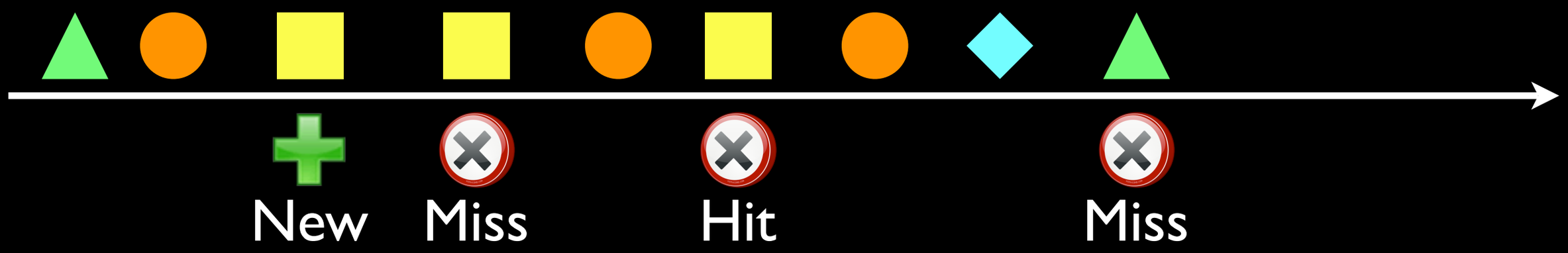
- Least recently used (LRU)
  Unload the entities that have the least recently found defect.
- Least frequently changed (CHANGE)
  Unload the entities that have the least number of changes.
- Least frequent defects (BUG)
  Unload the entities that have the least number of defects.

