Automatic Identification of Bug-Introducing Changes

Sunghun Kim
Kai Pan
E. James Whitehead, Jr.

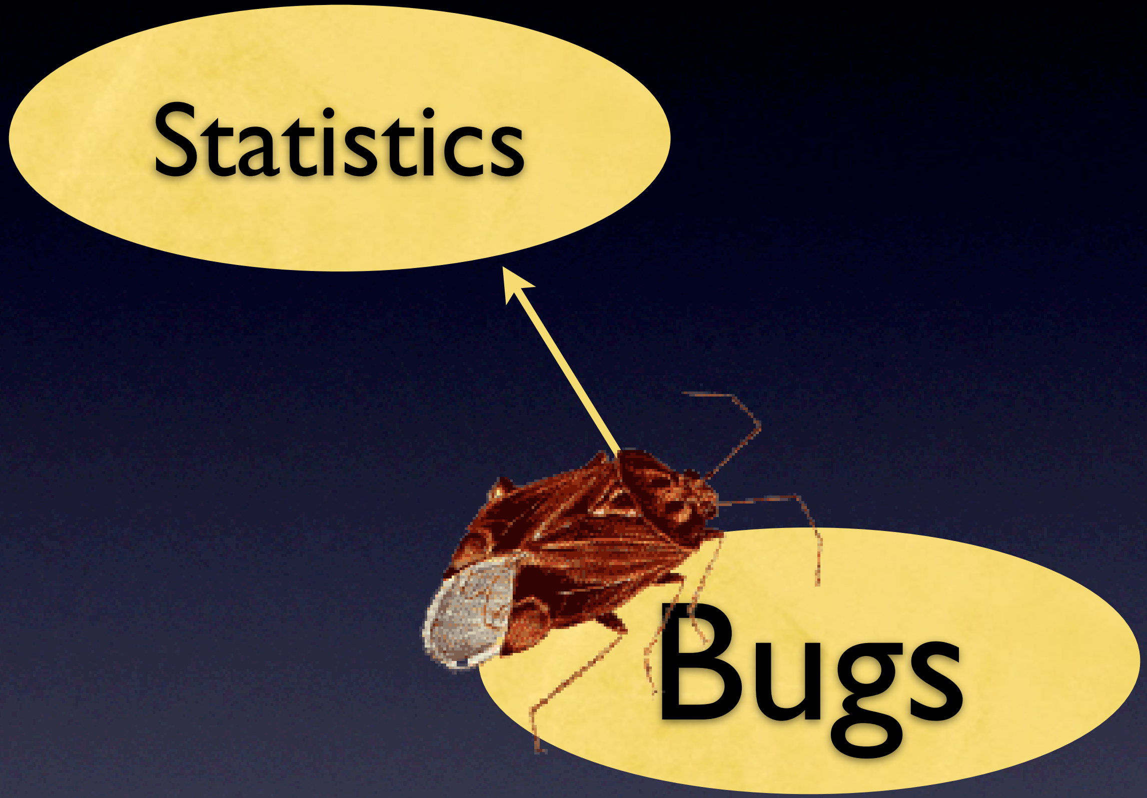
University of California, Santa Cruz, USA

Tom Zimmermann

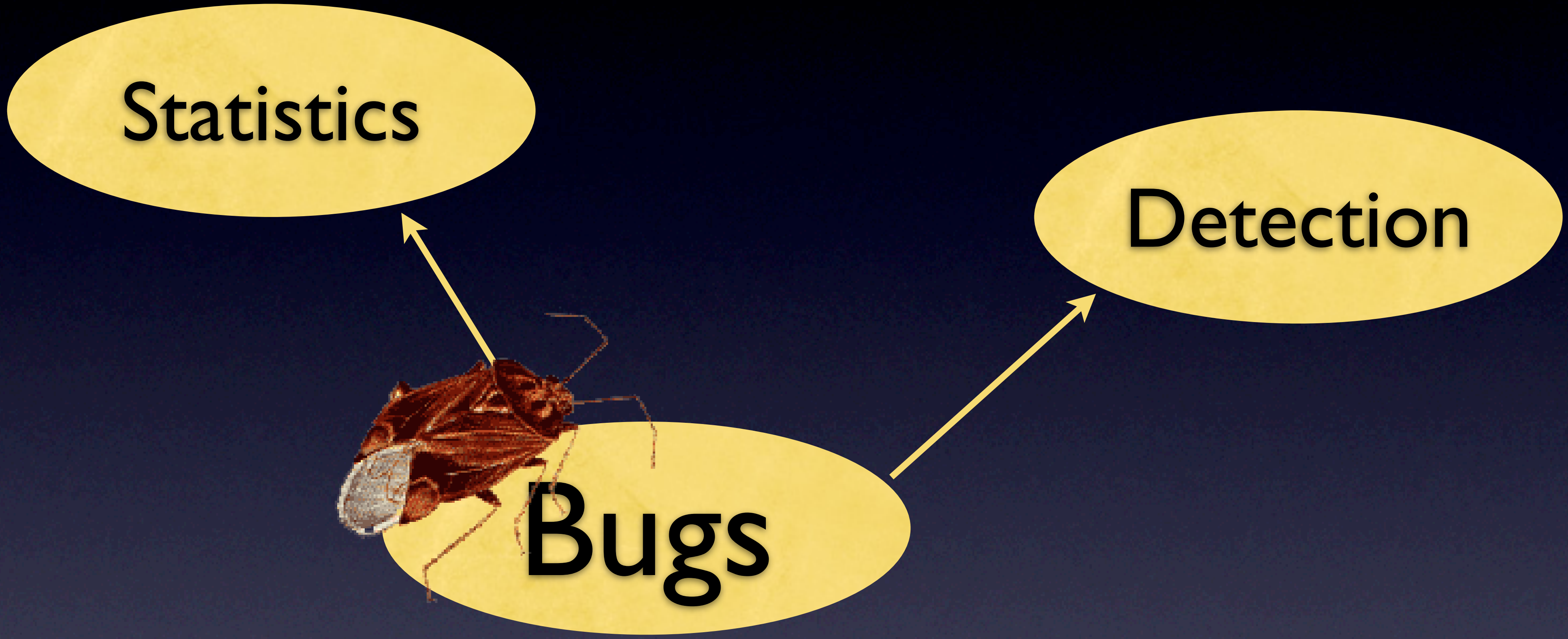
Saarland University, Saarbrücken, Germany

Motivation

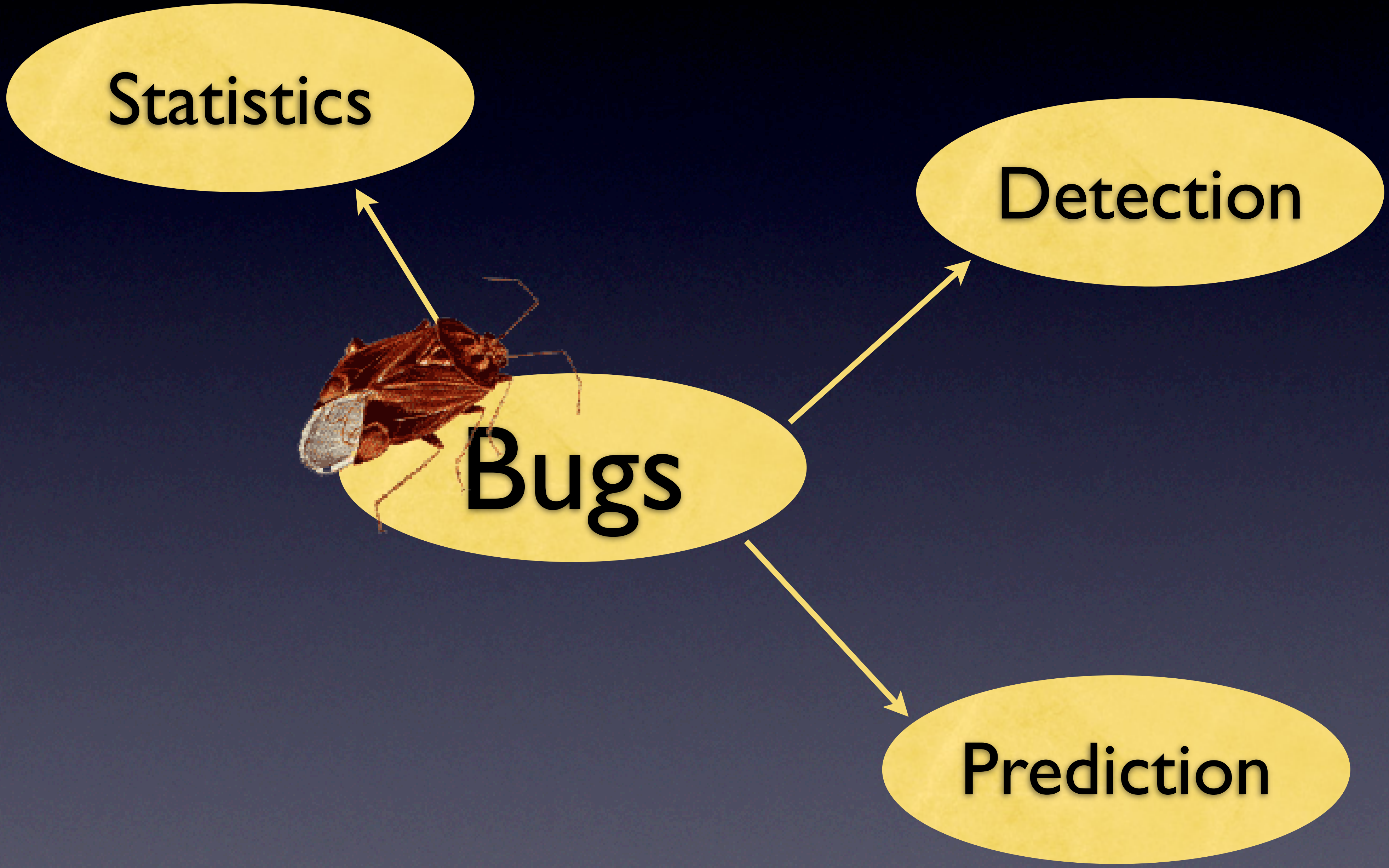
Motivation



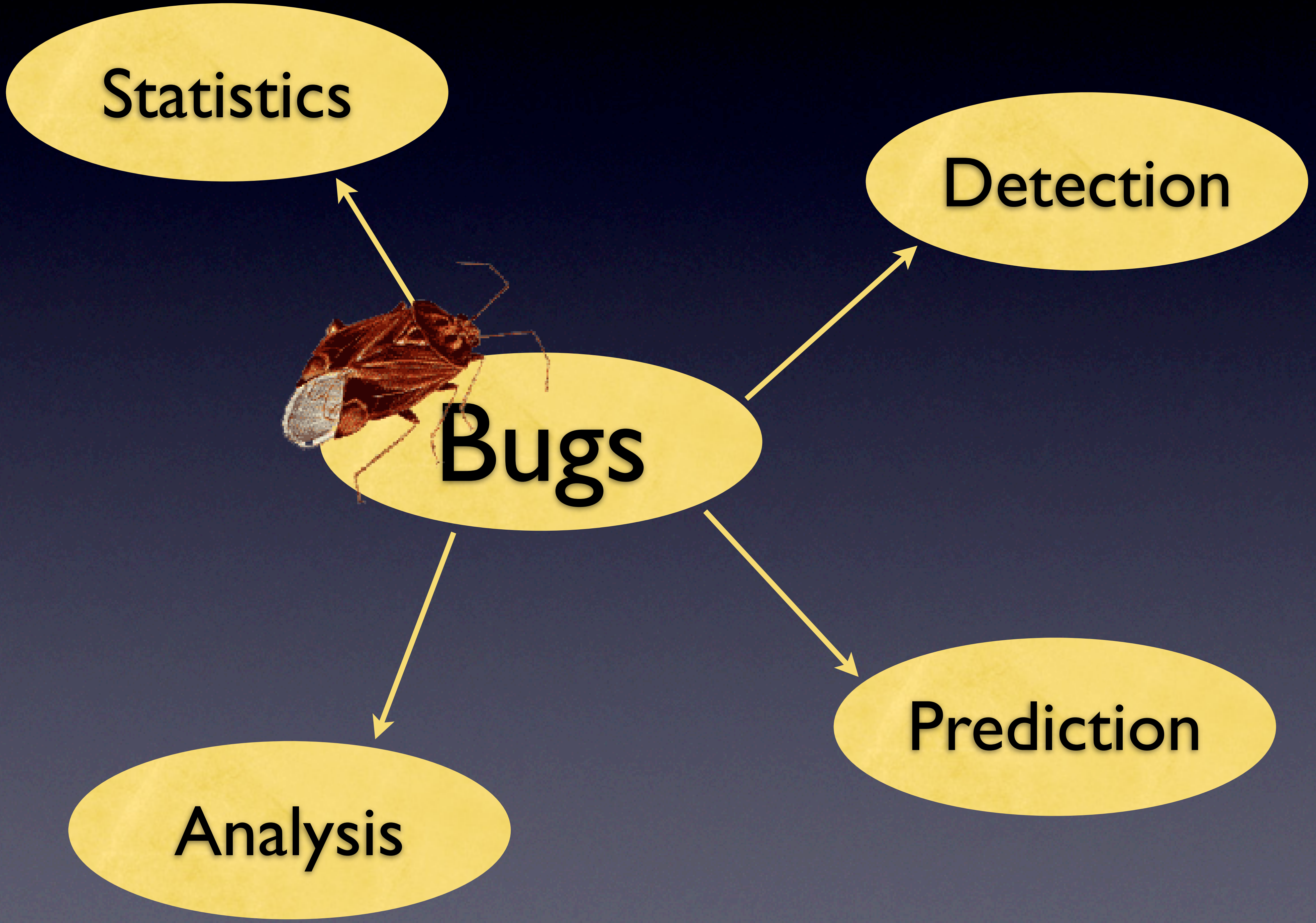
Motivation



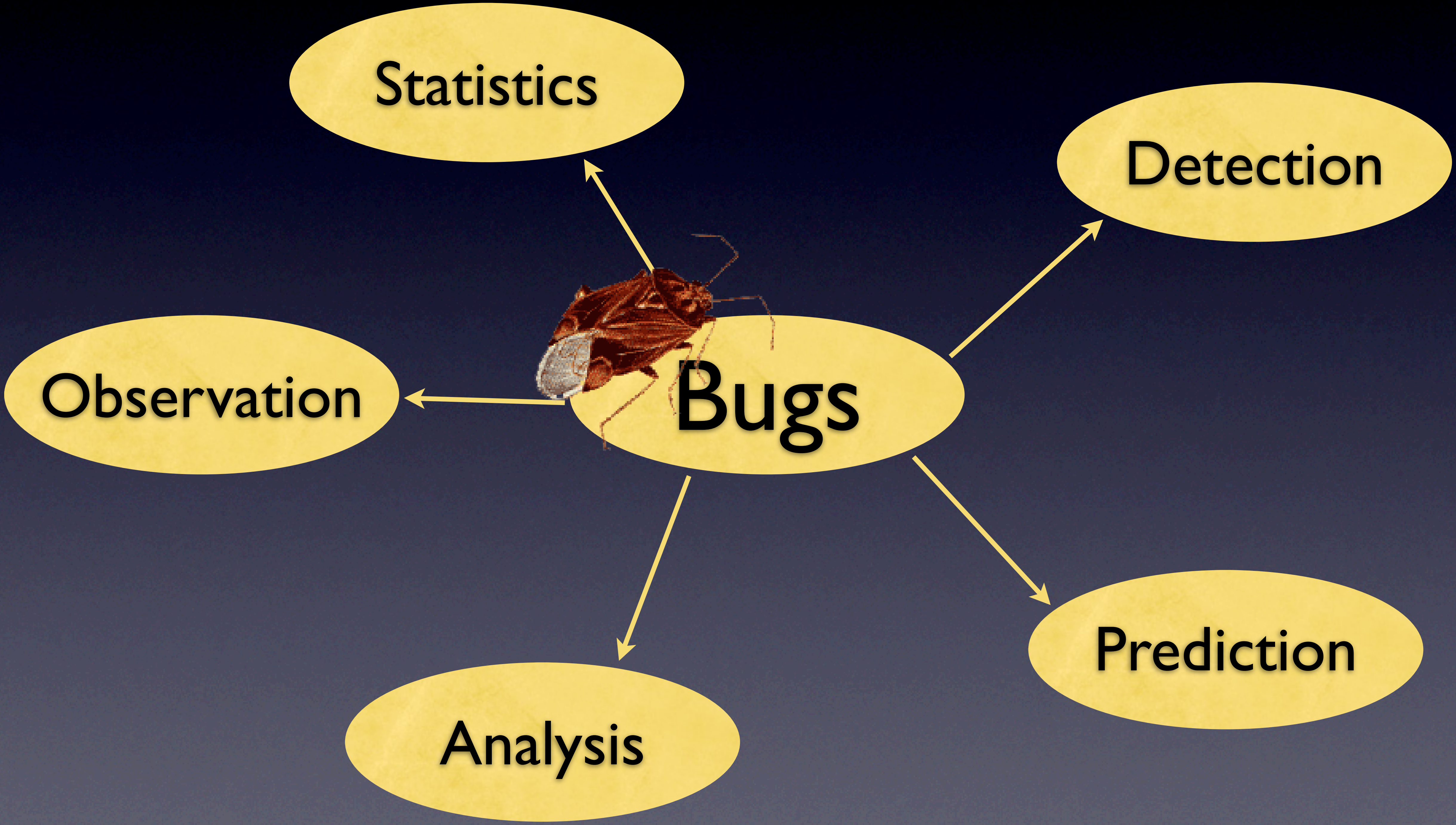
Motivation



Motivation



Motivation


Statistics
Detection
Observation
Bugs
Analysis
Prediction

So far: Focus on fixes

```
teicher     2003-10-29 16:11:01
fixes issues mentioned in bug 45635: [hovering] rollover hovers
- mouse exit detection is safer and should not allow for
  loopholes any more, except for shell deactiviation
- hovers behave like normal ones:
        - tooltips pop up below the control
        - they move with subjectArea
        - once a popup is showing, they will show up instantly
```

