Where do Bugs Come From?

Andreas Zeller
Saarland University

bug.c

```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
```

Boskoop: bug (~/tmp/bug) <zeller.zeller> — bash — 80x24 — ⌘1

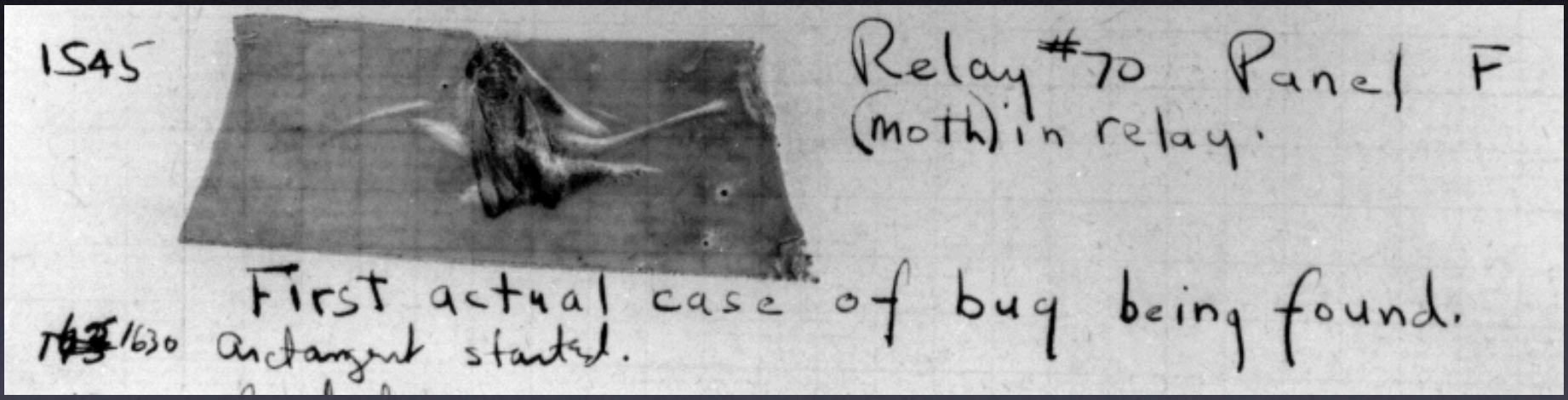
```
$
```

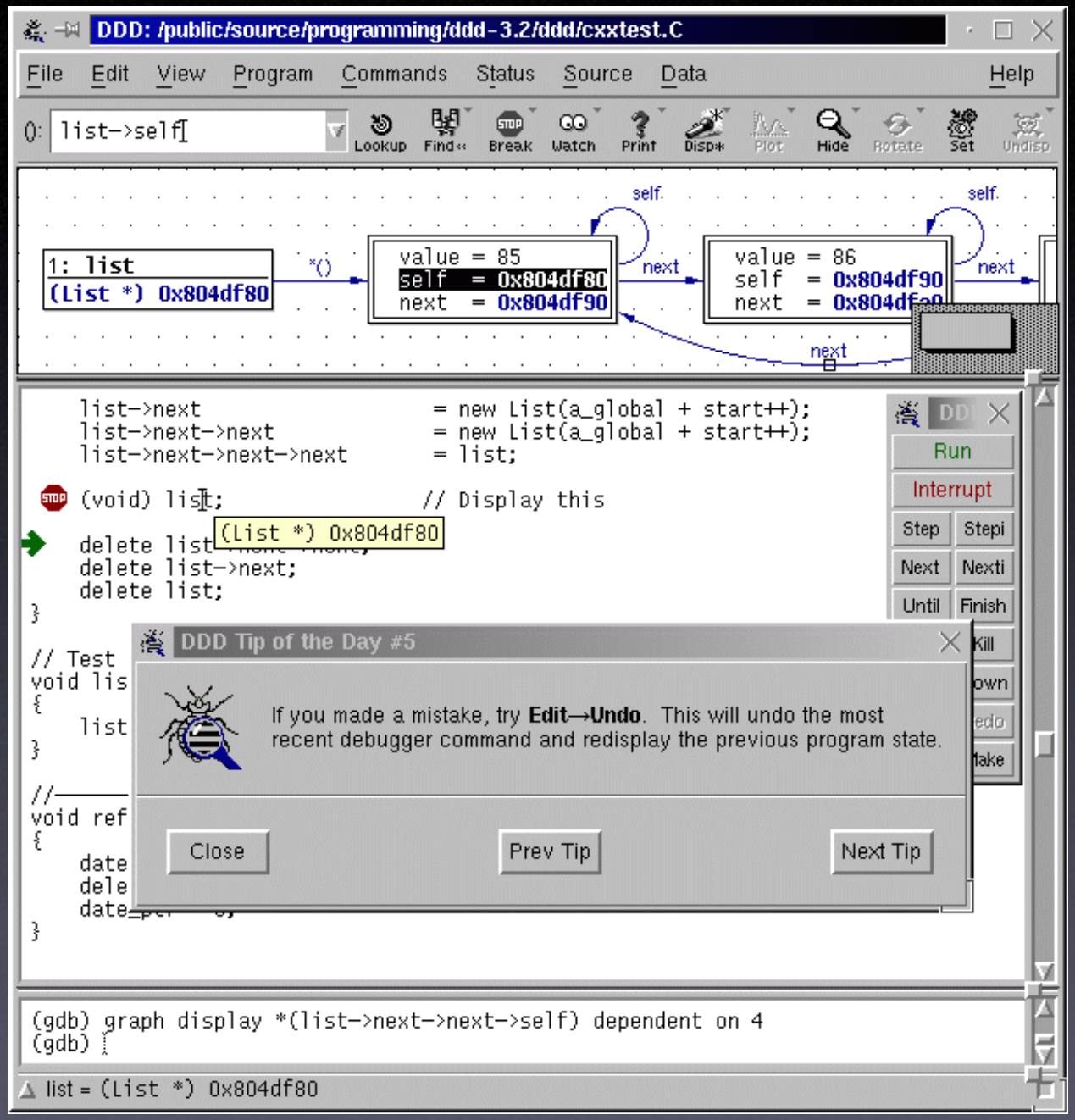
Where does this bug come from?