Parameter: Replacement Policy

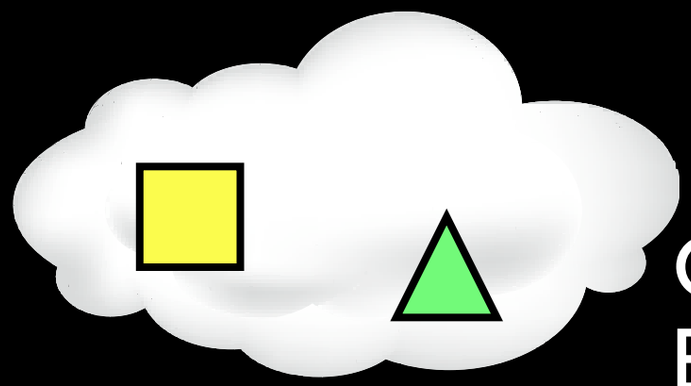
The BugCache Model



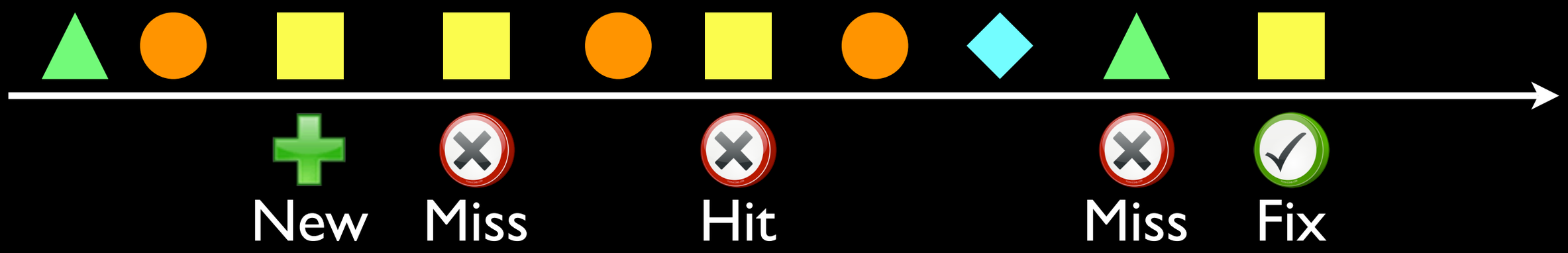
Cache size: 2
Block size: 1



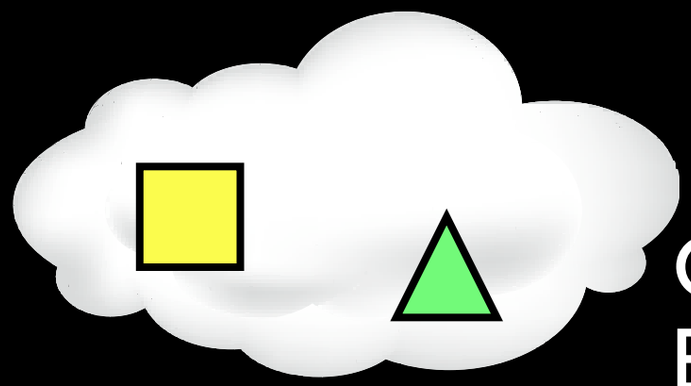
The BugCache Model



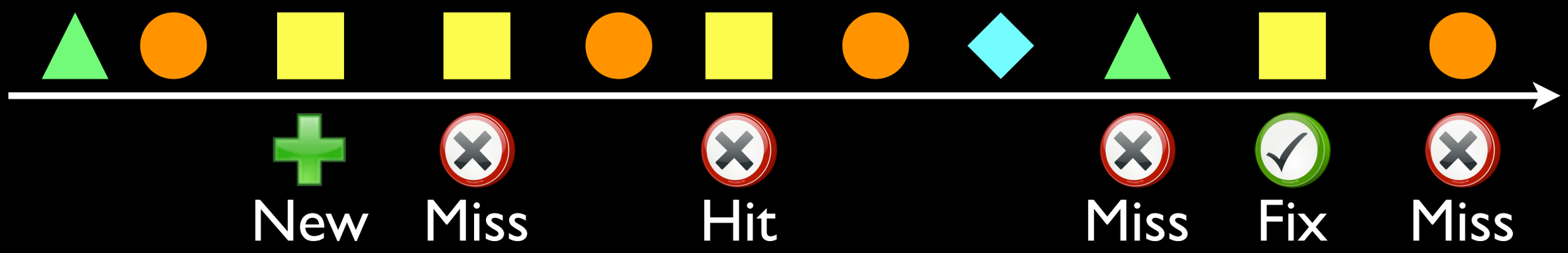
Cache size: 2
Block size: 1



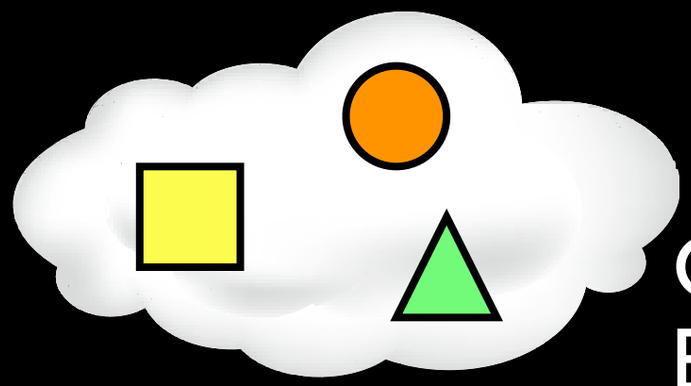
The BugCache Model