So far: Focus on fixes

```
teicher      2003-10-29 16:11:01
fixes issues mentioned in bug 45635: [hovering] rollover hovers
- mouse exit detection is safer and should not allow for
  loopholes any more, except for shell deactiviation
- hovers behave like normal ones:
        - tooltips pop up below the control
        - they move with subjectArea
        - once a popup is showing, they will show up instantly
```

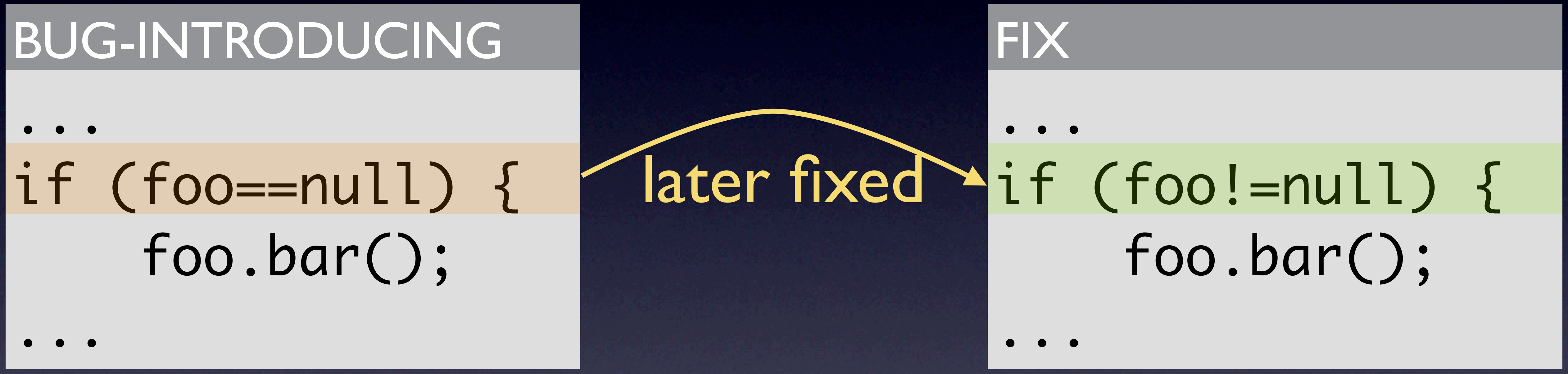
Fixes give only the location of a defect,
not when it was introduced.

Bug-introducing changes

BUG-INTRODUCING

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

Bug-introducing changes



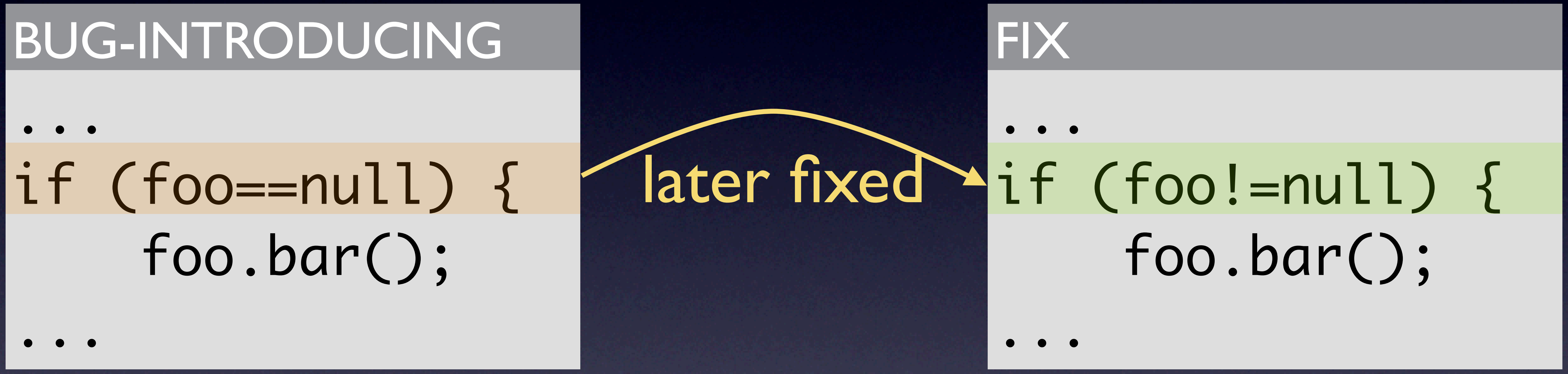
BUG-INTRODUCING

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

FIX

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

Bug-introducing changes

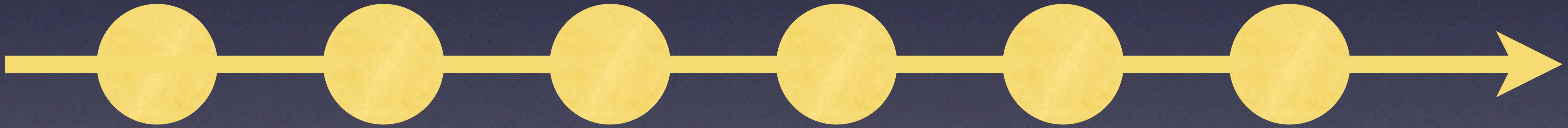


```
...
if (foo==null) {
    foo.bar();
...
```

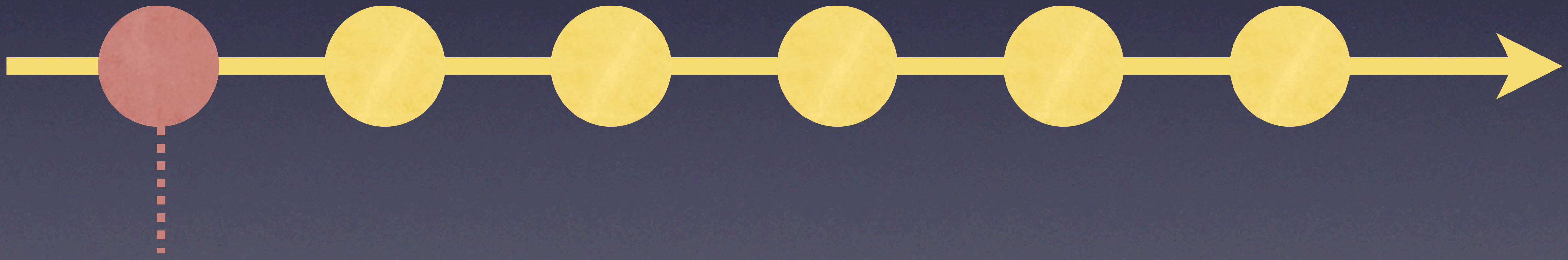
```
...
if (foo!=null) {
    foo.bar();
...
```

Bug-introducing changes are changes that lead to problems as indicated by later fixes.

Life-cycle of a “bug”

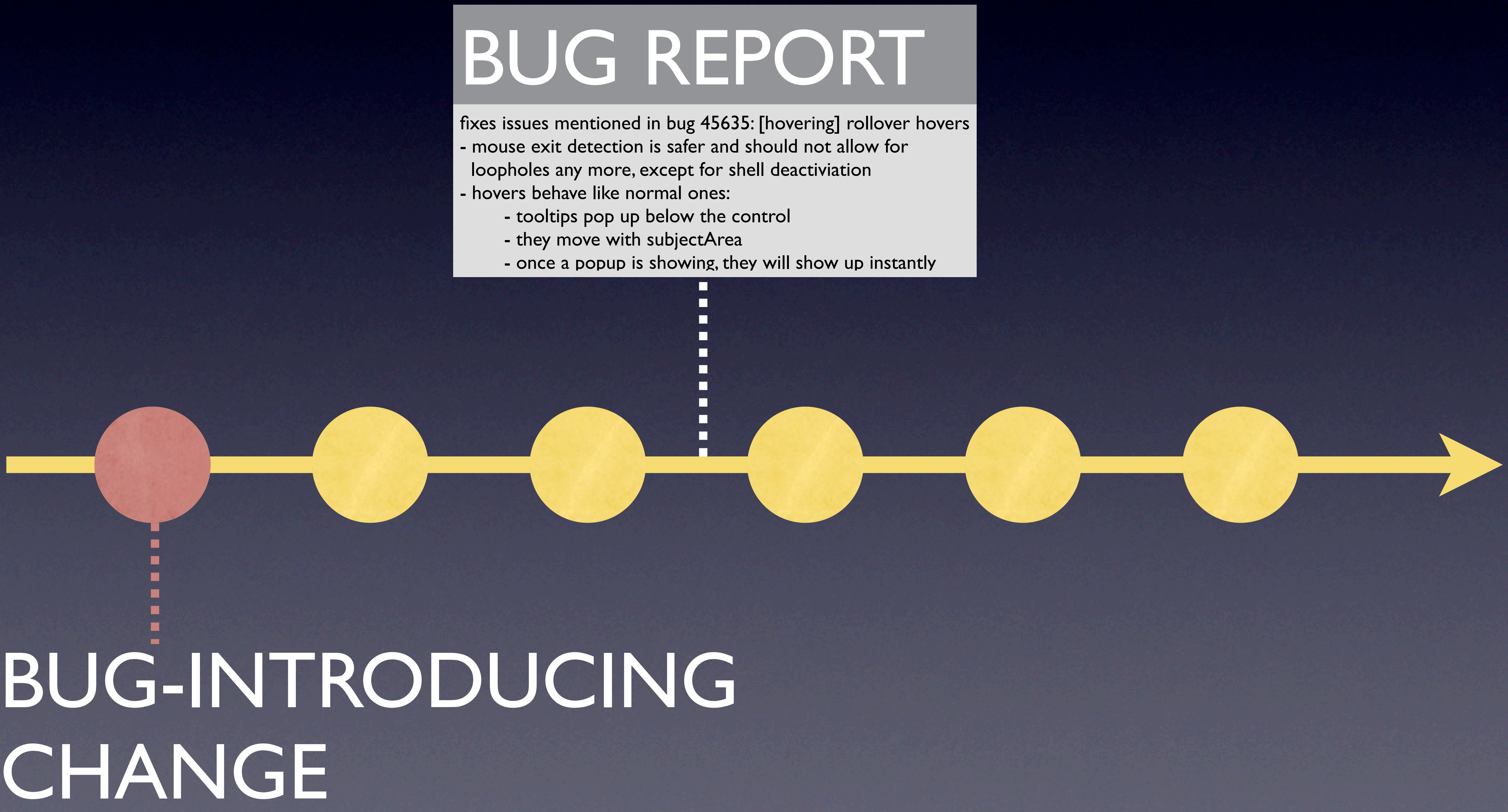


Life-cycle of a "bug"



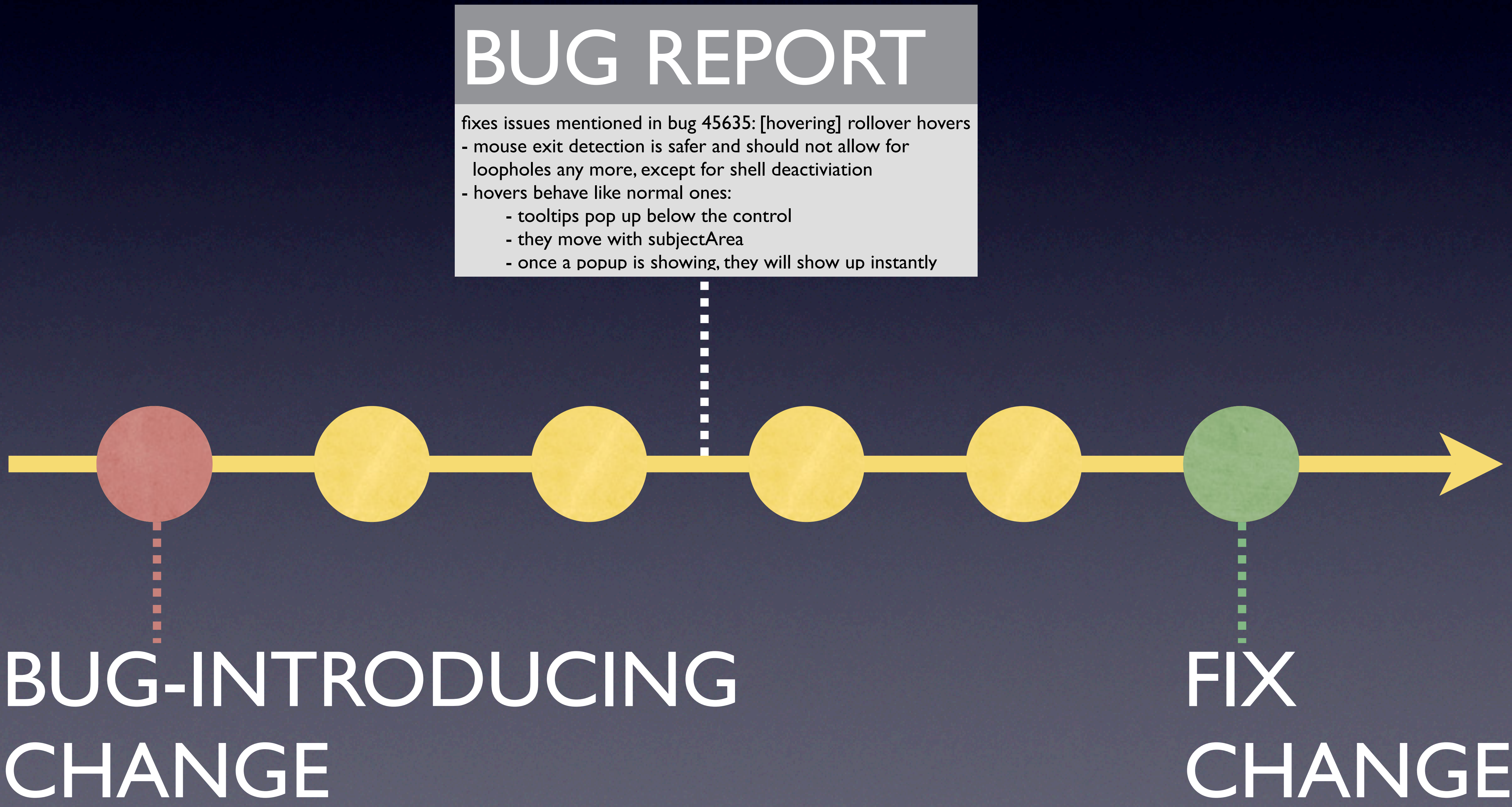
BUG-INTRODUCING CHANGE

Life-cycle of a "bug"



fixes issues mentioned in bug 45635: [hovering] rollover hovers
- mouse exit detection is safer and should not allow for
  loopholes any more, except for shell deactiviation
- hovers behave like normal ones:
    - tooltips pop up below the control
    - they move with subjectArea
    - once a popup is showing, they will show up instantly