Mama bug, papa bug, they all go back to the mother of all bugs...

The First Bug

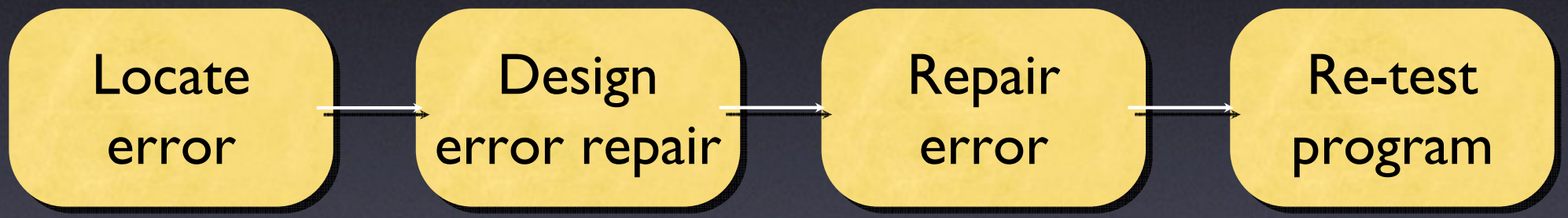
(September 9, 1947)

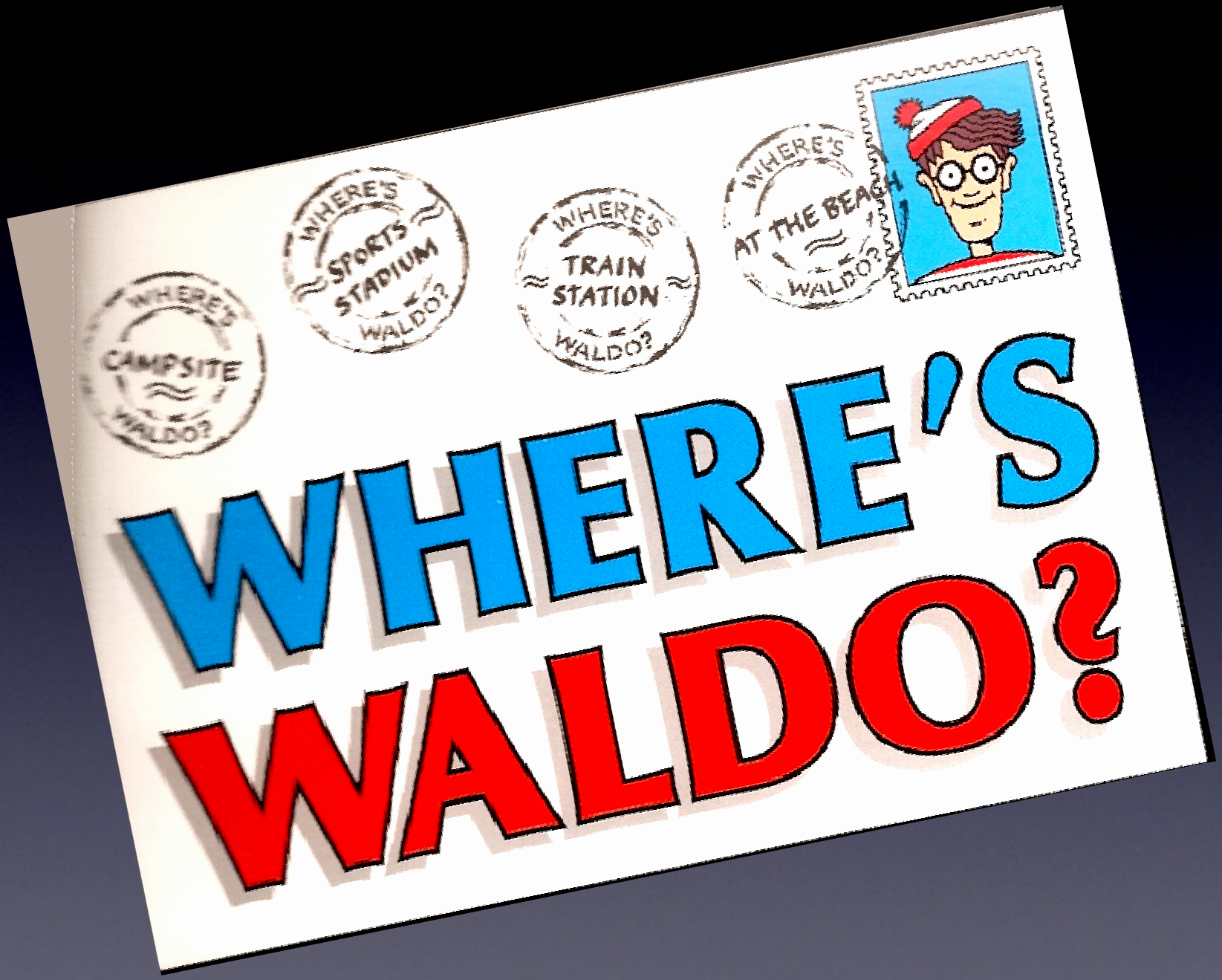


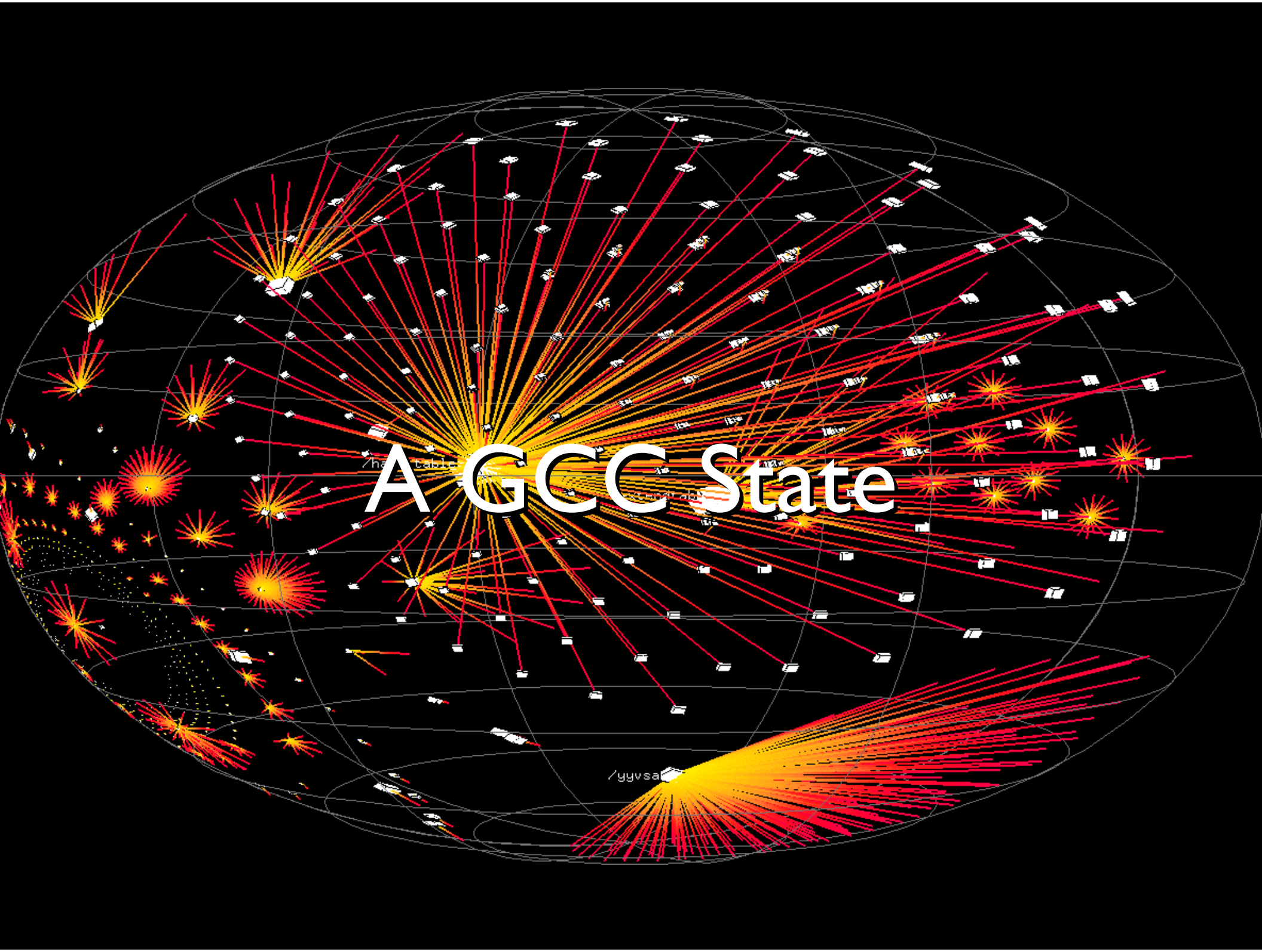

DDD: /public/source/programming/ddd-3.2/ddd/cxxtest.C
File Edit View Program Commands Status Source Data Help
list->self
Lookup Find« Break Watch Print Disp* Plot Hide Rotate Set Undisp
1: list
(List *) 0x804df80
value = 85
self = 0x804df80
next = 0x804df90
value = 86
self = 0x804df90
next = 0x804dfa0
self
next
list->next = new List(a_global + start++);
list->next->next = new List(a_global + start++);
list->next->next->next = list;
(void) list; // Display this
(List *) 0x804df80
delete list;
delete list->next;
delete list;
Run
Interrupt
Step Stepi
Next Nexti
Until Finish
DDD Tip of the Day #5
If you made a mistake, try Edit→Undo. This will undo the most recent debugger command and redisplay the previous program state.
Close
Prev Tip
Next Tip
(gdb) graph display *(list->next->next->self) dependent on 4
(gdb)
list = (List *) 0x804df80