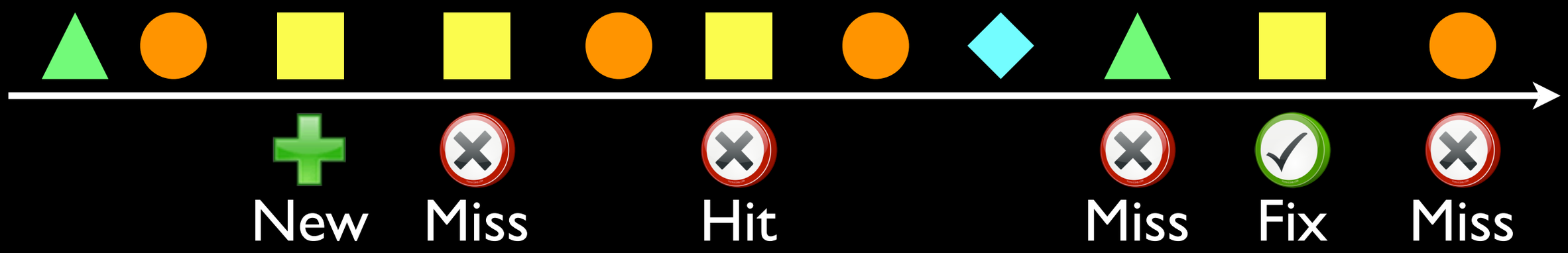
Cache size: 2
Block size: 1



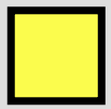
The BugCache Model

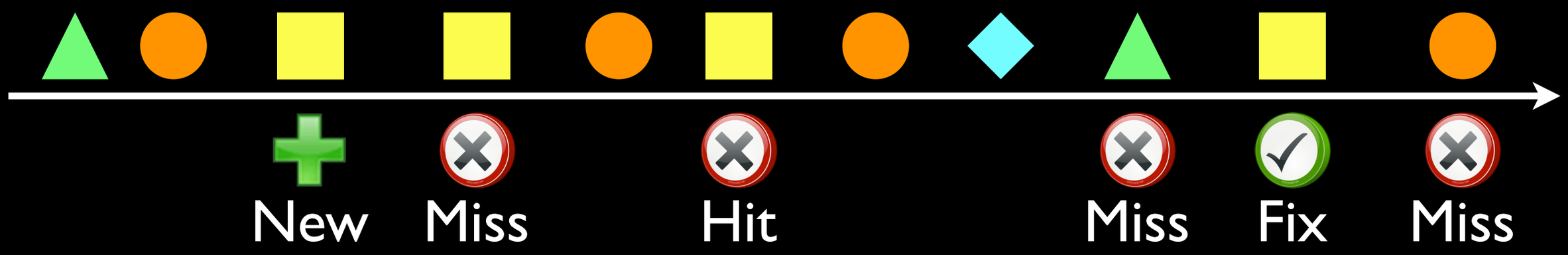


Cache size: 2
Block size: 1

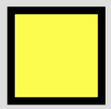


The BugCache Model

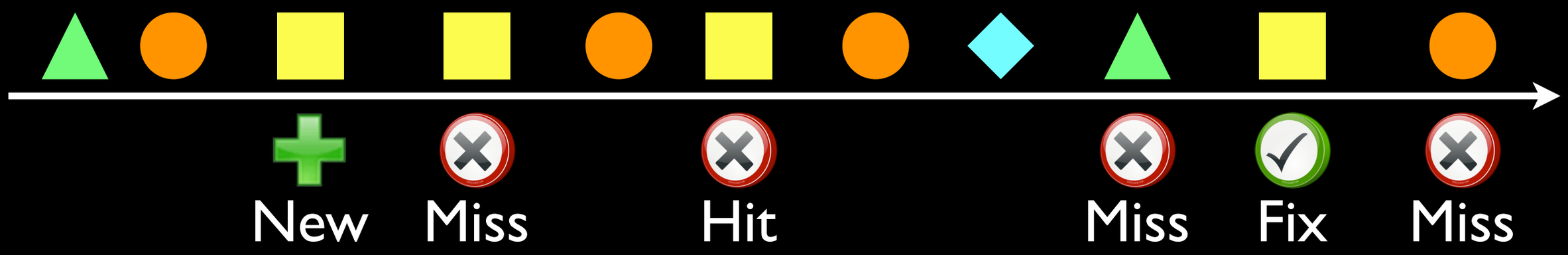
Entity	Last change	#Changes	#Bugs
	–1	4	2
	–2	2	1