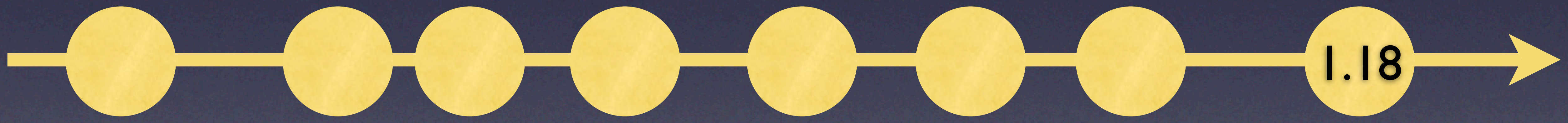
Life-cycle of a “bug”



BUG REPORT

fixes issues mentioned in bug 45635: [hovering] rollover hovers
- mouse exit detection is safer and should not allow for
  loopholes any more, except for shell deactiviation
- hovers behave like normal ones:
  - tooltips pop up below the control
  - they move with subjectArea
  - once a popup is showing, they will show up instantly

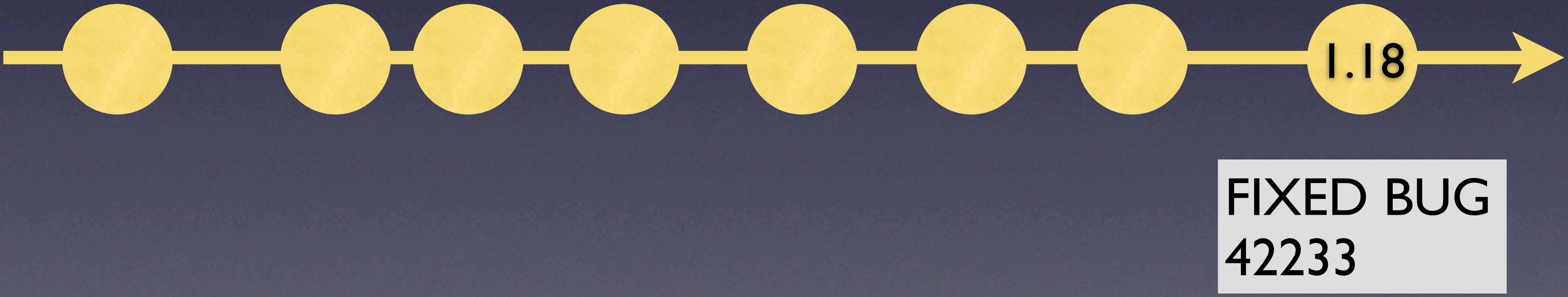
The SZZ algorithm



FIXED BUG
42233

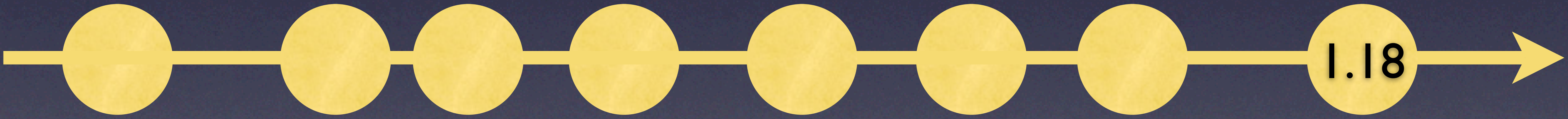
The SZZ algorithm

```
    $ cvs annotate -r 1.17 Foo.java
    ...
20: 1.11 (john 12-Feb-03):     return i/0;
    ...
40: 1.14 (kate 23-May-03):     return 42;
    ...
60: 1.16 (mary 10-Jun-03):     int i=0;
```



The SZZ algorithm

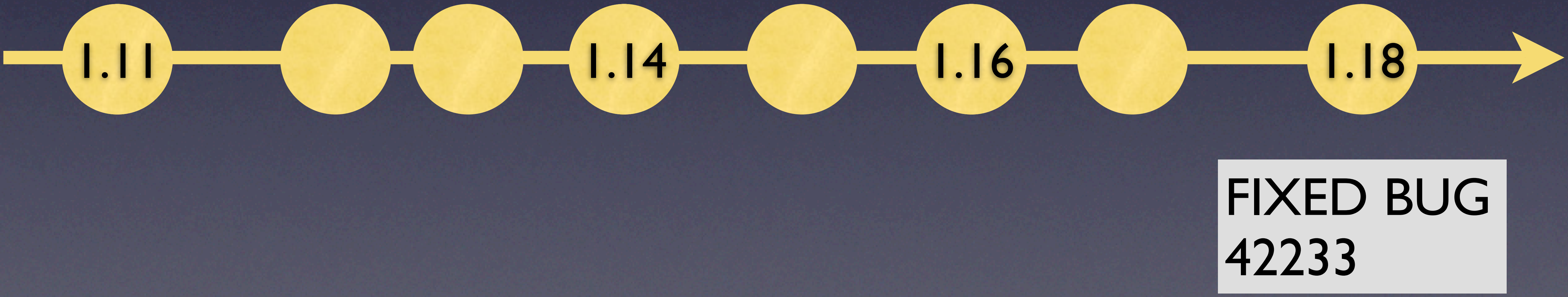
```
    $ cvs annotate -r 1.17 Foo.java
    ...
20: 1.11 (john 12-Feb-03):     return i/0;
    ...
40: 1.14 (kate 23-May-03):     return 42;
    ...
60: 1.16 (mary 10-Jun-03):     int i=0;
```



FIXED BUG
42233

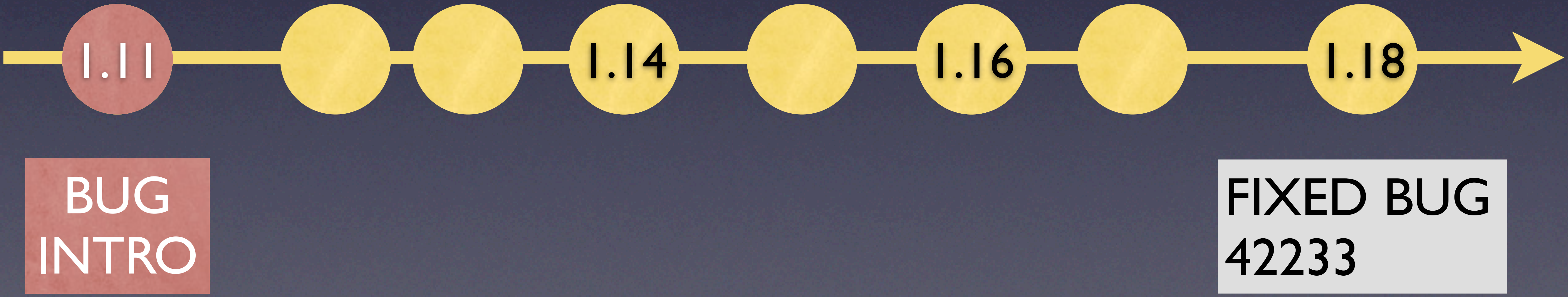
The SZZ algorithm

```
    $ cvs annotate -r 1.17 Foo.java
    ...
20: 1.11 (john 12-Feb-03):     return i/0;
    ...
40: 1.14 (kate 23-May-03):     return 42;
    ...
60: 1.16 (mary 10-Jun-03):     int i=0;
```



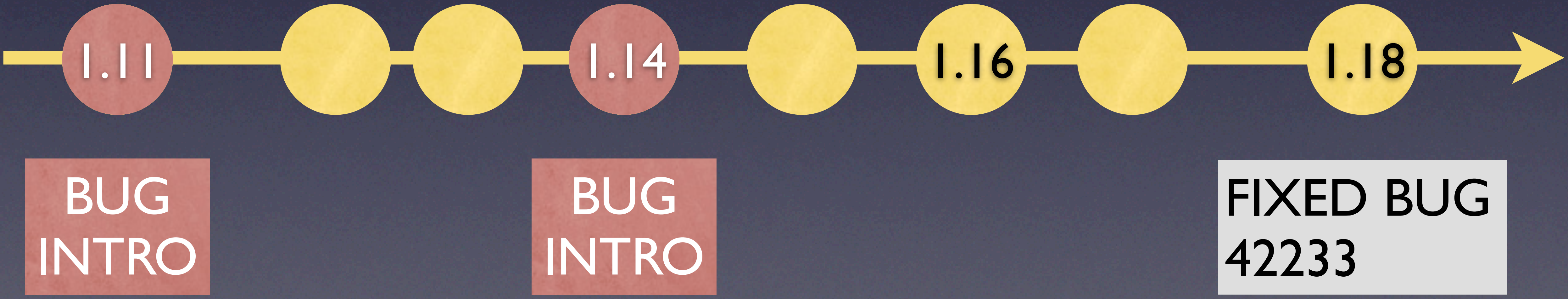
The SZZ algorithm

```
     $ cvs annotate -r 1.17 Foo.java
     ...
20: 1.11 (john 12-Feb-03):     return i/0;
     ...
40: 1.14 (kate 23-May-03):     return 42;
     ...
60: 1.16 (mary 10-Jun-03):     int i=0;
```



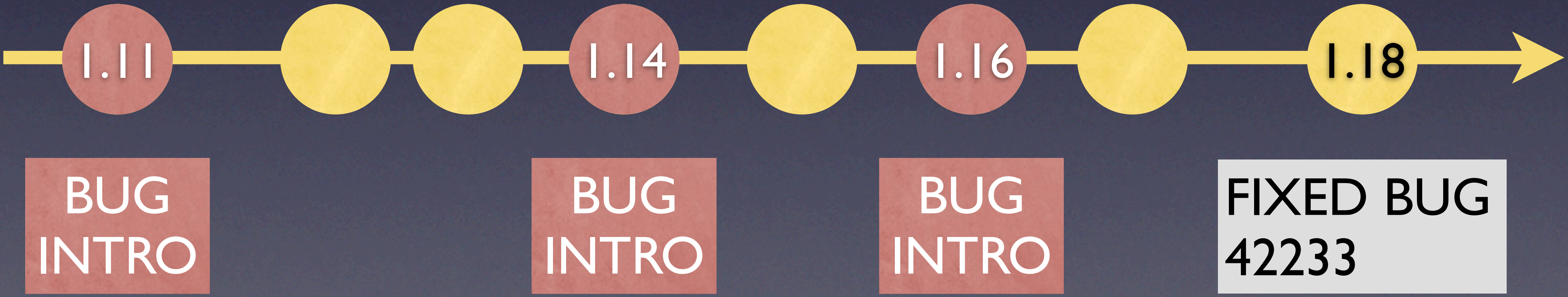
The SZZ algorithm

```
    $ cvs annotate -r 1.17 Foo.java
    ...
20: 1.11 (john 12-Feb-03):     return i/0;
    ...
40: 1.14 (kate 23-May-03):     return 42;
    ...
60: 1.16 (mary 10-Jun-03):     int i=0;
```

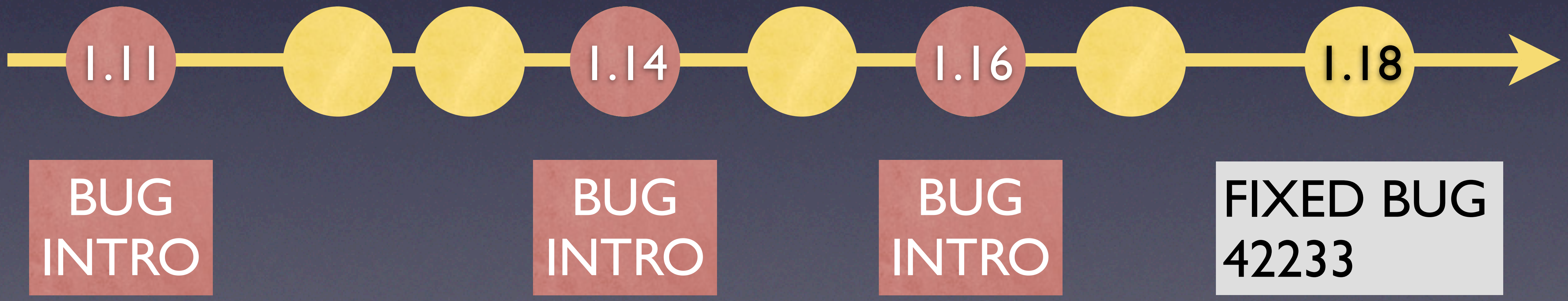


The SZZ algorithm

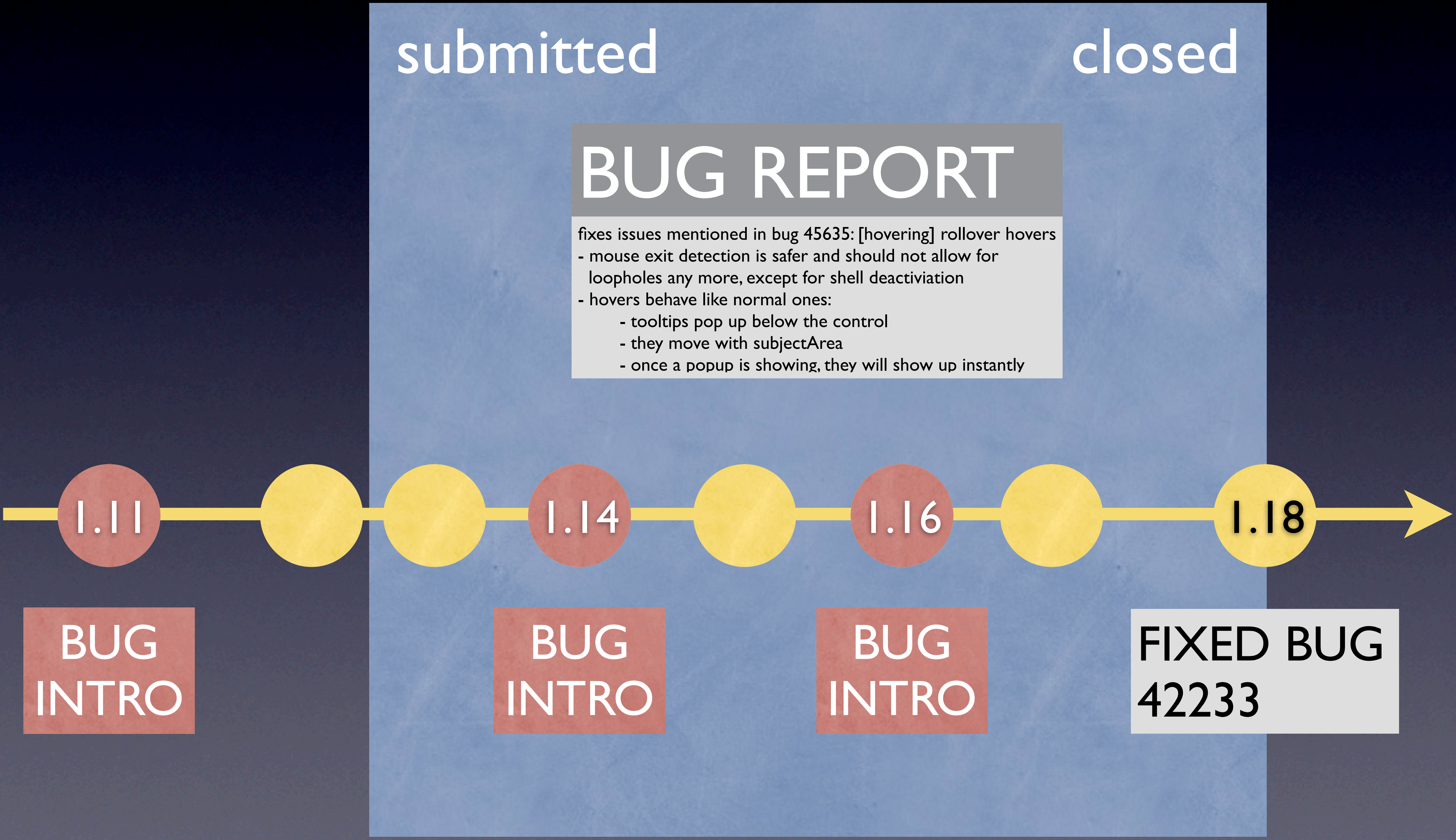
```
    $ cvs annotate -r 1.17 Foo.java
    ...
20: 1.11 (john 12-Feb-03):     return i/0;
    ...
40: 1.14 (kate 23-May-03):     return 42;
    ...
60: 1.16 (mary 10-Jun-03):     int i=0;
```