How to Debug

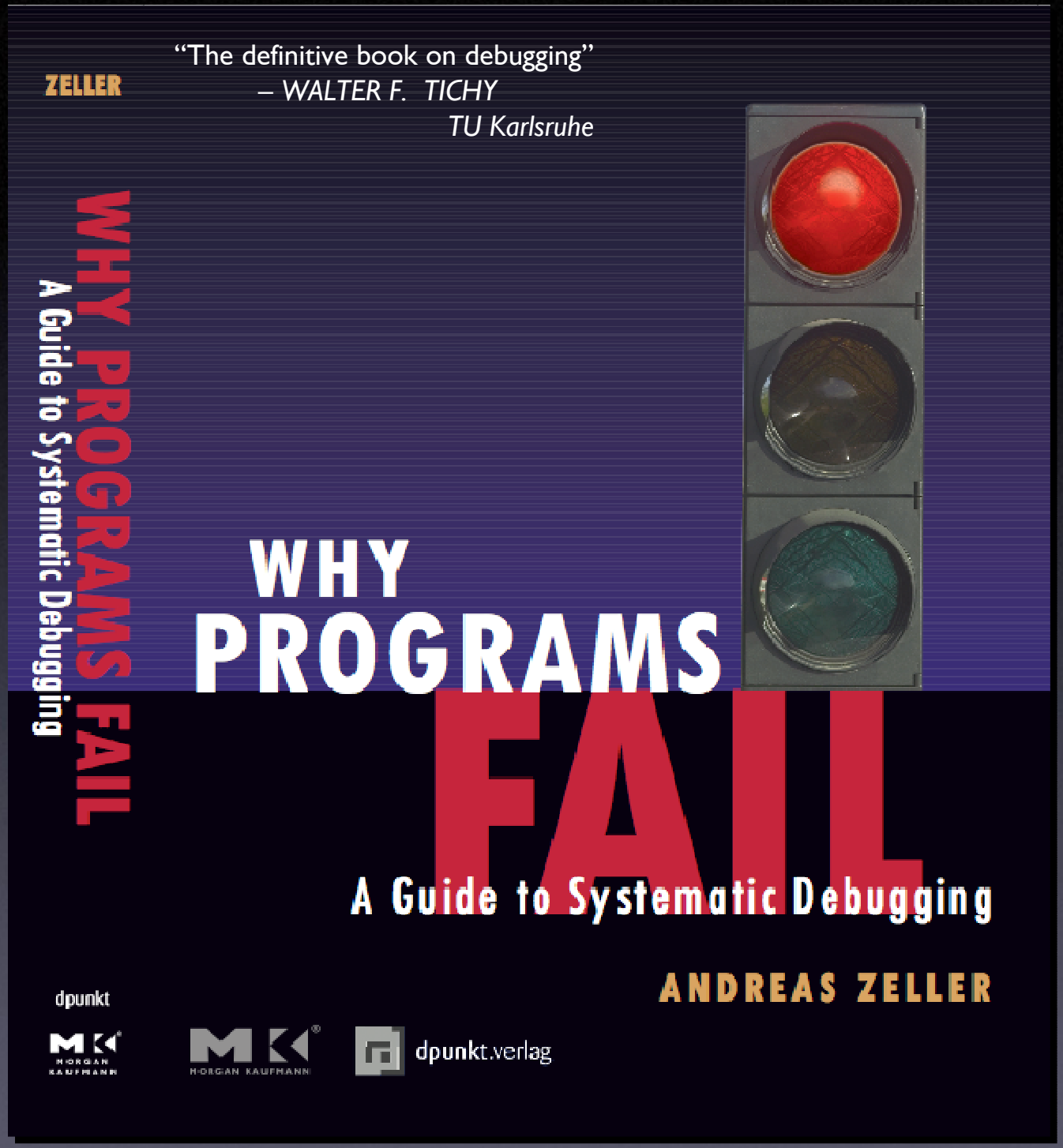
(Sommerville 2004)




WHERE'S CAMPSITE WALDO?
WHERE'S SPORTS STADIUM WALDO?
WHERE'S TRAIN STATION WALDO?
WHERE'S AT THE BEACH WALDO?
WHERE'S WALDO?



A GCC State


"The definitive book on debugging"
– WALTER F. TICHY
TU Karlsruhe
ZELLER
WHY PROGRAMS FAIL
A Guide to Systematic Debugging
WHY
PROGRAMS
FAIL
A Guide to Systematic Debugging
ANDREAS ZELLER
dpunkt
MORGAN KAUFMANN
dpunkt.verlag