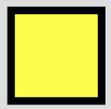
The BugCache Model

Entity	Last change	#Changes	#Bugs
	–1	4	2
	–2	2	1

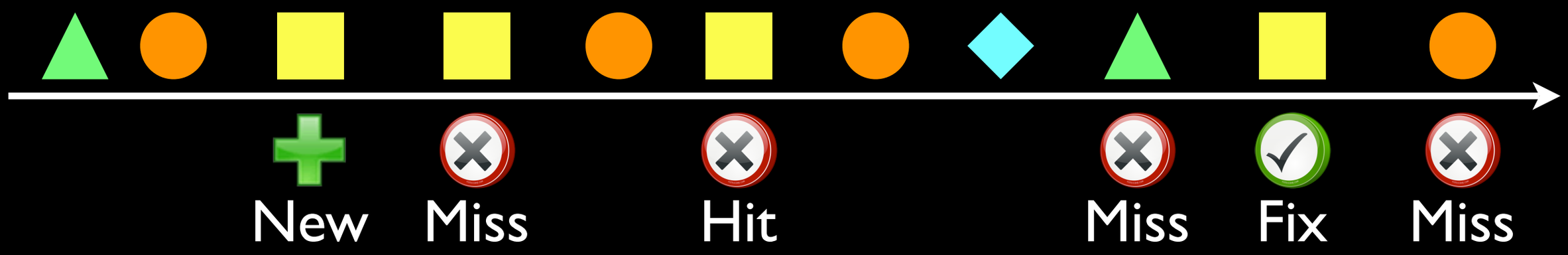
LRU



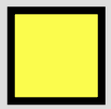
The BugCache Model

Entity	Last change	#Changes	#Bugs
	–1	4	2
	–2	2	1

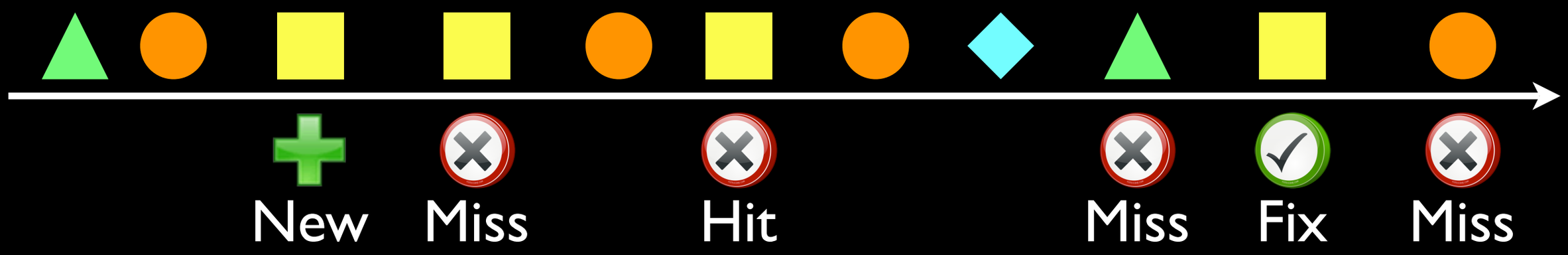
LRU CHANGE



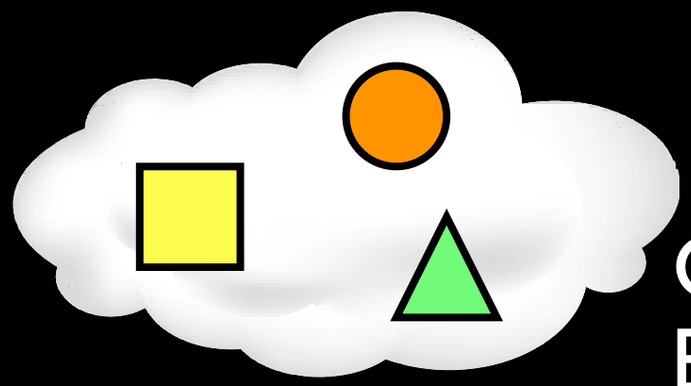
The BugCache Model

Entity	Last change	#Changes	#Bugs
	–1	4	2
	–2	2	1

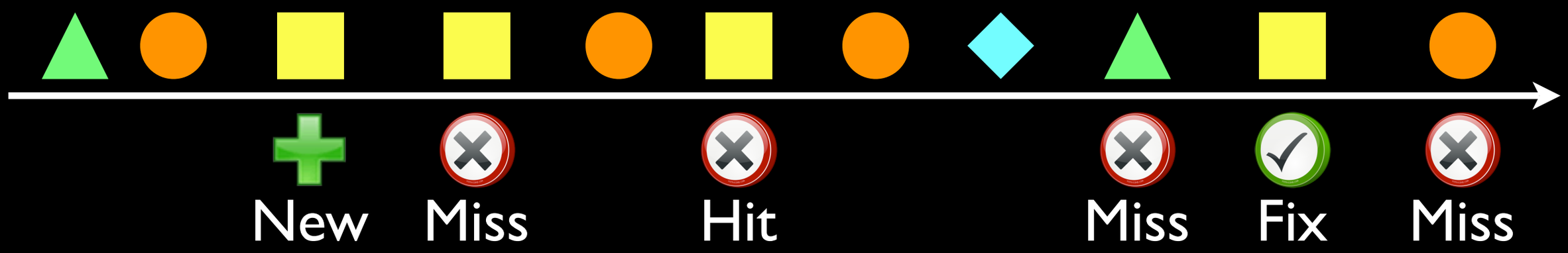
LRU CHANGE BUG



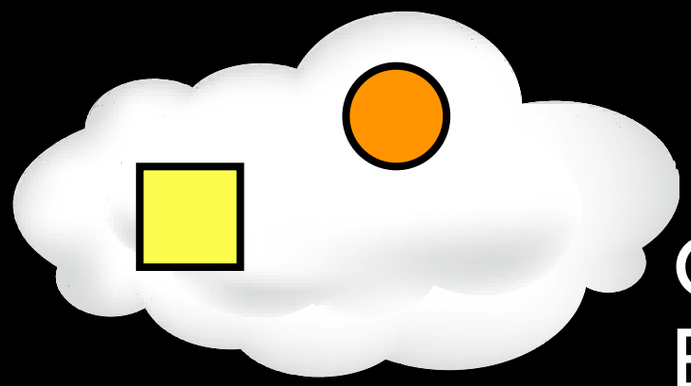
The BugCache Model



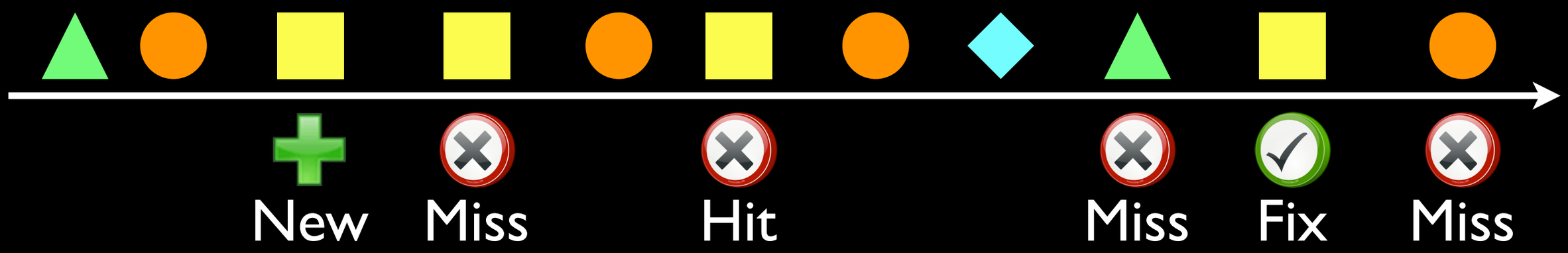
Cache size: 2
Block size: 1



The BugCache Model



Cache size: 2
Block size: 1



Loading Elements

Temporal locality – as shown before

Spatial locality – load "nearby" elements (i.e., co-changed before)

Changed-entity locality – load changed elements

New-entity locality – load new elements

Initial pre-fill – start with a loaded cache

Evaluation

Mozilla

jEdit

PostgreSQL

Columba

Exhaustive Evaluation

- Cache size: fixed
- Vary block size:
  0% to 100% of cache size
- Vary pre-fetch size:
  0% to 100% of cache size
- Vary replacement: LRU, CHANGE, BUG

several thousand experiments per project

Optimal Hit Rates

Methods

Project	BugCache	FixCache
Apache 1.3	59.6%	61.5%
Columba	58.9%	67.6%
Eclipse	64.5%	71.6%
JEdit	50.5%	48.9%
Mozilla	49.3%	55.0%
PostgreSQL	61.9%	59.2%
Subversion	68.3%	43.8%