The SZZ algorithm



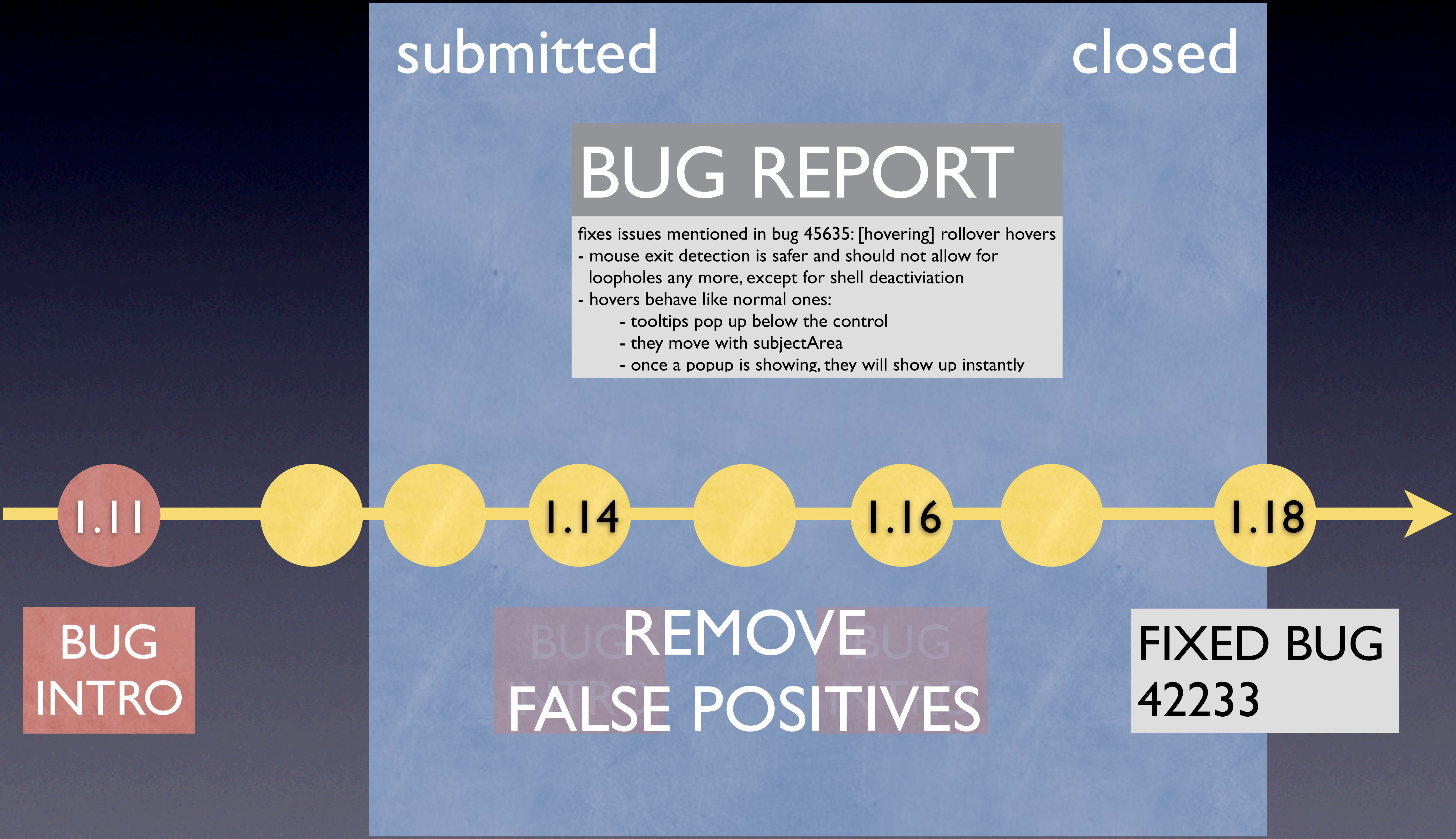
The SZZ algorithm



BUG REPORT

fixes issues mentioned in bug 45635: [hovering] rollover hovers
- mouse exit detection is safer and should not allow for loopholes any more, except for shell deactiviation
- hovers behave like normal ones:
    - tooltips pop up below the control
    - they move with subjectArea
    - once a popup is showing, they will show up instantly

The SZZ algorithm



fixes issues mentioned in bug 45635: [hovering] rollover hovers
- mouse exit detection is safer and should not allow for loopholes any more, except for shell deactiviation
- hovers behave like normal ones:
    - tooltips pop up below the control
    - they move with subjectArea
    - once a popup is showing, they will show up instantly

Drawbacks of SZZ

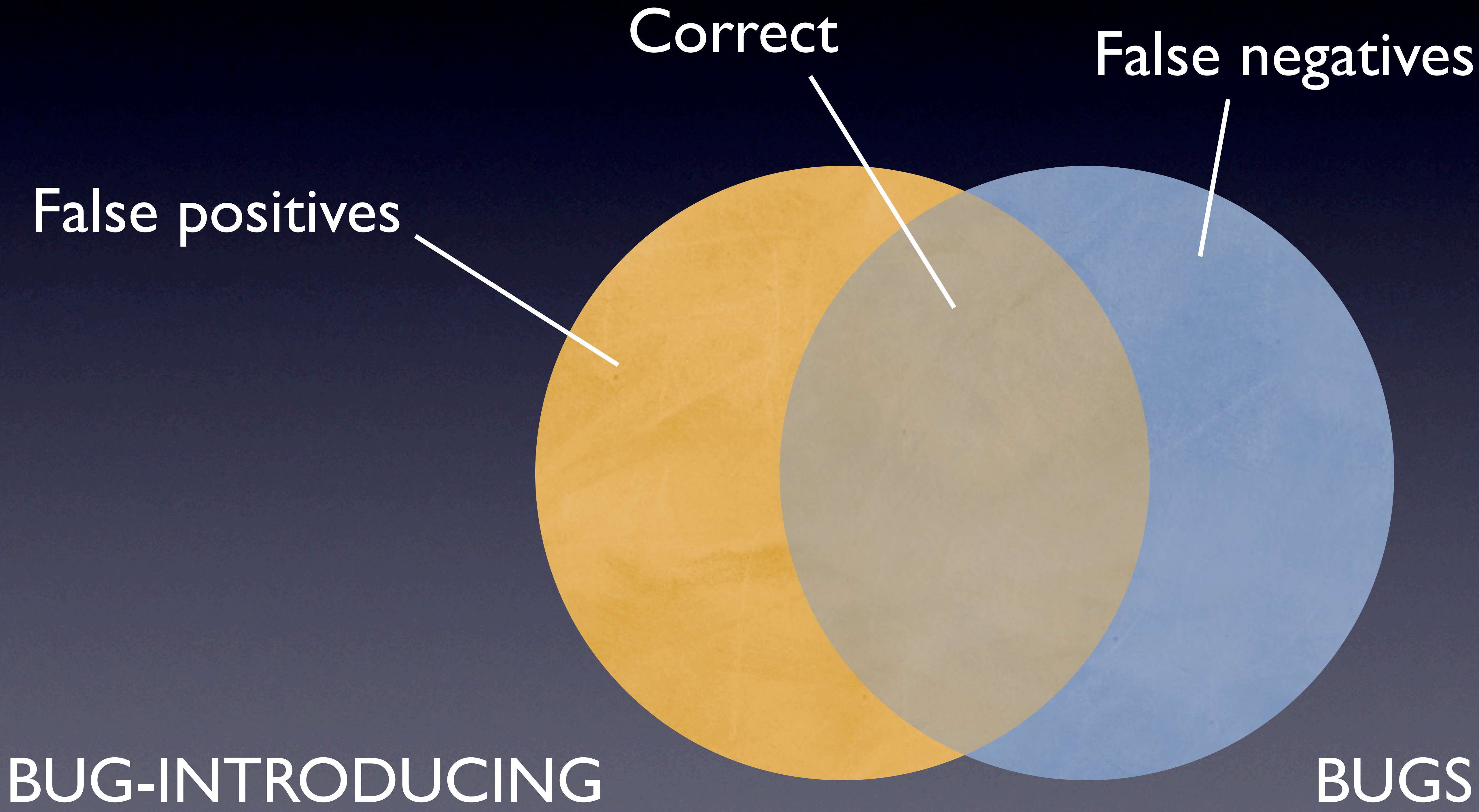
Annotation by SCMs is insufficient.
(line number in bug-introducing revision is missing)

Drawbacks of SZZ

Annotation by SCMs is insufficient.
(line number in bug-introducing revision is missing)

Not all modifications are fixes.
(blank lines, comments, etc.)

False negatives and positives



An example

BUG-INTRODUCING

```
void bar() {
  if (val==null) {
    println(val);
  }
...
```

Revision 7: tom
- introduces the defects

An example

BUG-INTRODUCING

```
void bar() {
  if (val==null) {
    println(val);
  }
...
```

Revision 7: tom
- introduces the defects

```
void foo() {
  // print val
  if (val==null)
  {
    println(val);
  }
...
```

Revision 23: jim
- inserts a comment
- reformats if statement

BUG FIX

```
void foo() {
  // print value
  if (val!=null)
  {
    println(val);
  }
...
```

Revision 42: kim
- changes comment
- corrects defect

An example

BUG-INTRODUCING

```
void bar() {
  if (val==null) {
    println(val);
  }
...
```

Revision 7: tom
- introduces the defects

```
void foo() {
  // print val
  if (val==null)
  {
    println(val);
  }
...
```

Revision 23: jim
- inserts a comment
- reformats if statement

BUG FIX

```
void foo() {
  // print value
  if (val!=null)
  {
    println(val);
  }
...
```

Revision 42: kim
- changes comment
- corrects defect

An example

BUG-INTRODUCING

```
void bar() {
  if (val==null) {
    println(val);
  }
...
```

Revision 7: tom
- introduces the defects

```
void foo() {
  // print val
  if (val==null)
  {
    println(val);
  }
...
```

Revision 23: jim
- inserts a comment
- reformats if statement

BUG FIX

```
void foo() {
  // print value
  if (val!=null)
  {
    println(val);
  }
...
```

Revision 42: kim
- changes comment
- corrects defect

An example

BUG-INTRODUCING

```
void bar() {
  if (val==null) {
    println(val);
  }
...
```

Revision 7: tom
- introduces the defects

```
void foo() {
  // print val
  if (val==null)
  {
    println(val);
  }
...
```

Revision 23: jim
- inserts a comment
- reformats if statement

BUG FIX

```
void foo() {
  // print value
  if (val!=null)
  {
    println(val);
  }
...
```

Revision 42: kim
- changes comment
- corrects defect

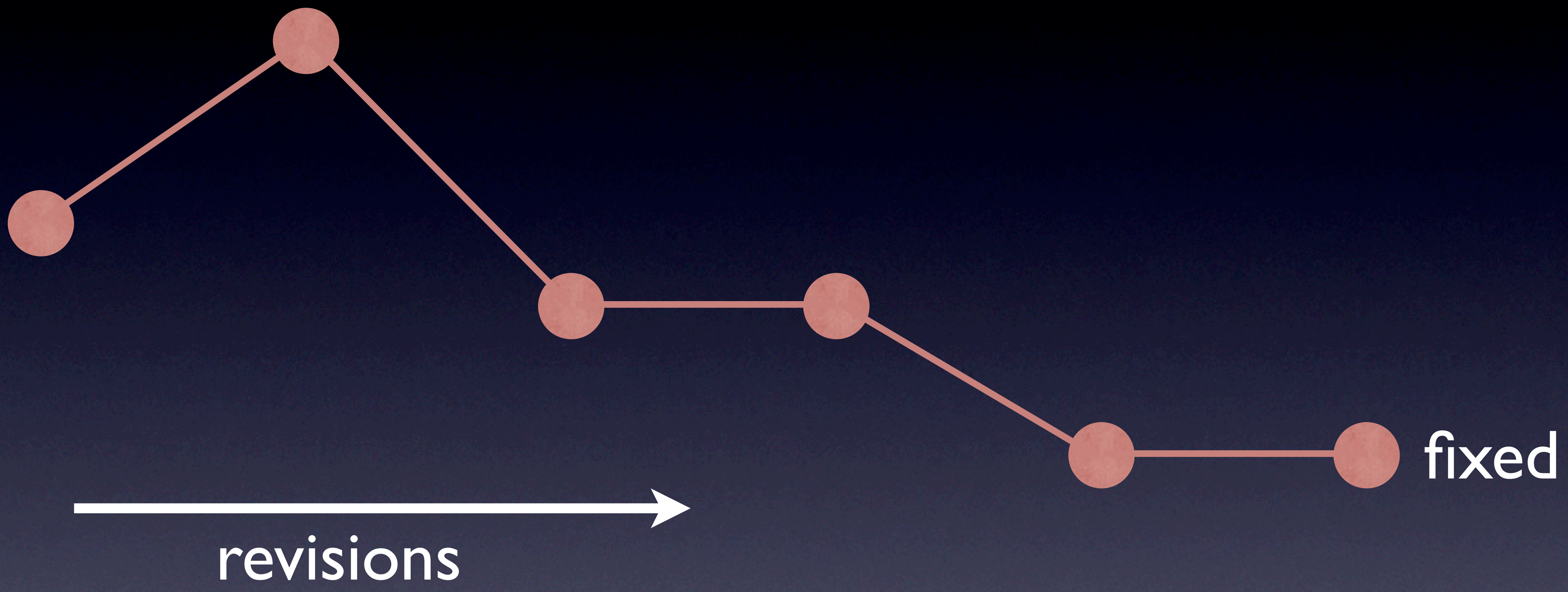
The originial SZZ algorithm has too many false positives (rev 23) and false negatives (rev 7).