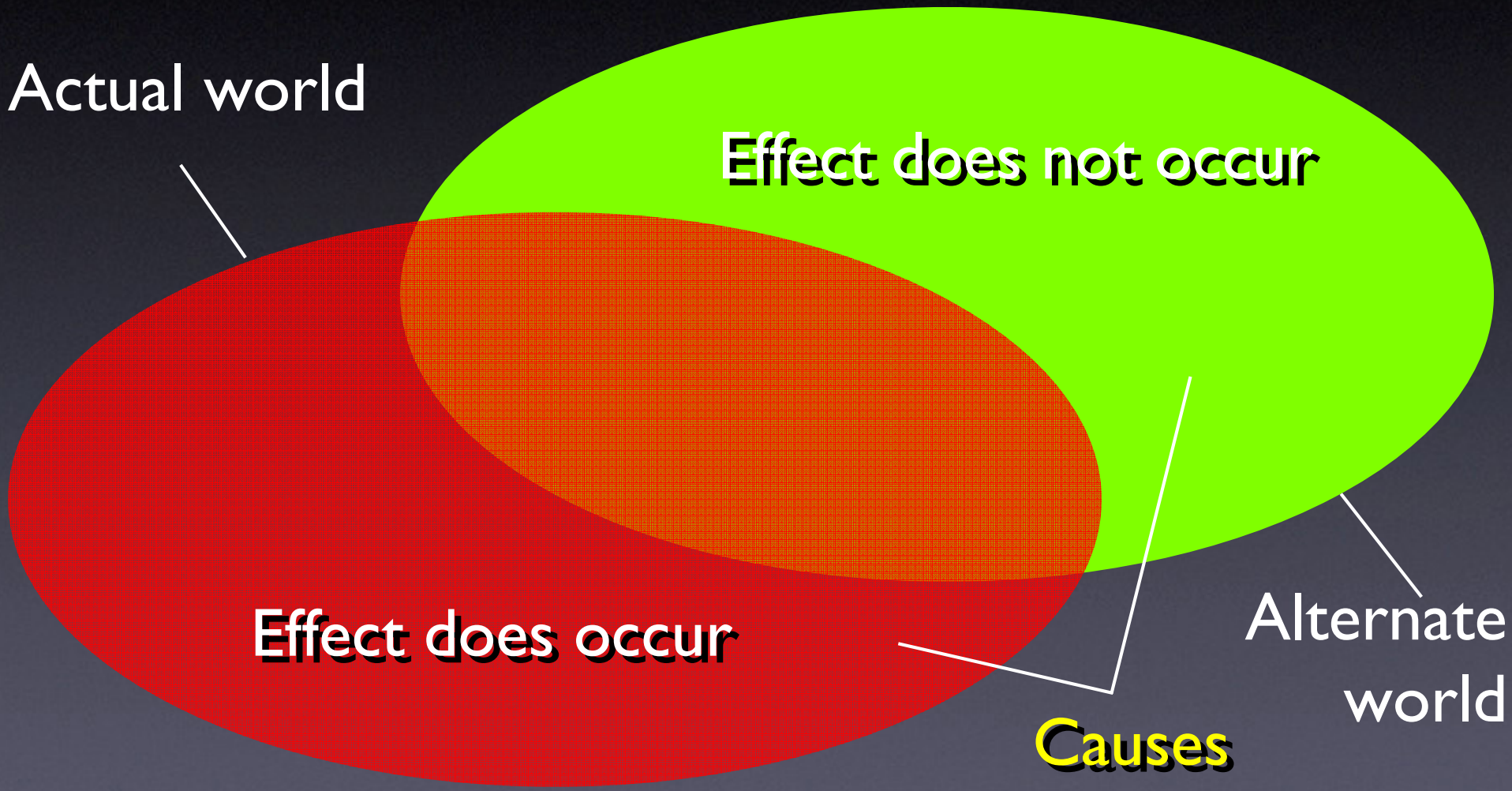
Boskoop: bug (~/tmp/bug) <zeller.zeller> — bash — 80x24 — ⌘1

```
$ ls
bug.c
$ gcc-2.95.2 -O bug.c
gcc: Internal error: program cc1 got fatal signal 11
Segmentation fault
$
```

What is the *cause* of this failure?

Mama bug, papa bug, they all go back to the mother of all bugs...