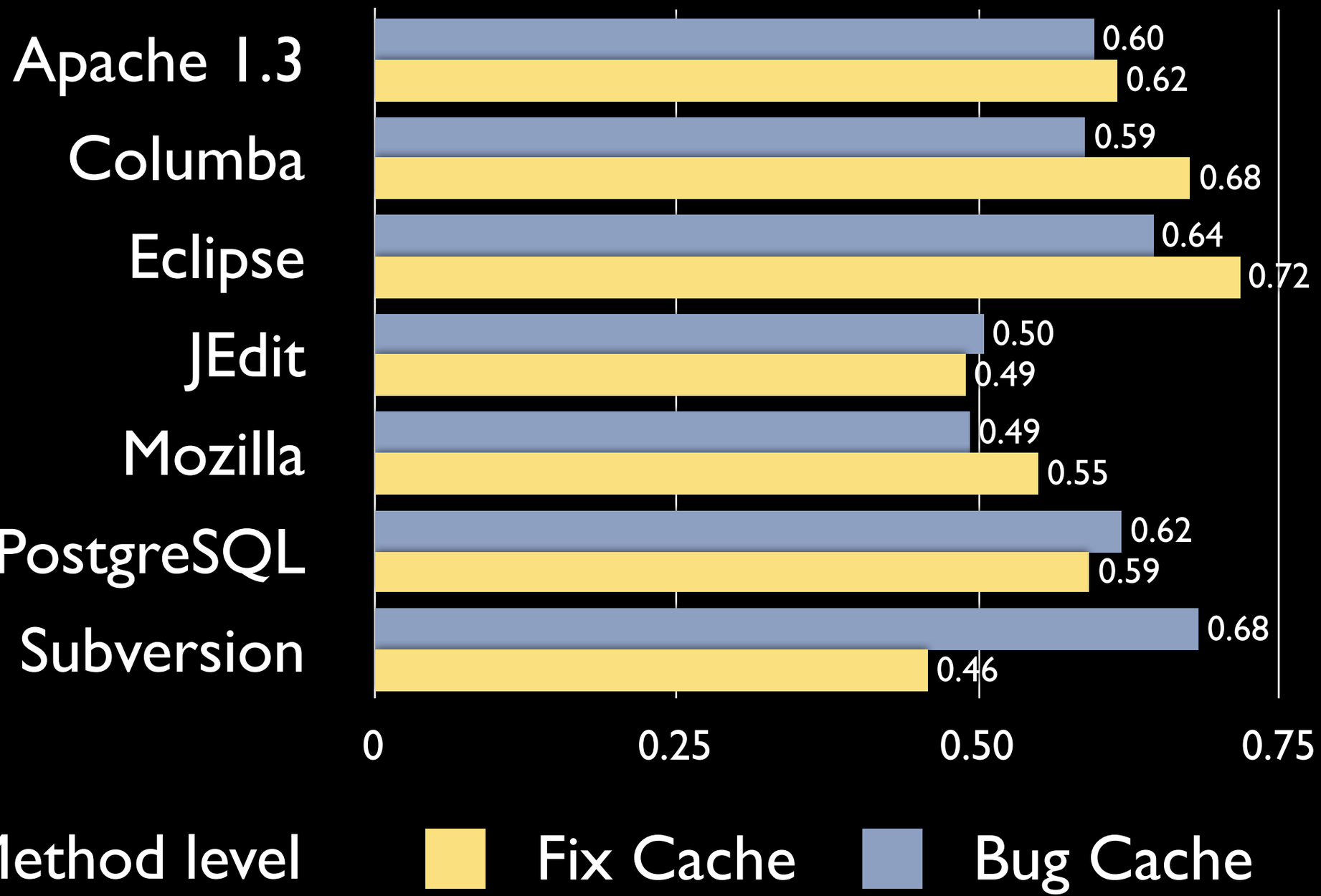
Cache size = 10%

Optimal Hit Rates

	Methods		Files	
Project	BugCache	FixCache	BugCache	FixCache
Apache 1.3	59.6%	61.5%	83.9%	81.5%
Columba	58.9%	67.6%	83.5%	83.0%
Eclipse	64.5%	71.6%	95.1%	95.0%
JEdit	50.5%	48.9%	85.7%	85.4%
Mozilla	49.3%	55.0%	93.3%	88.0%
PostgreSQL	61.9%	59.2%	73.9%	71.0%
Subversion	68.3%	43.8%	82.0%	81.3%

Cache size = 10%

BugCache vs FixCache



Optimal Hit Rates

Project	Hit rate	Block	Pre-fetch	Policy
Apache 1.3	83.9%	46.7%	0%	LRU
Columba	83.5%	32.2%	0%	BUG
Eclipse	95.1%	99.4%	0%	BUG
JEdit	85.7%	40.5%	0%	LRU
Mozilla	93.3%	42.5%	0%	LRU
PostgreSQL	73.9%	1.7%	0%	LRU
Subversion	82.0%	23.1%	0%	LRU