Our study

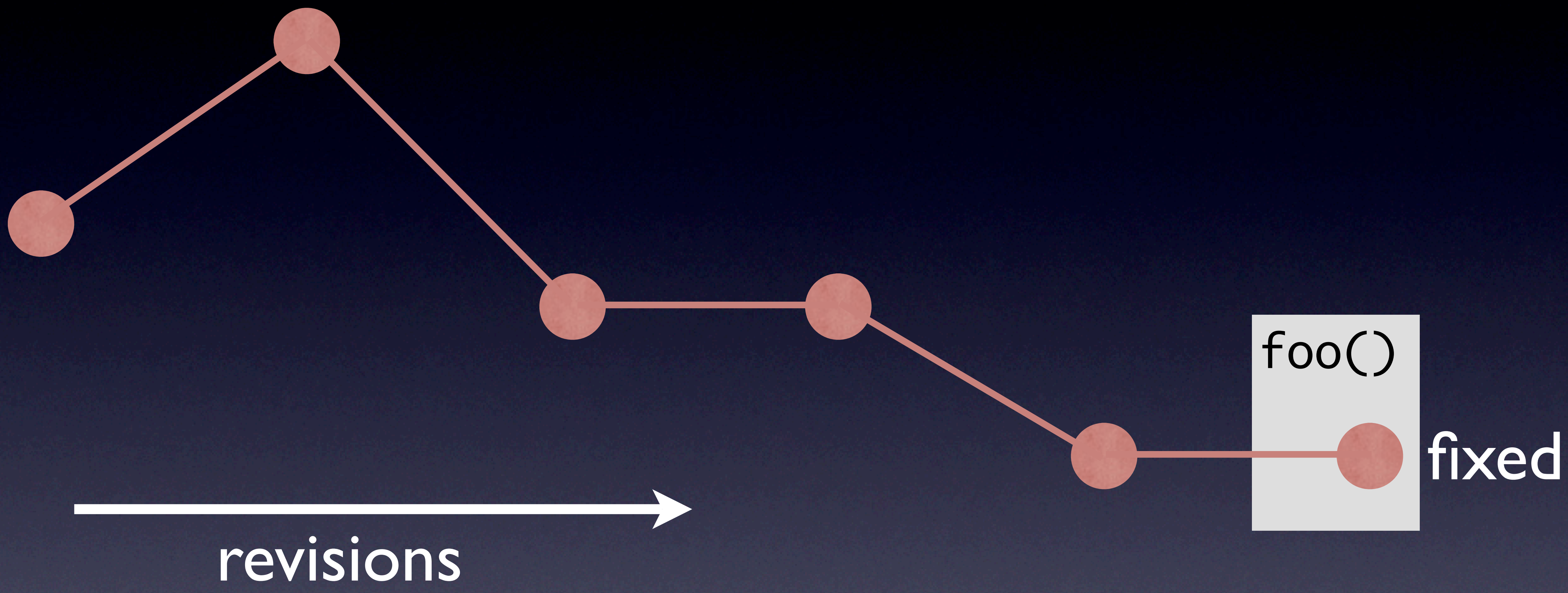
Project	COLUMBA	ECLIPSE (jdt.core)
Software type	Email client	IDE
Investigated period	11/2002-06/2003	06/2001-03/2002
Number of revisions	500	1000
Number of fixes	143 (29%)	158 (16%)
Average LOC	48,135	111,059

Annotation graphs

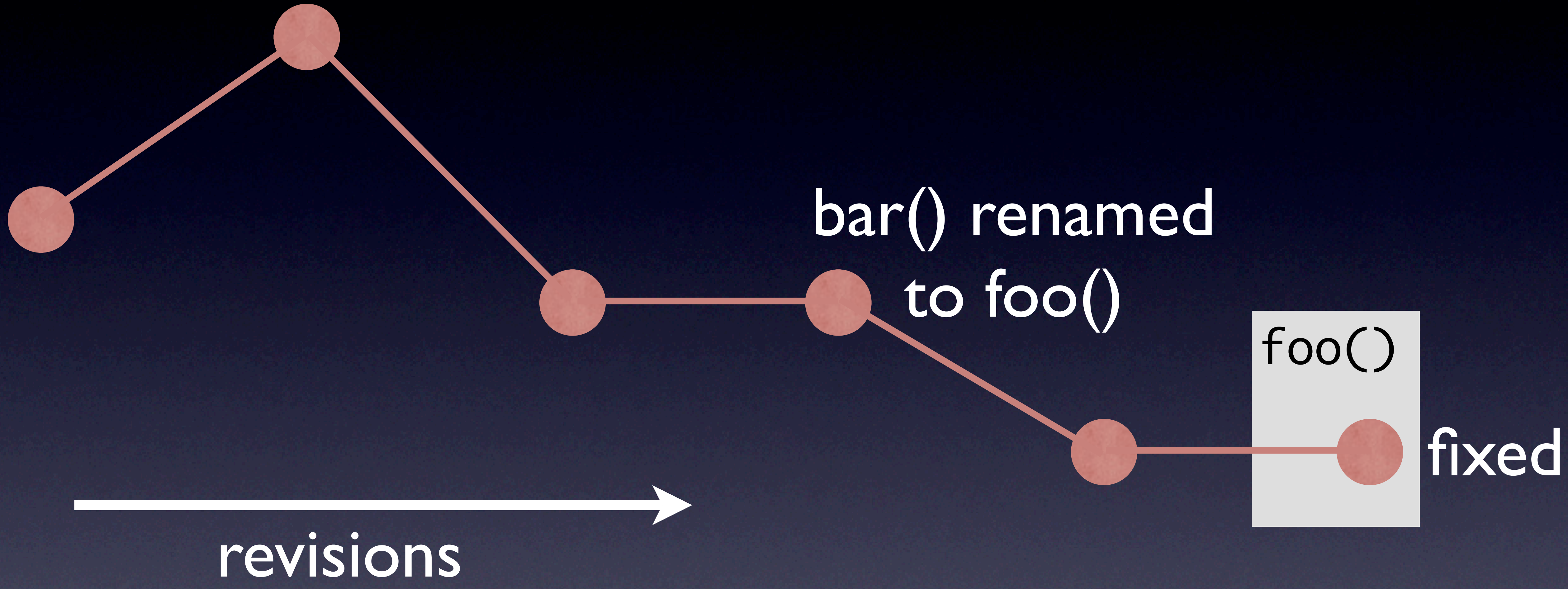
Annotation graphs



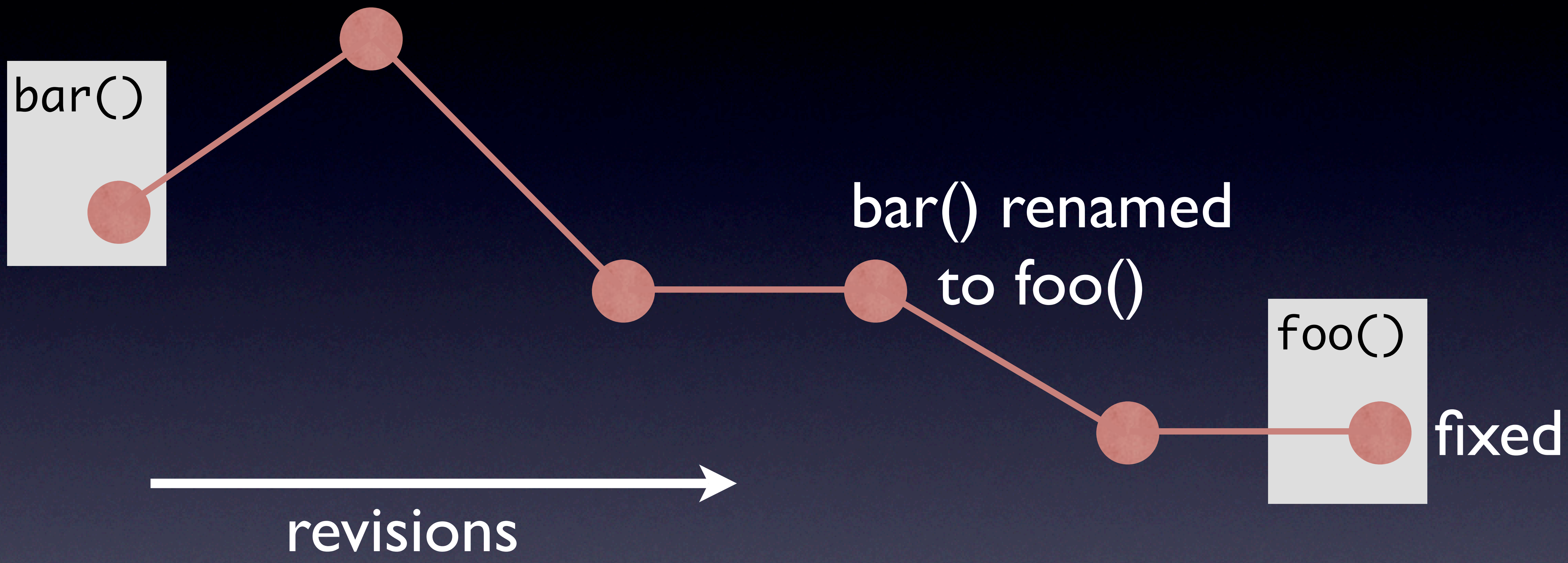
Annotation graphs



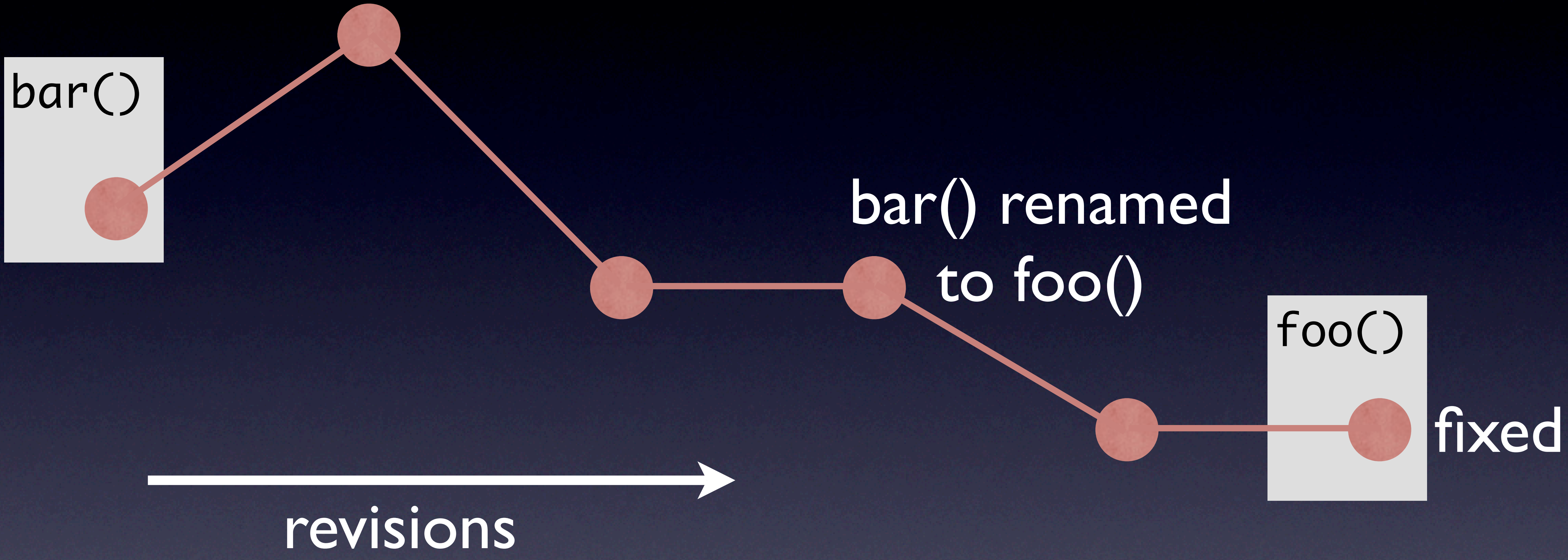
Annotation graphs



Annotation graphs

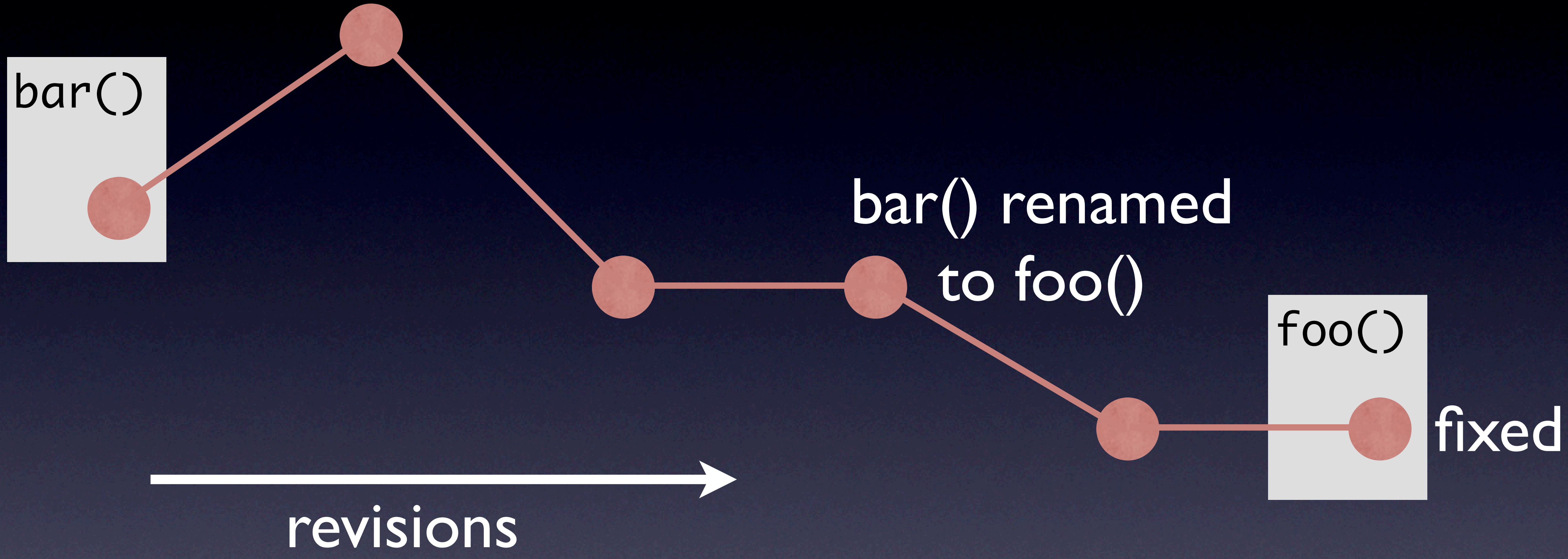


Annotation graphs



SZZ reports a bug-introducing change for `foo` (false positive) but not for `bar` (false negative).

Annotation graphs



Using annotation graphs we can remove 2% as false positives and identifies further 1%~4%.

Comments & blank lines

```
  public void notifySourceElementRequestor() {
-
+ if (reportReferenceInfo) {
+     notifyAllUnknownReferences();
+ }
  // collect the top level ast nodes
  int length = 0;
```

Comments & blank lines

```
  public void notifySourceElementRequestor() {
-
+ if (reportReferenceInfo) {
+     notifyAllUnknownReferences();
+ }
  // collect the top level ast nodes
  int length = 0;
```

Ignoring comments and blank lines removes 14%~20% as false positives.

Format changes

BUG-INTRODUCING

```
if ( a==true ) return;
```

Format changes

BUG-INTRODUCING

```
if ( a==true ) return;
```

```
if (a==true)
    return;
```

BUG FIX

```
if (a==false)
    return;
```

Format changes

BUG-INTRODUCING

```
if ( a==true ) return;
```

```
if (a==true)
    return;
```

FALSE POSITIVE

BUG FIX

```
if (a==false)
    return;
```

Format changes

BUG-INTRODUCING

```
if ( a==true ) return;
```

FALSE NEGATIVE

```
if (a==true)
    return;
```

FALSE POSITIVE

BUG FIX

```
if (a==false)
    return;
```

Format changes

BUG-INTRODUCING

```
if ( a==true ) return;
```

FALSE NEGATIVE

```
if (a==true)
     return;
```

FALSE POSITIVE

BUG FIX

```
if (a==false)
     return;
```

Ignoring format changes removes 18%~25% as false positives and identifies further 13%~14%.