Counterfactual Causality



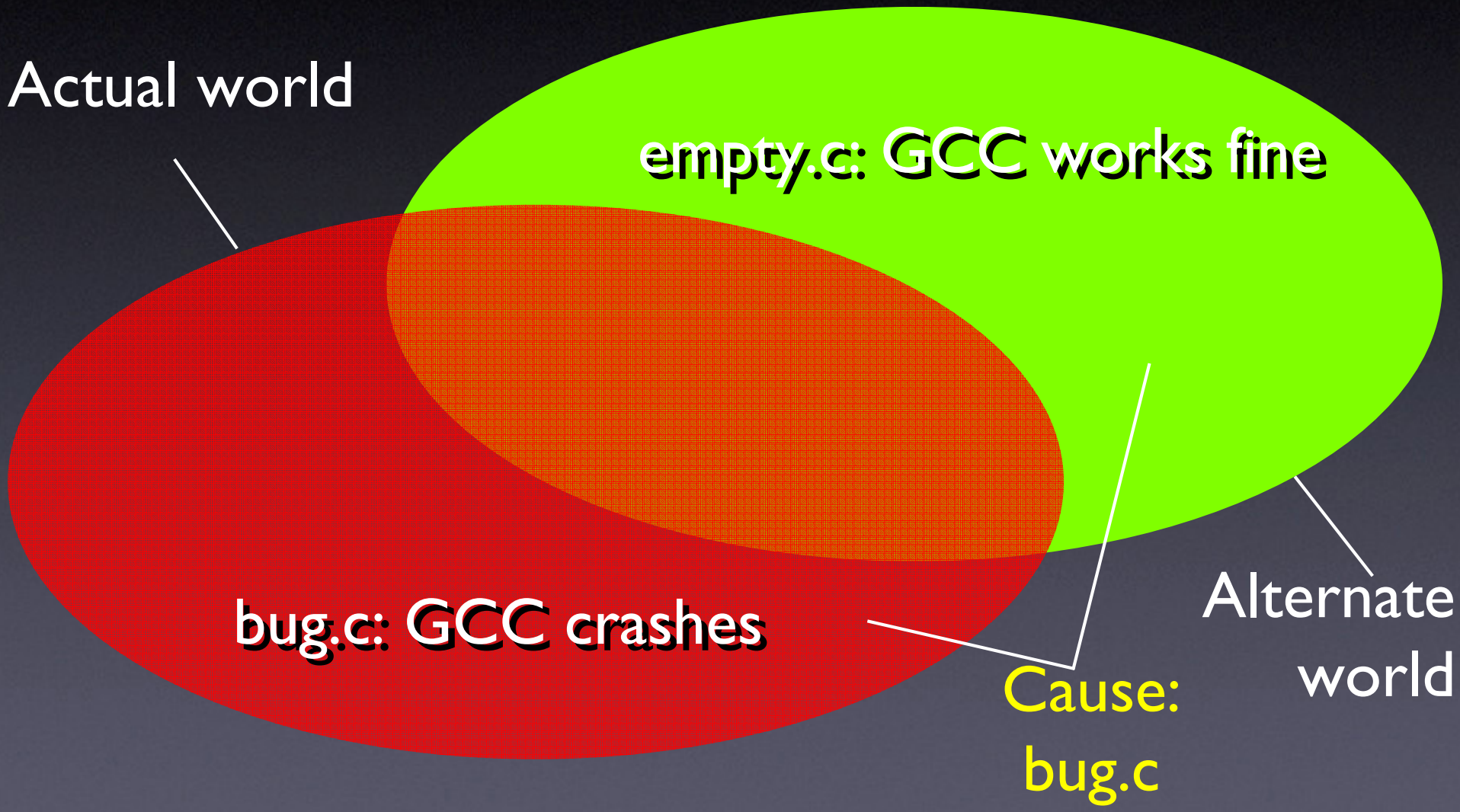
bug.c

```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0
    }
    return z[n];
}
```

empty.c

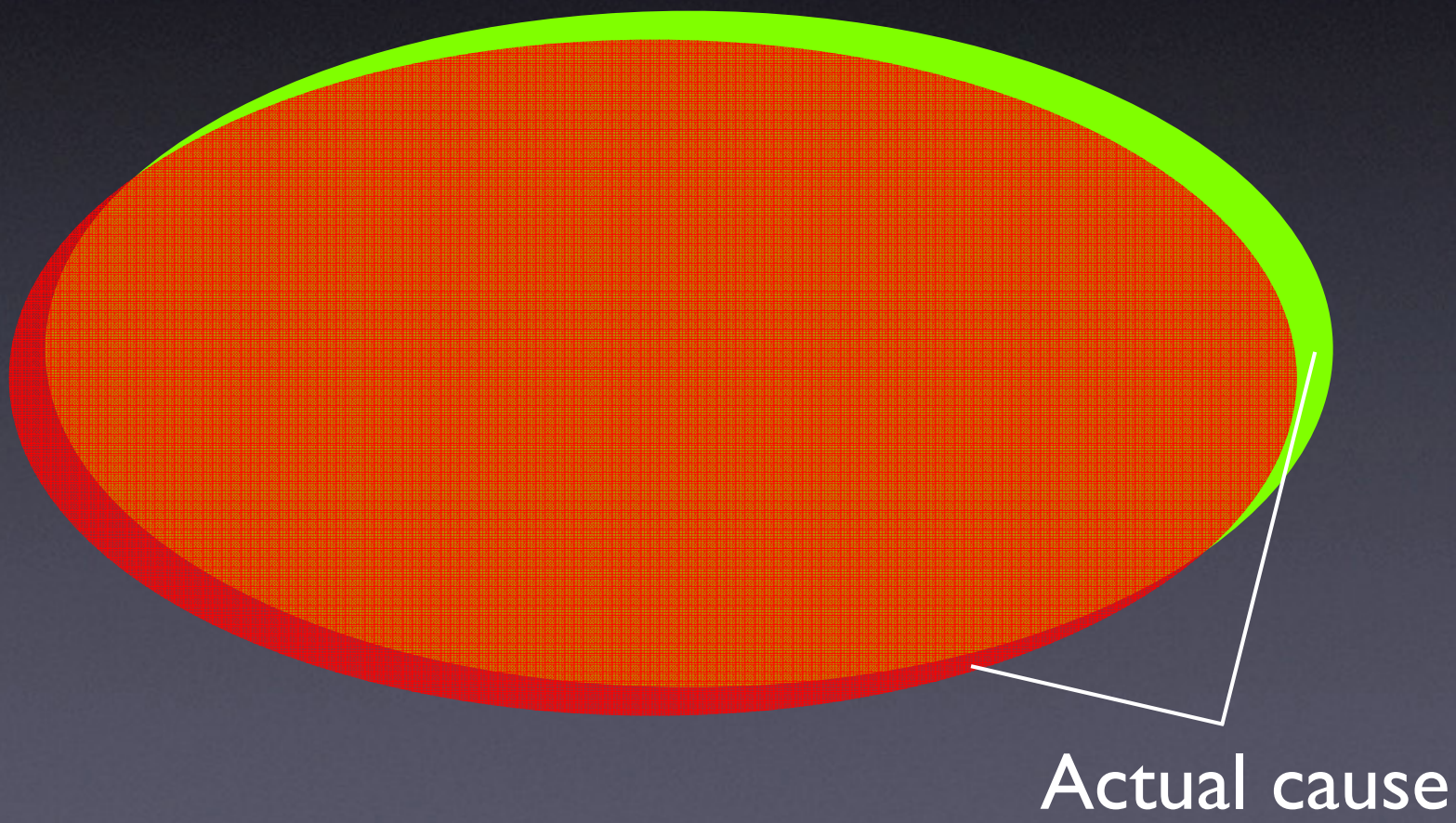
```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[i] + 1.0);
    }
    return z[n];
}
```