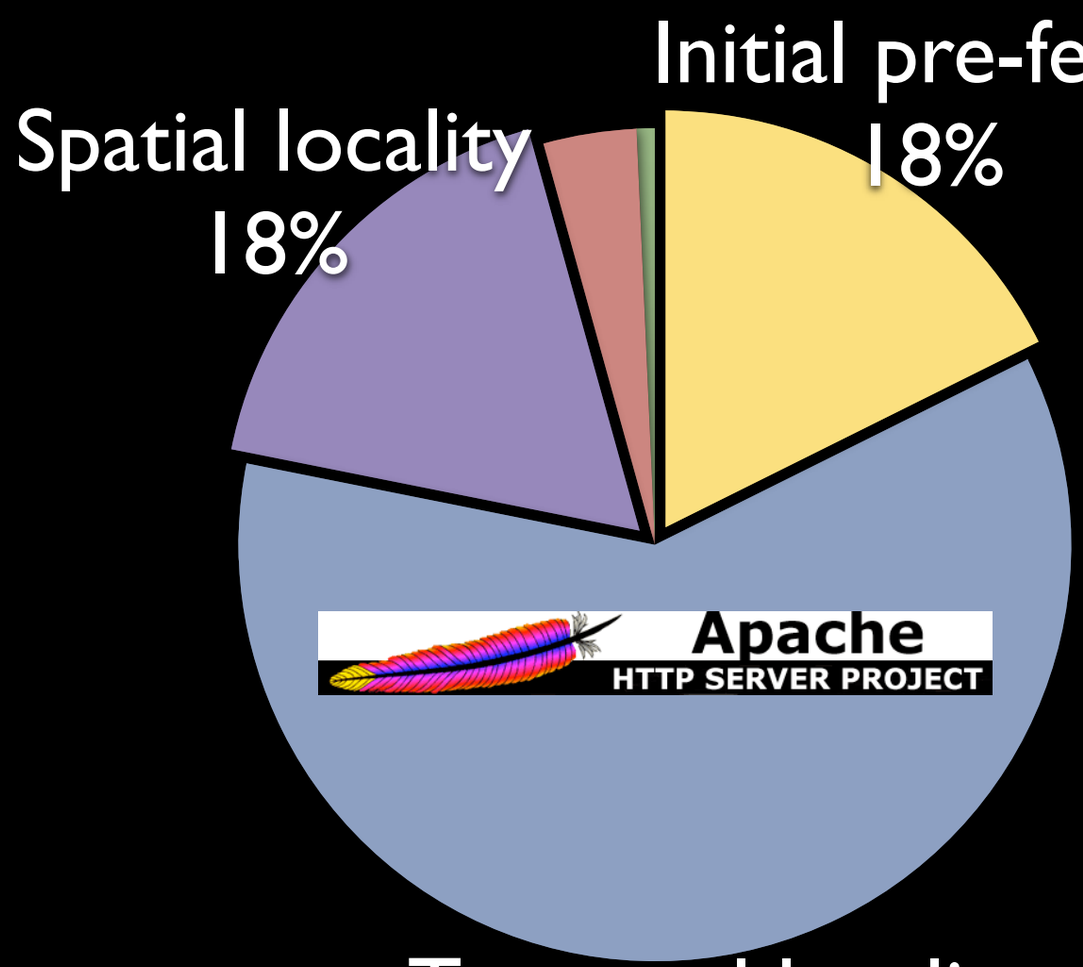
Cache size = 10% of all files

Optimal Hit Rates

Project	Hit rate	Block	Pre-fetch	Policy
Apache 1.3	59.6%	60.2%	11.4%	BUG
Columba	58.9%	99.9%	15.7%	BUG
Eclipse	64.5%	20.0%	0.0%	BUG
JEdit	50.5%	0.2%	7.7%	BUG
Mozilla	49.3%	79.9%	12.0%	LRU
PostgreSQL	61.9%	40.1%	11.8%	BUG
Subversion	68.3%	99.2%	14.6%	BUG