Fixes that affect many files

Most large fixes are refactoring

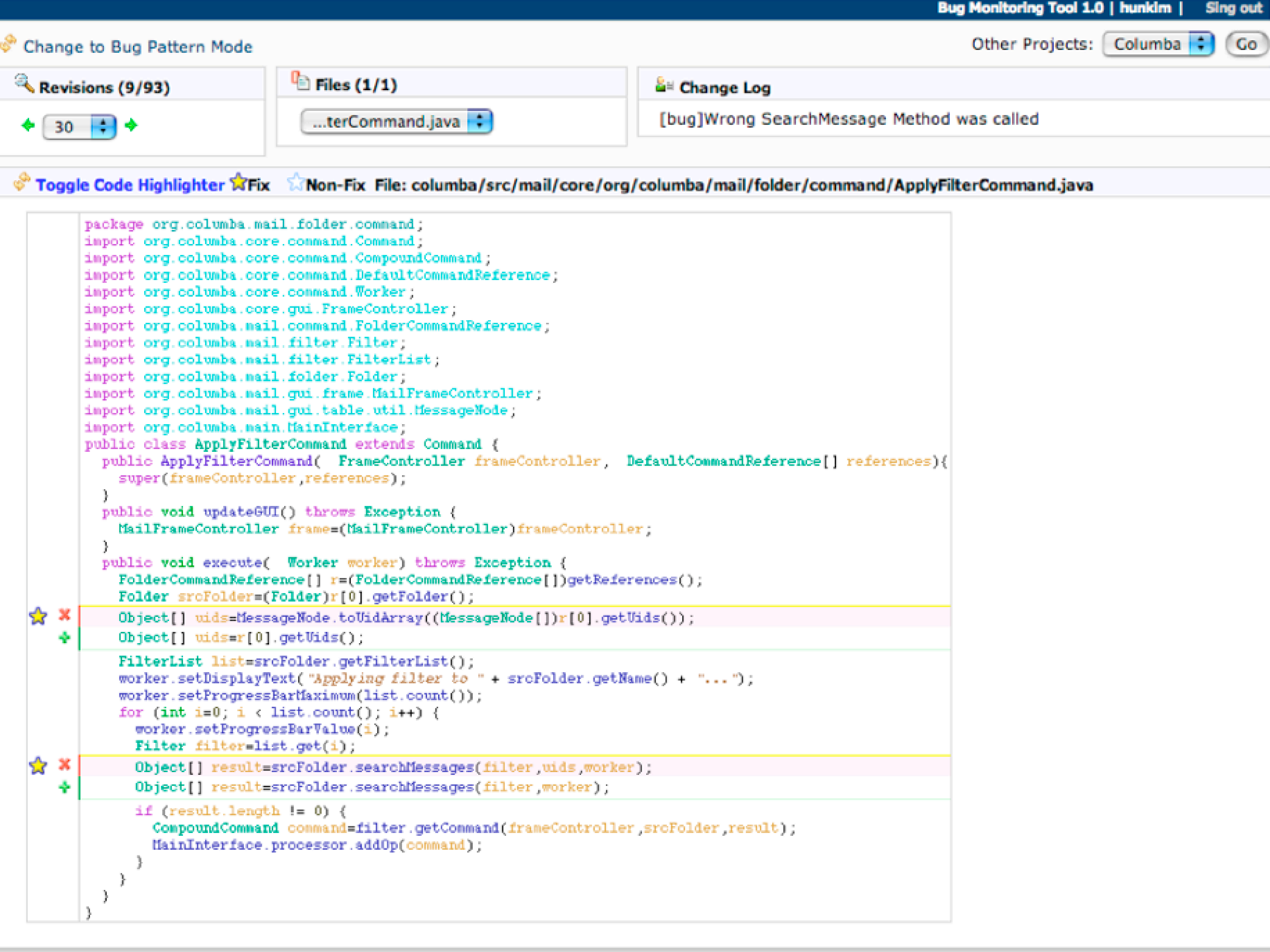
```
- public boolean visit(TypeDeclaration
-   typeDeclaration, BlockScope scope){
+ public boolean visit(LocalTypeDeclaration
+  typeDeclaration, BlockScope scope){
```

Fixes that affect many files

Most large fixes are refactoring

```
- public boolean visit(TypeDeclaration
-    typeDeclaration, BlockScope scope){
+ public boolean visit(LocalTypeDeclaration
+   typeDeclaration, BlockScope scope){
```

Ignoring fixes that affect many files
(=more than five times the median)
removes 7%~16% as false positives


Bug Monitoring Tool 1.0 | hunkim | Sing out
Change to Bug Pattern Mode
Other Projects: Columba Go
Revisions (9/93)
30
Files (1/1)
...terCommand.java
Change Log
Toggle Code Highlighter Fix Non-Fix

[bug]Wrong SearchMessage Method was called

File: columba/src/mail/core/org/columba/mail/folder/command/ApplyFilterCommand.java

```
package org.columba.mail.folder.command;
import org.columba.core.command.Command;
import org.columba.core.command.CompoundCommand;
import org.columba.core.command.DefaultCommandReference;
import org.columba.core.command.Worker;
import org.columba.core.gui.FrameController;
import org.columba.mail.command.FolderCommandReference;
import org.columba.mail.filter.Filter;
import org.columba.mail.filter.FilterList;
import org.columba.mail.folder.Folder;
import org.columba.mail.gui.frame.MailFrameController;
import org.columba.mail.gui.table.util.MessageNode;
import org.columba.main.MainInterface;
public class ApplyFilterCommand extends Command {
  public ApplyFilterCommand( FrameController frameController,  DefaultCommandReference[] references){
    super(frameController,references);
  }
  public void updateGUI() throws Exception {
    MailFrameController frame=(MailFrameController)frameController;
  }
  public void execute(  Worker worker) throws Exception {
    FolderCommandReference[] r=(FolderCommandReference[])getReferences();
    Folder srcFolder=(Folder)r[0].getFolder();
-   Object[] uids=MessageNode.toUidArray((MessageNode[])r[0].getUids());
+   Object[] uids=r[0].getUids();
    FilterList list=srcFolder.getFilterList();
    worker.setDisplayText("Applying filter to " + srcFolder.getName() + "...");
    worker.setProgressBarMaximum(list.count());
    for (int i=0; i < list.count(); i++) {
      worker.setProgressBarValue(i);
      Filter filter=list.get(i);
-     Object[] result=srcFolder.searchMessages(filter,uids,worker);
+     Object[] result=srcFolder.searchMessages(filter,worker);
      if (result.length != 0) {
        CompoundCommand command=filter.getCommand(frameController,srcFolder,result);
        MainInterface.processor.addOp(command);
      }
    }
  }
}
```

Manual inspection of fixes

Two judges check whether a fix is actually a fix.

```
  deleteResources(actualNonJavaResources,fForce);
- IResource[] remaingFiles;
+ IResource[] remainingFiles;
  try {
-      remaingFiles=((IFolder)res).members();
+      remainingFiles=((IFolder)res).members();
  }
```

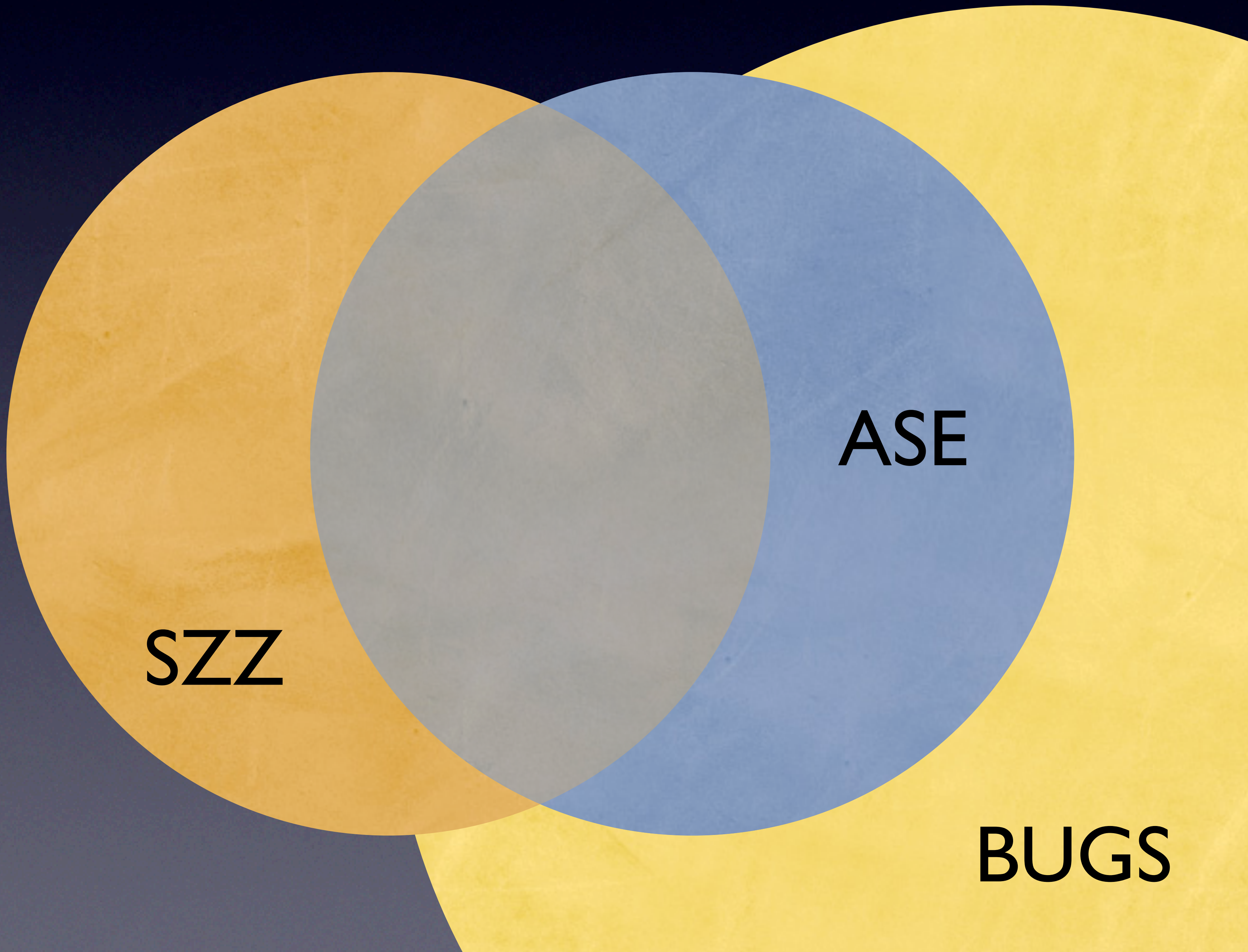
Manual inspection of fixes

Two judges check whether a fix is actually a fix.

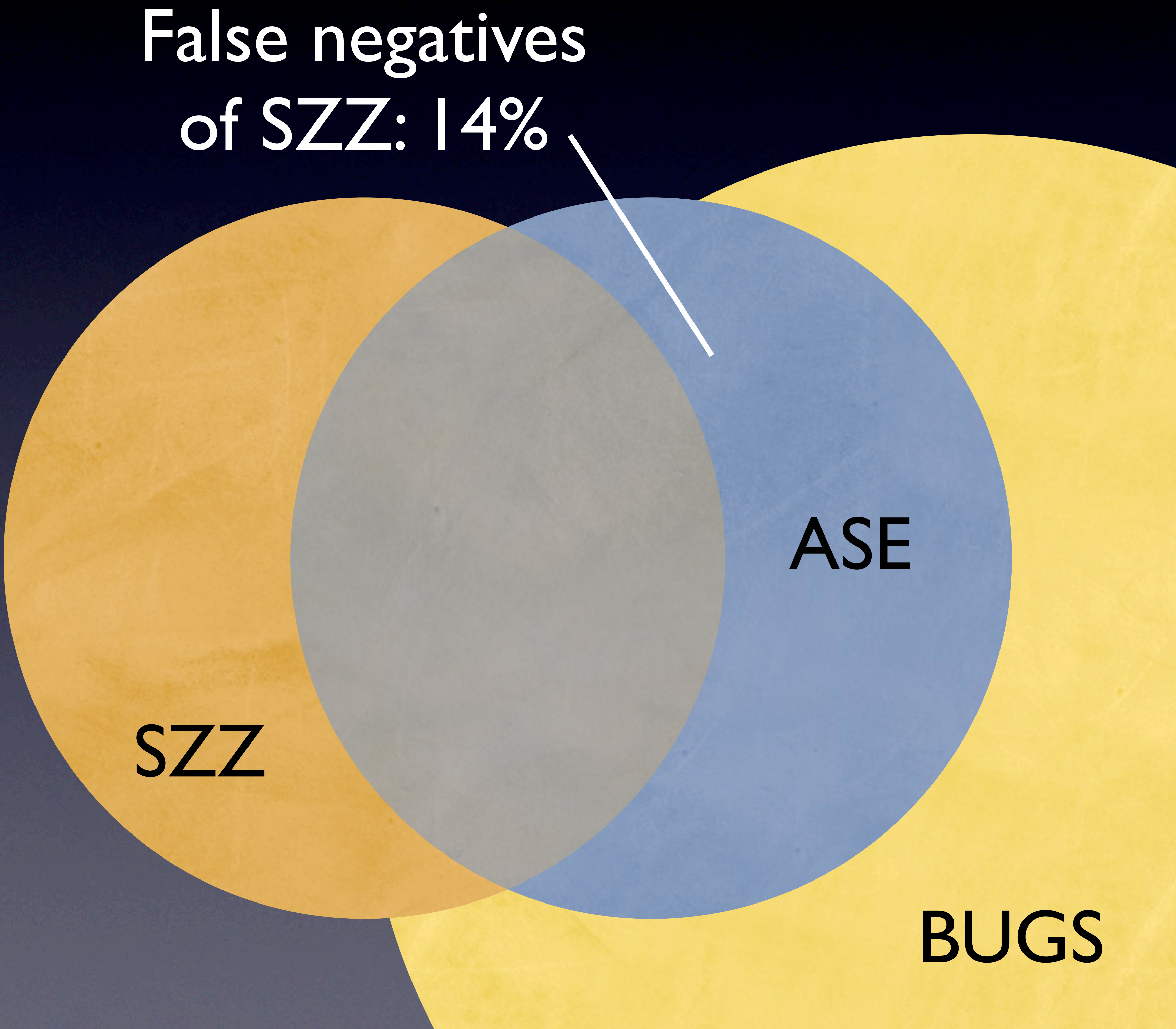
```
  deleteResources(actualNonJavaResources,fForce);
- IResource[] remaingFiles;
+ IResource[] remainingFiles;
  try {
-       remaingFiles=((IFolder)res).members();
+       remainingFiles=((IFolder)res).members();
  }
```

Manual inspection of fixes removes only 4%~5% as false positives.