Causes as Differences



Actual Causes

"The" cause (*actual cause*) is a *minimal difference*



Isolating Causes

```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (
    }
    return z[n];
}
```

Isolating Causes

```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[i] + 1.0);
    }
    return z[n];
}
```

Isolating Causes

```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0
    }
    return z[n];
}
```

Isolating Causes

```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
```

Actual cause narrowed down

Isolating Causes

```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (i + 1.0);
    }
    return z[n];
}
```

Isolating Causes

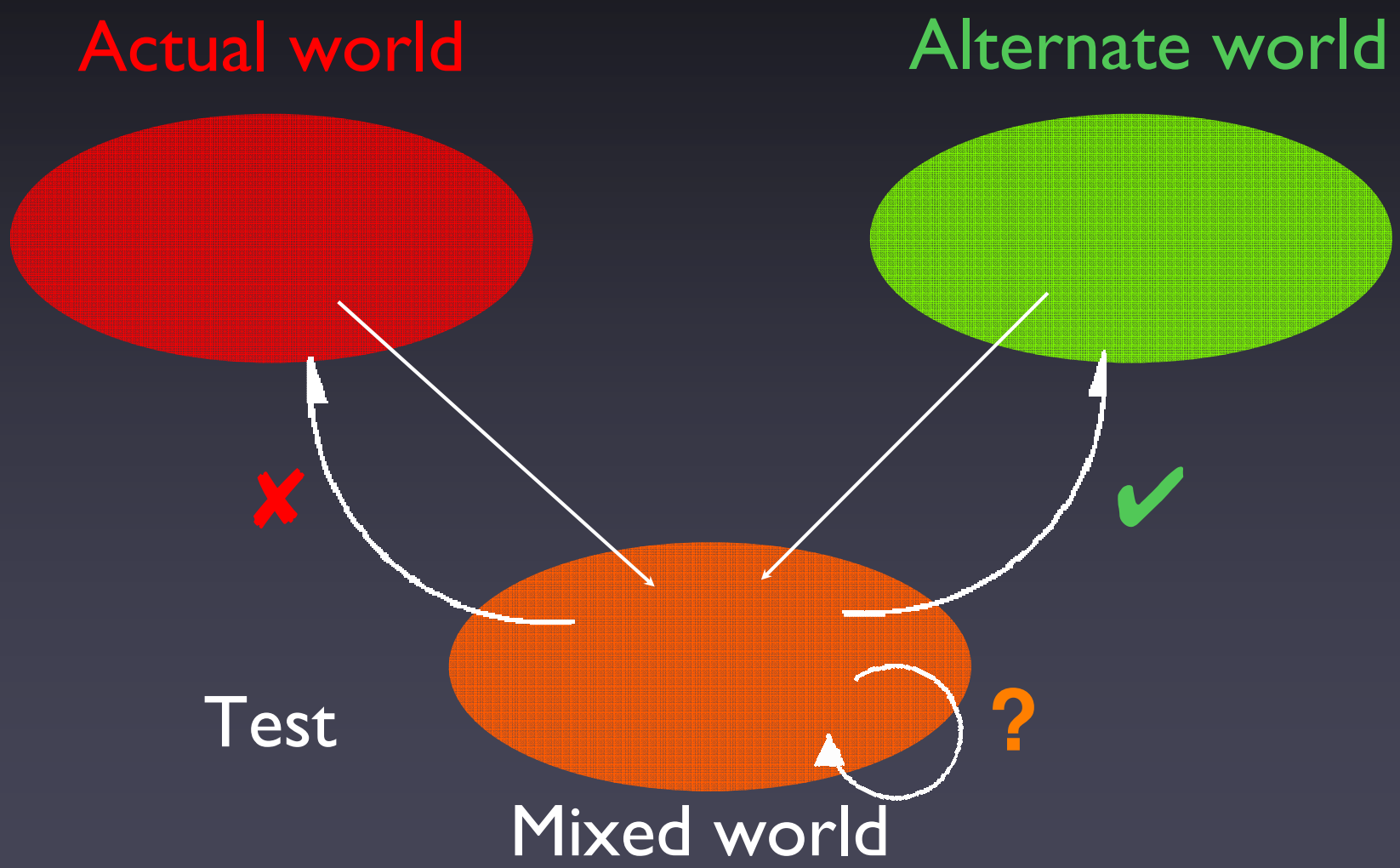
```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0
    }
    return z[n];
}
```