Cache size = 10% of all functions/methods

Reasons for Hits

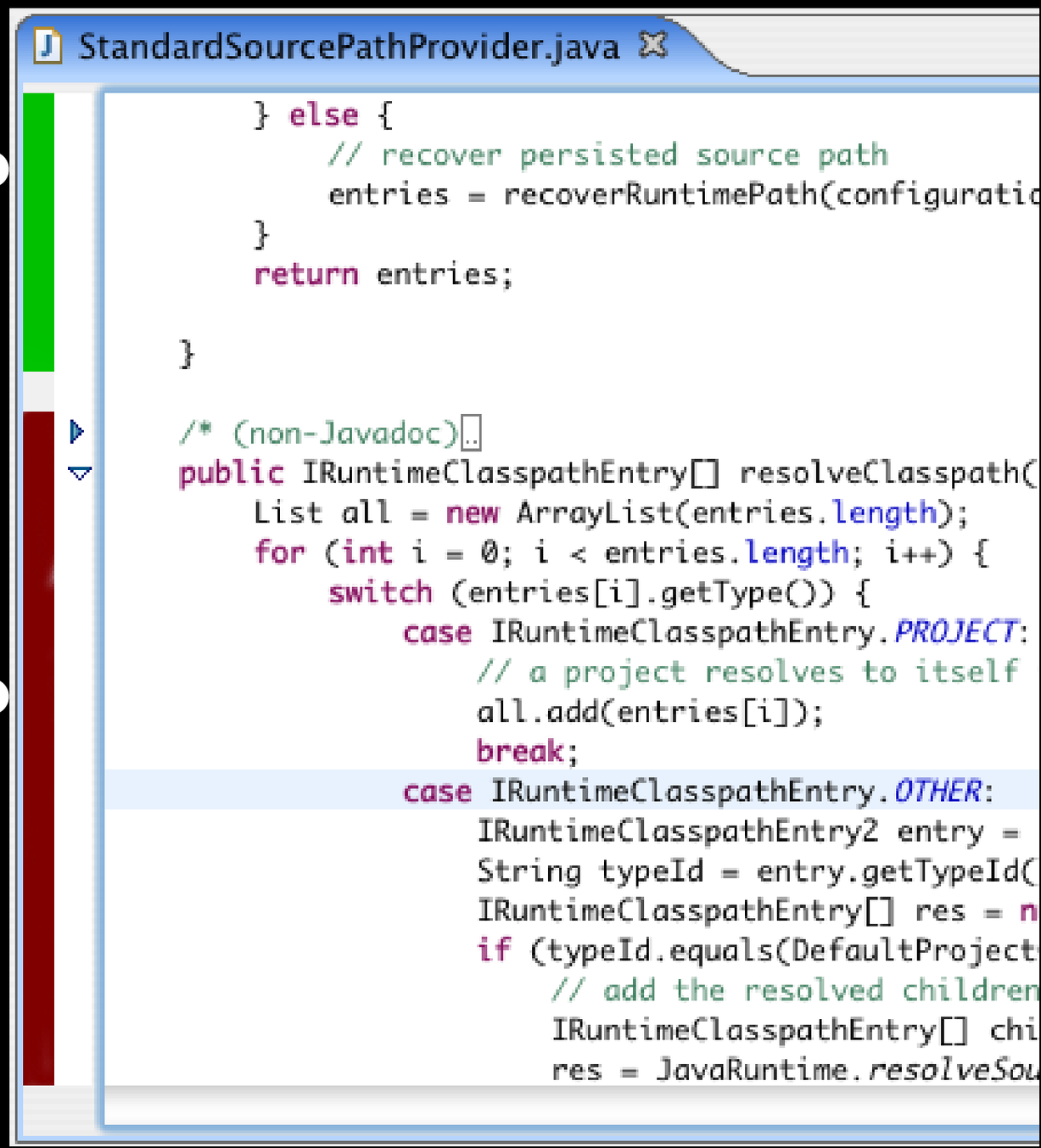


- Initial pre-fetch
- Temporal locality
- Spatial locality
- Changed-entity locality
- New-entity locality

Warning Developers

"Safe" Location
(not in FixCache)

Risky Location
(red, in FixCache)



Conclusion

- Imports correlate with defects/vulnerabilities
- Metrics also correlate with defects
- But no universal predictor for defects, use historic data to train predictors.
- Localities between defects

Register for exam!

- Deadline for registration is TOMORROW! Tomorrow is June 22nd.
- You will need an RZ account and a TAN list. Get them in the Mensa building.
- To register log on to the following website: https://www.lsf.uni-saarland.de/