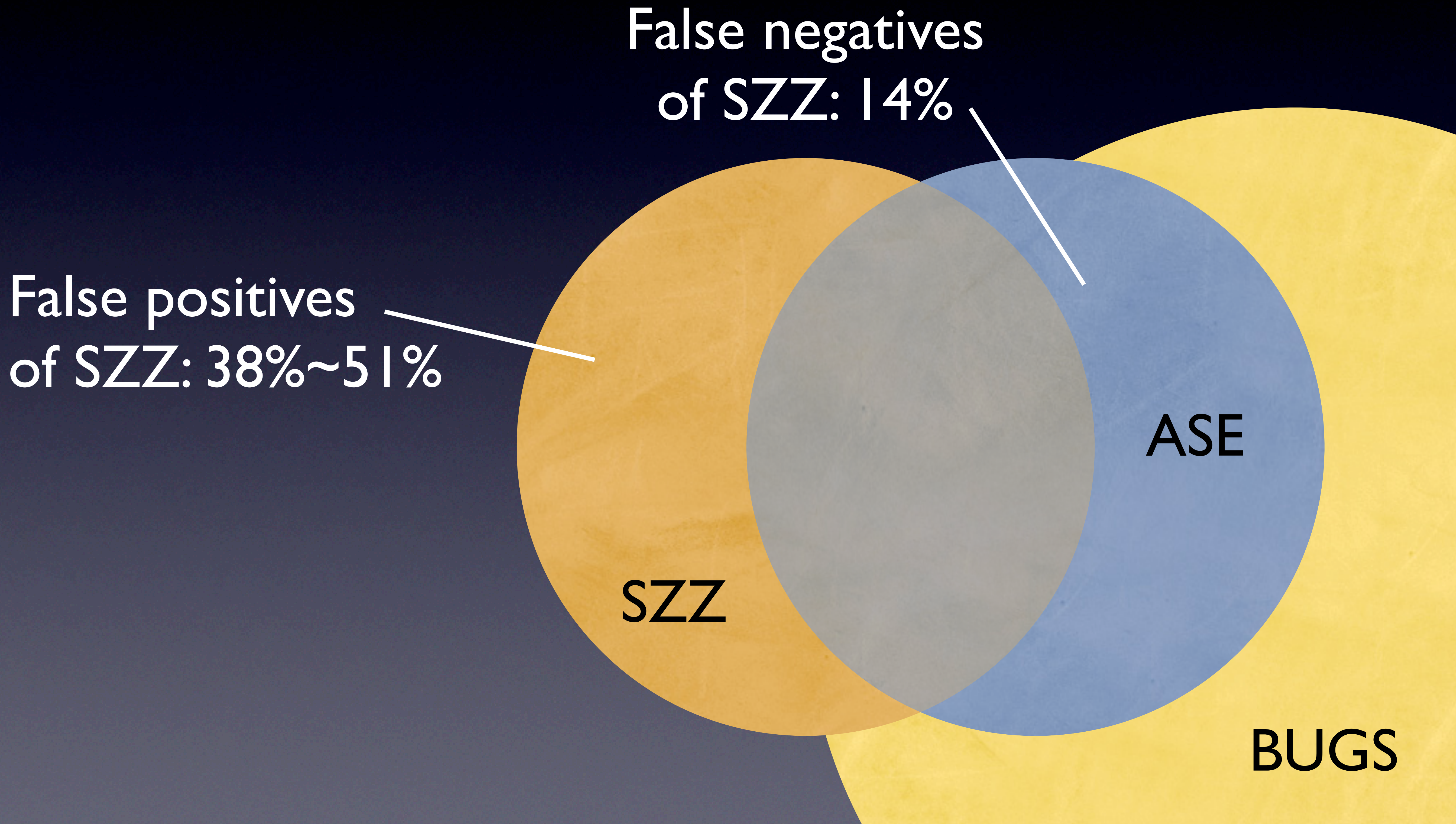
Putting it together



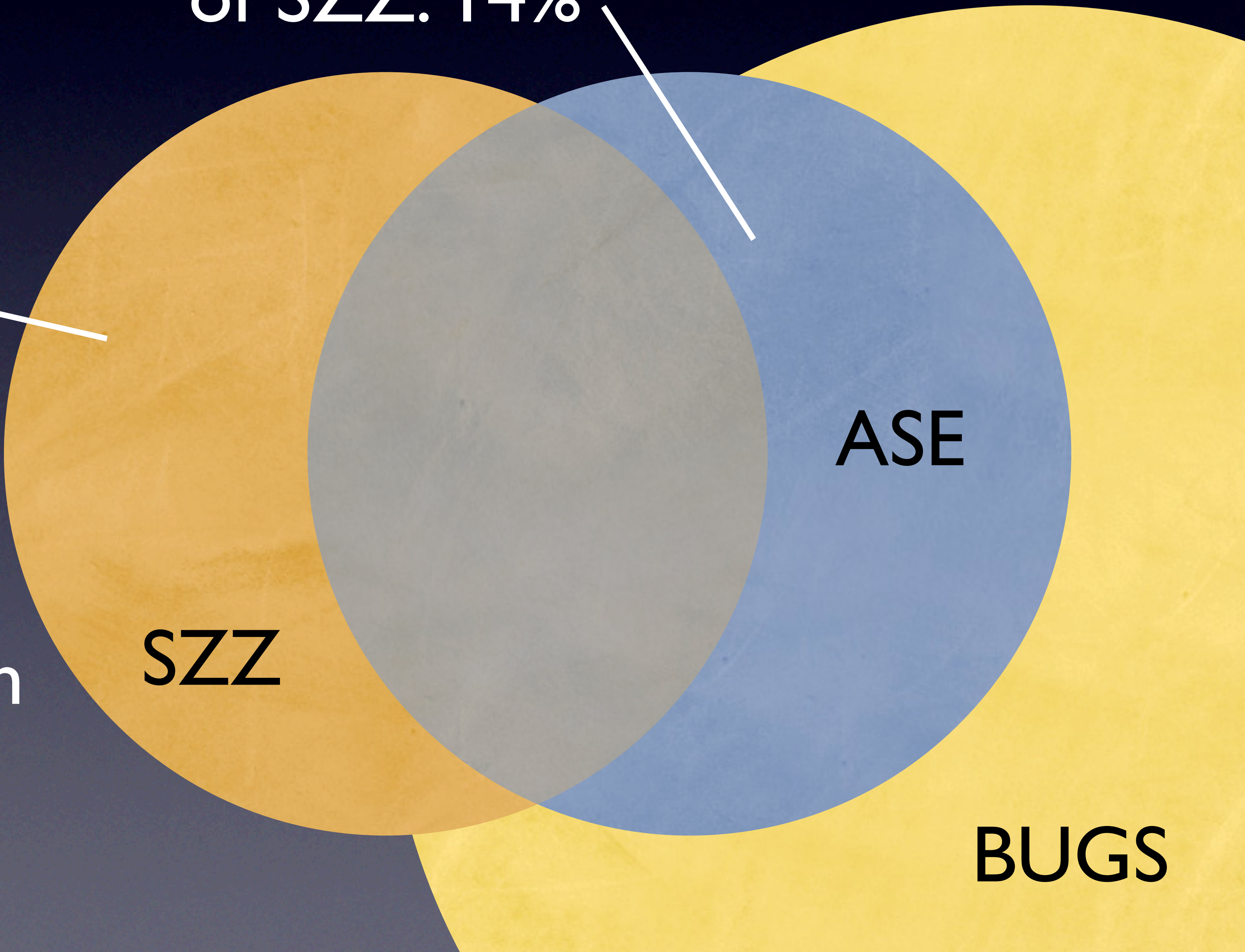
Putting it together



Putting it together



Putting it together

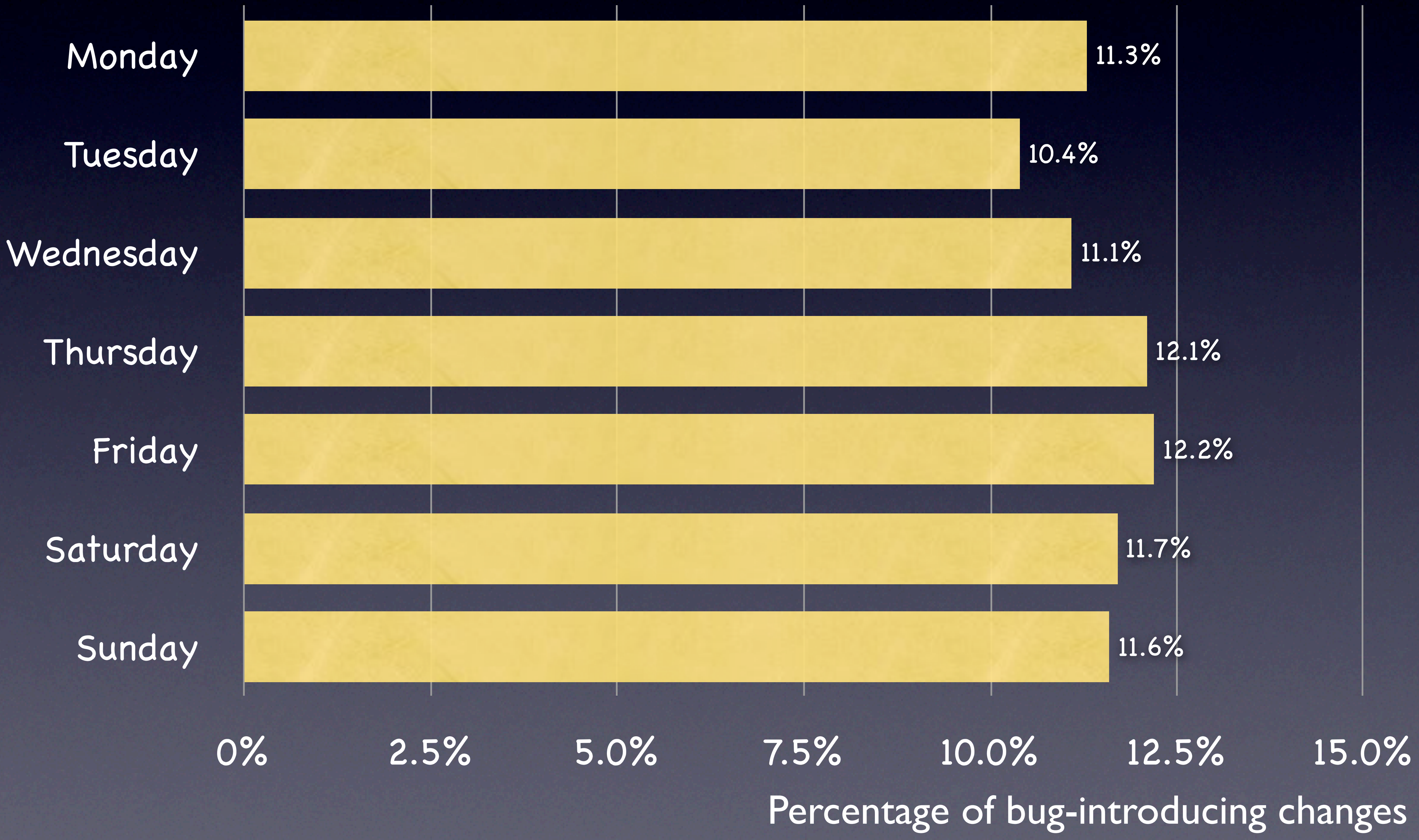


False negatives of SZZ: 14%

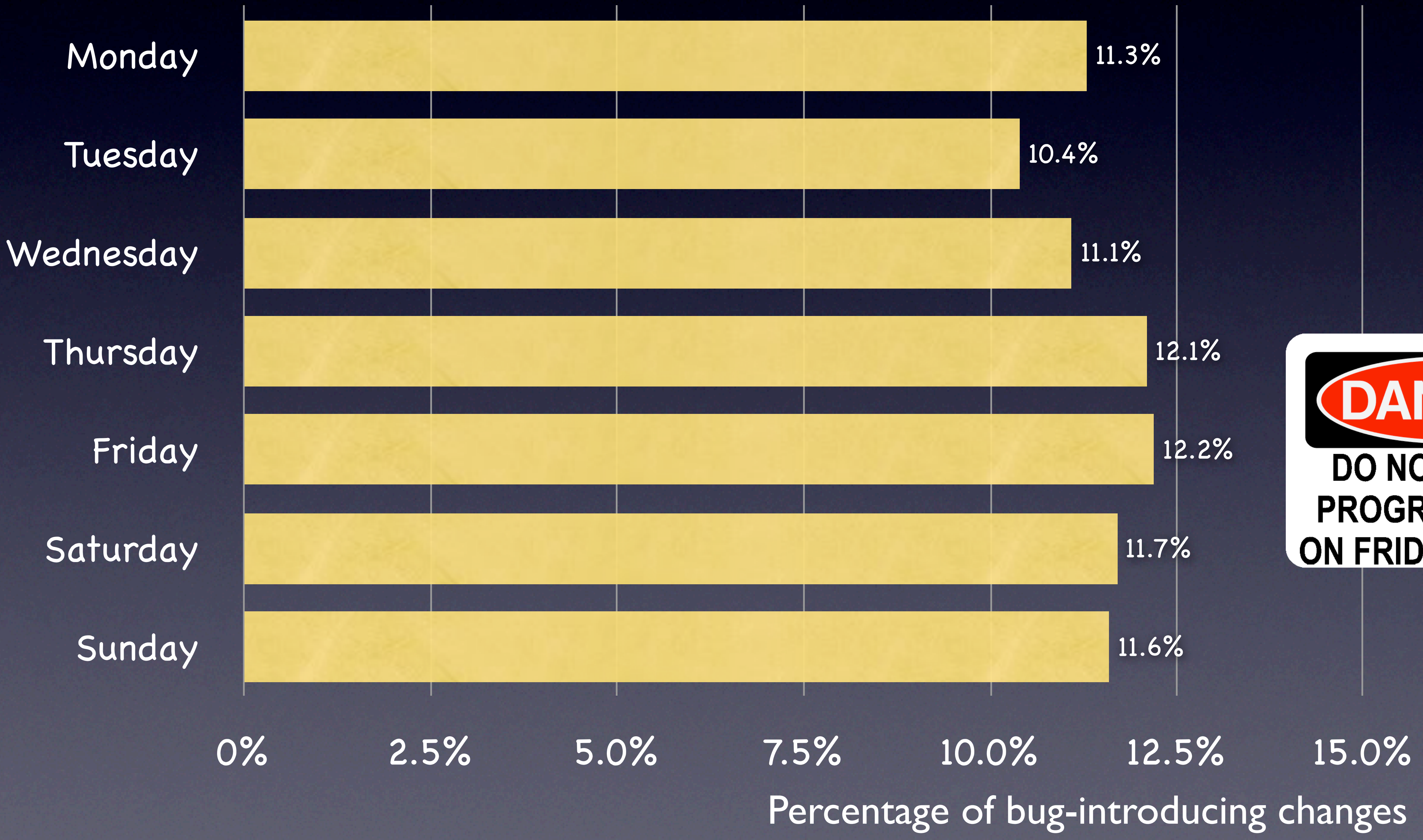
False positives of SZZ: 38%~51%

Automatically we can remove 36%~48% as false positive.

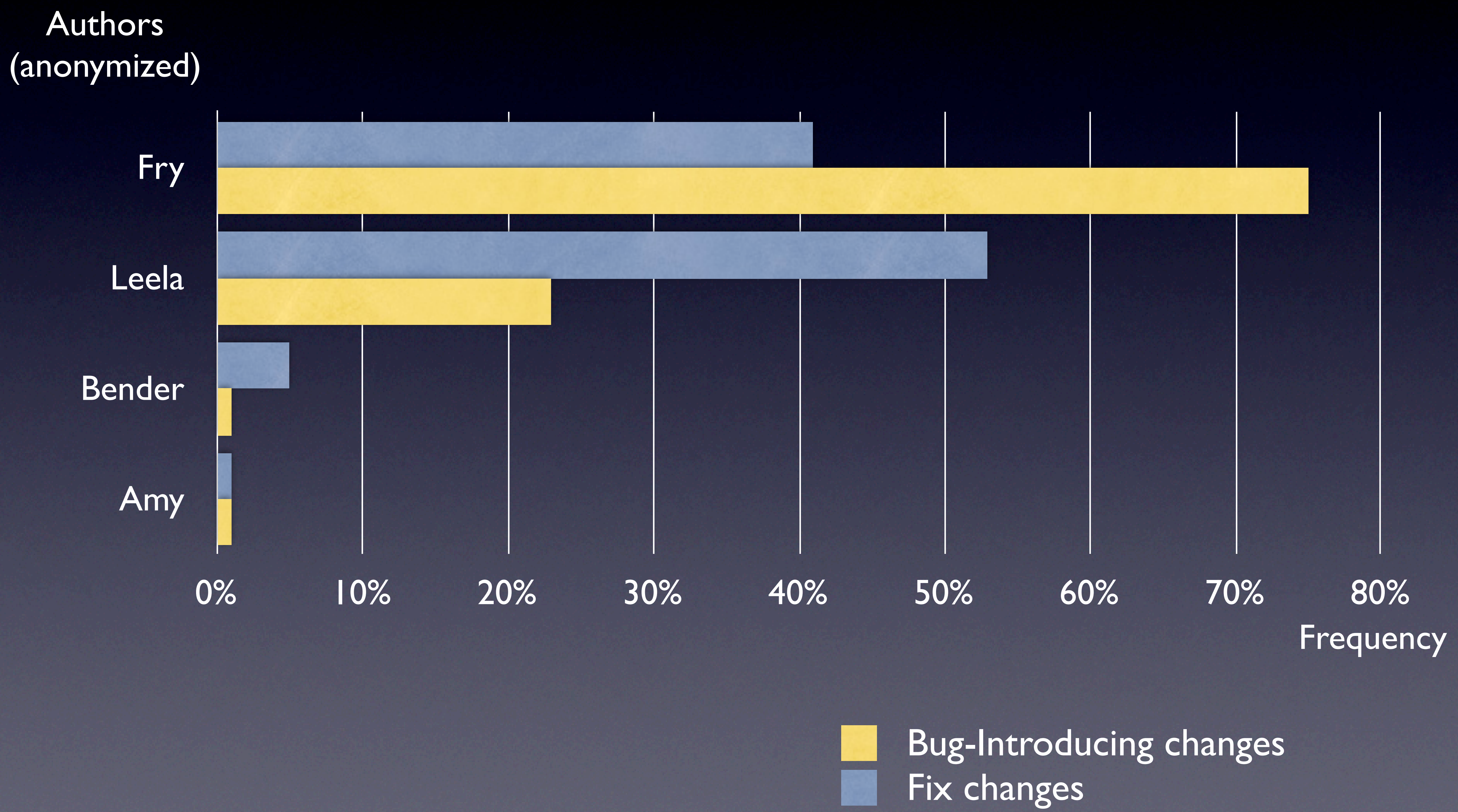
Don’t program on Fridays ;-)



Don’t program on Fridays ;-)



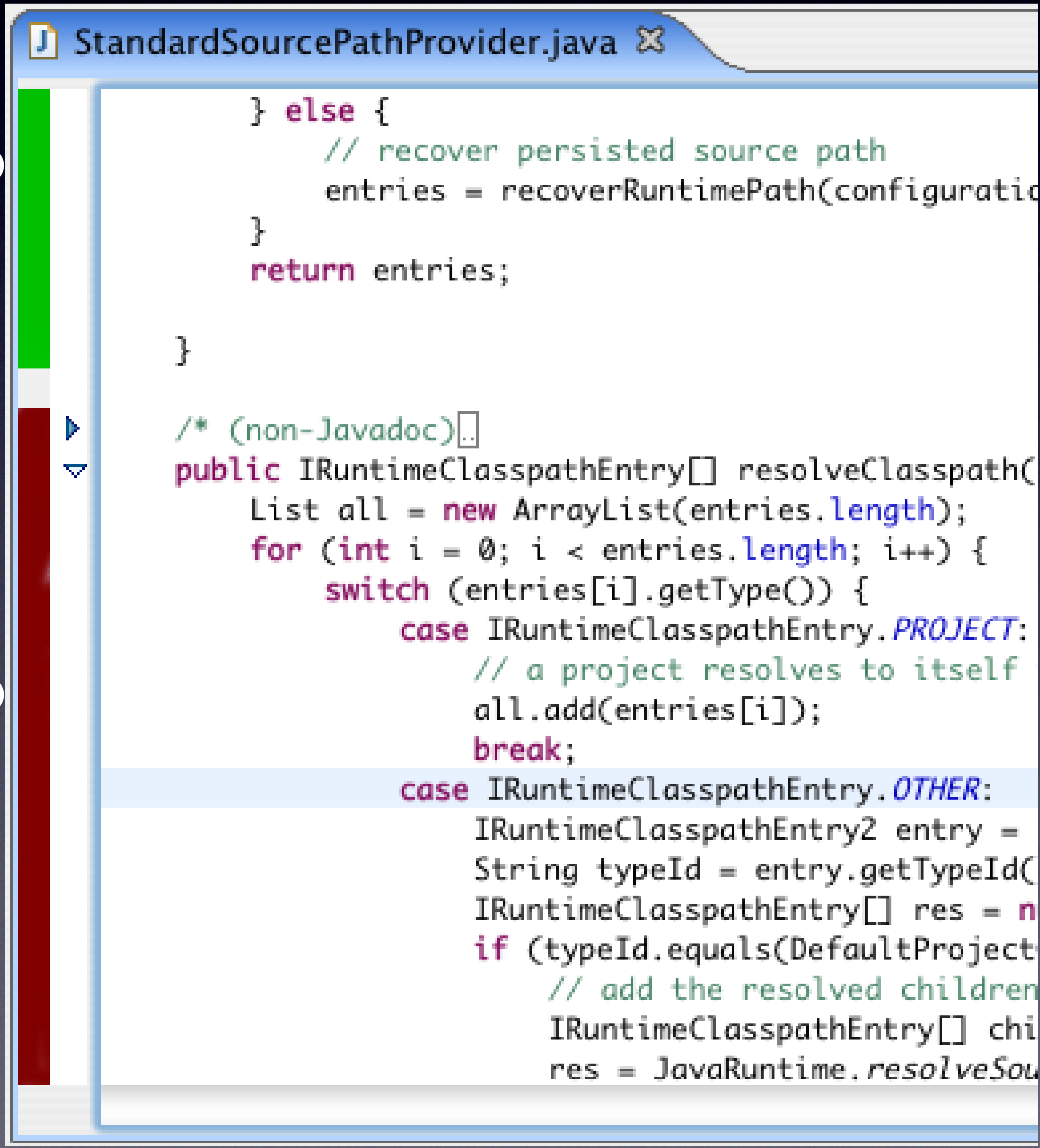
Defect-prone authors



Risk awareness

"Safe" Location (green)

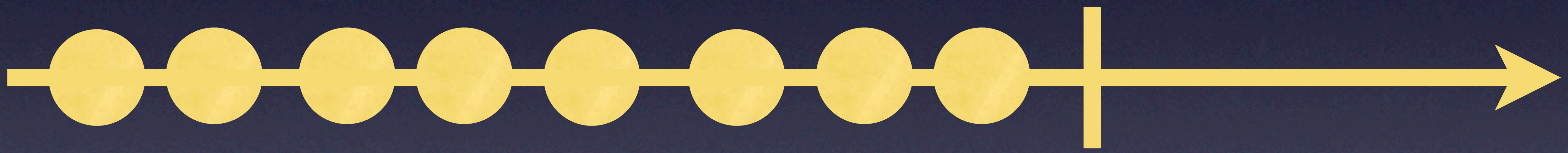
Risky Location (dark red)



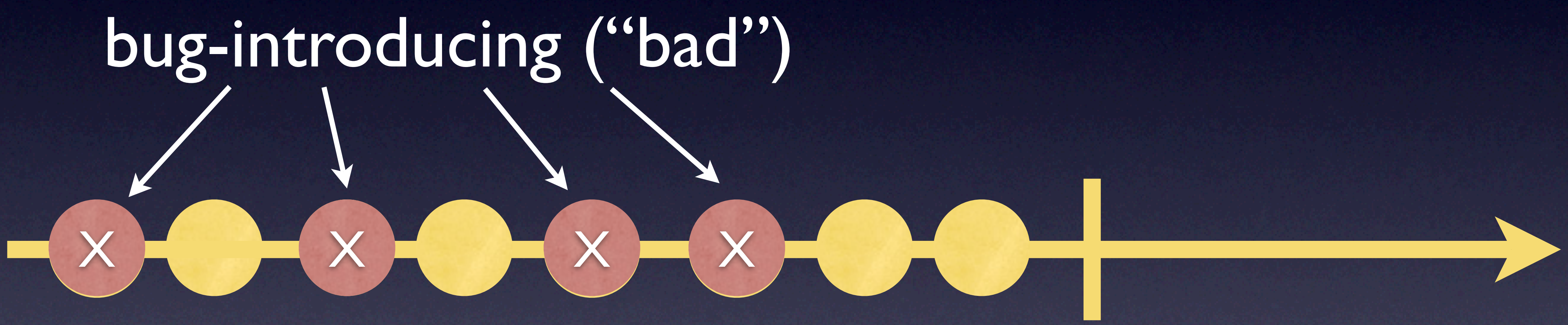
StandardSourcePathProvider.java

```
        } else {
            // recover persisted source path
            entries = recoverRuntimePath(configuratio
        }
        return entries;

    }

    /* (non-Javadoc)
    public IRuntimeClasspathEntry[] resolveClasspath(
        List all = new ArrayList(entries.length);
        for (int i = 0; i < entries.length; i++) {
            switch (entries[i].getType()) {
                case IRuntimeClasspathEntry.PROJECT:
                    // a project resolves to itself
                    all.add(entries[i]);
                    break;
                case IRuntimeClasspathEntry.OTHER:
                    IRuntimeClasspathEntry2 entry =
                    String typeId = entry.getTypeId(
                    IRuntimeClasspathEntry[] res = n
                    if (typeId.equals(DefaultProject
                        // add the resolved children
                        IRuntimeClasspathEntry[] chi
                        res = JavaRuntime.resolveSou
```

Change classification

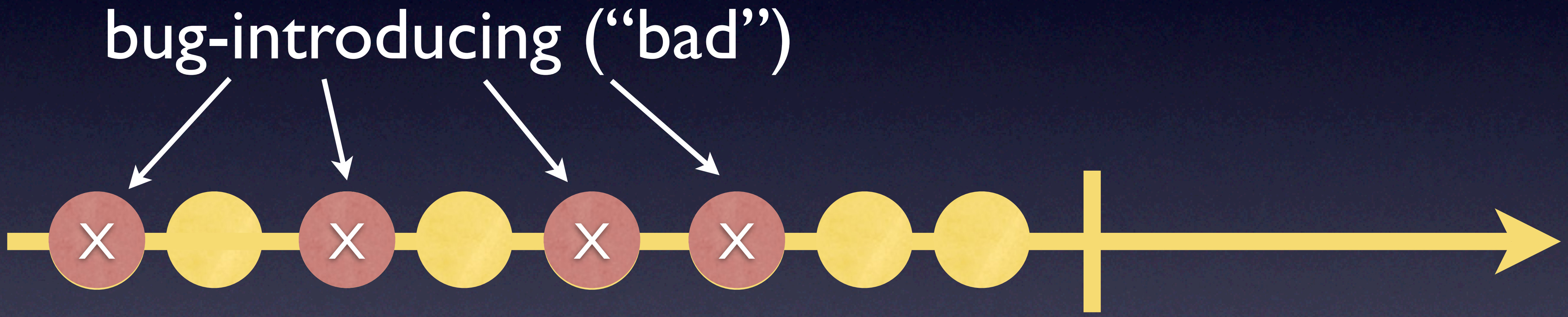


Change classification



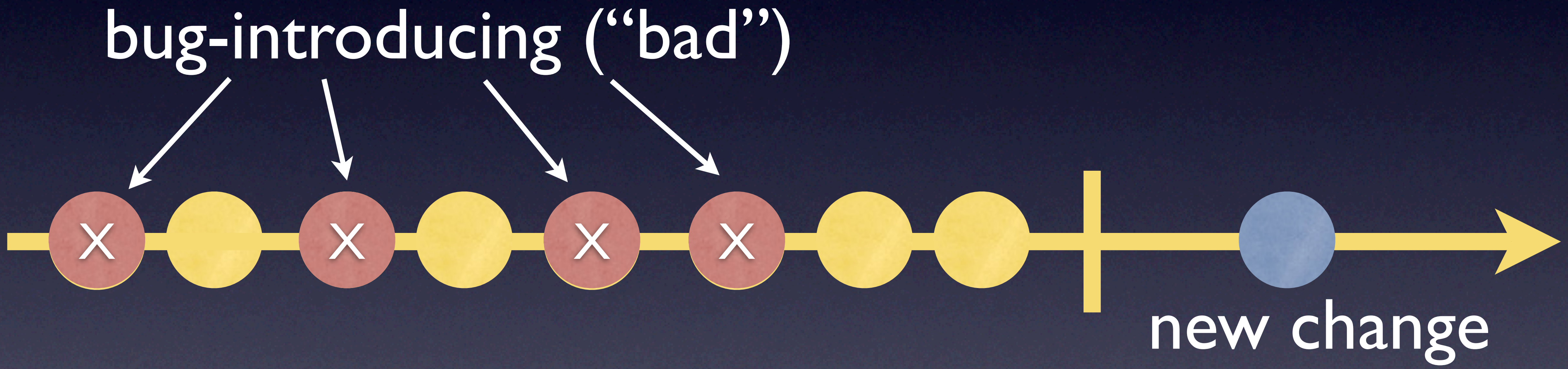
Change classification

BUILD A CLASSIFIER



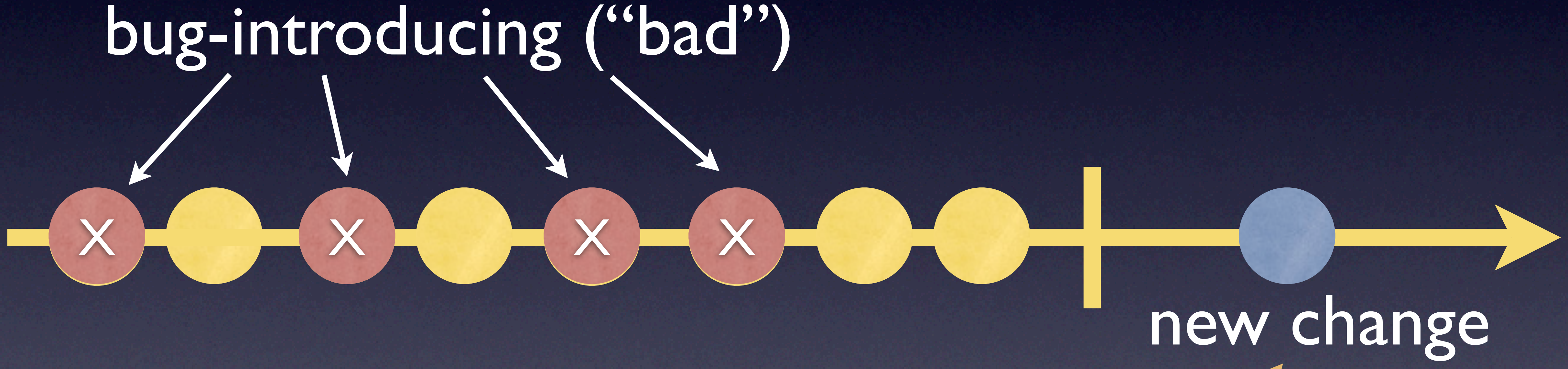
Change classification

BUILD A CLASSIFIER



Change classification

BUILD A CLASSIFIER



PREDICT QUALITY

Conclusions

Conclusions

- Bug-introducing changes tell <u>when</u> a defect was introduced, not only its location.

Conclusions

- Bug-introducing changes tell <u>when</u> a defect was introduced, not only its location.
- We can automatically identify
  – 36%~48% of SZZ as false positives and
  – further 14% of missed bug-introductions.

Conclusions

- Bug-introducing changes tell <u>when</u> a defect was introduced, not only its location.
- We can automatically identify
  – 36%~48% of SZZ as false positives and
  – further 14% of missed bug-introductions.
- Bug-introducing changes are useful for defect prediction and software evolution.

More mining @ ASE:
Friday morning.

Conclusions

- Bug-introducing changes tell <u>when</u> a defect was introduced, not only its location.
- We can automatically identify
  – 36%~48% of SZZ as false positives and
  – further 14% of missed bug-introductions.
- Bug-introducing changes are useful for defect prediction and software evolution.