Isolating Causes

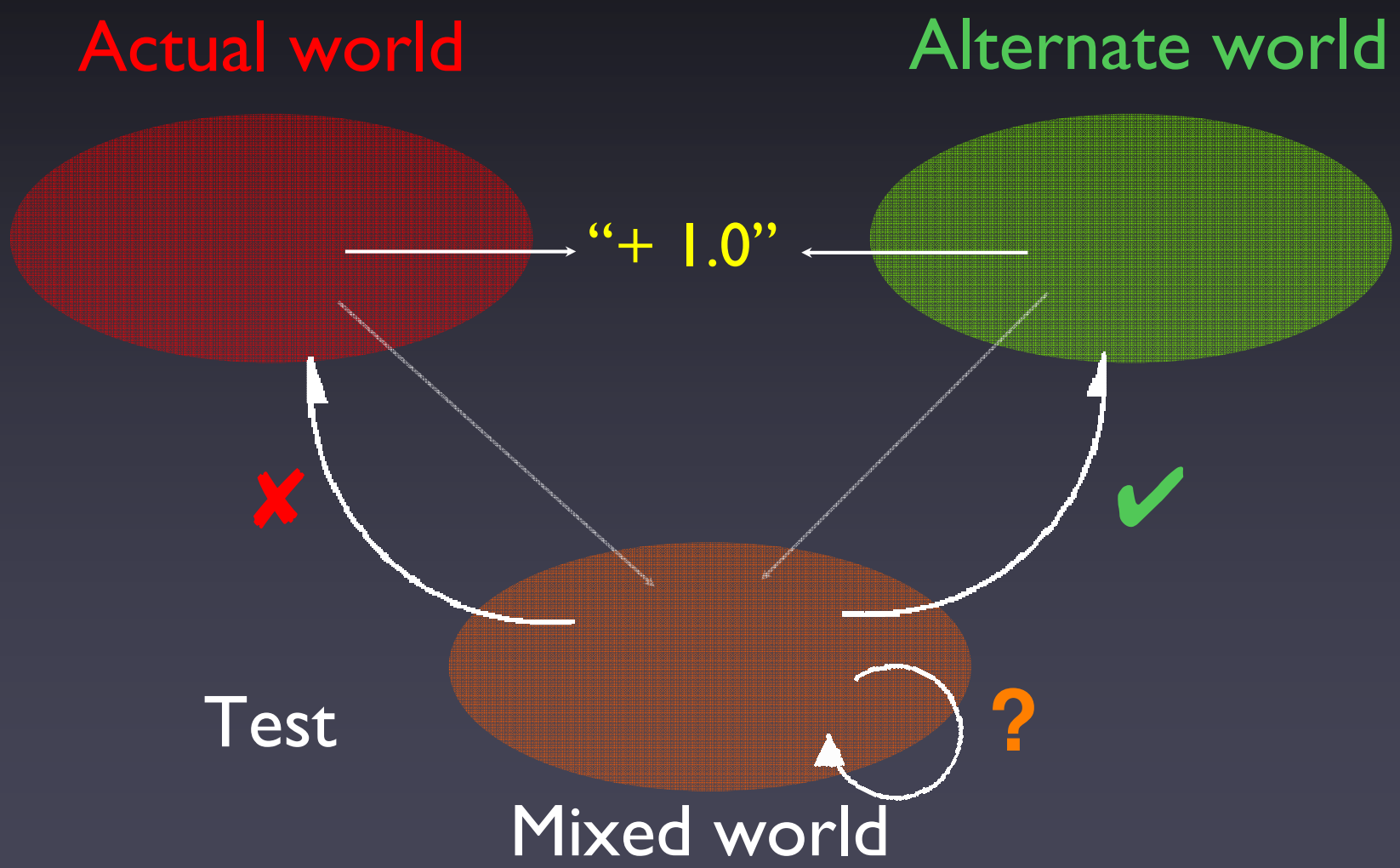
```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
```

Actual cause of the GCC crash

Isolating Causes



Isolating Causes



Delta Debugging

- Delta Debugging *isolates failure causes automatically:*
- Inputs: 1 of 900 HTML lines in Mozilla
- Code changes: 1 of 8,721 code changes in GDB
- Threads: 1 of 3.8 bln thread switches in Scene.java
- Messages: 2 of 347 Java method calls
- Fully automatic + purely test-based

Java - SystemUtilsTest.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Resource Java

Package Explorer JUnit

Finished after 0,047 seconds

Runs: 16/16 Errors: 0 Failures: 1

Failures Hierarchy

testConstructor - org.apache.co

Failure Trace

junit.framework.AssertionFailedError: expecte
at org.apache.commons.lang.SystemUtilsTest.
at sun.reflect.NativeMethodAccessorImpl.invo
at sun.reflect.NativeMethodAccessorImpl.invo
at sun.reflect.DelegatingMethodAccessorImpl.

Outline

SystemUtilsTest.java

```
* Copyright 2003,2004 The Apache Software Foundation.

package org.apache.commons.lang;

import java.io.File;
import java.lang.reflect.Constructor;
import java.lang.reflect.Modifier;
```

Revert all failure-inducing changes

Are you sure you want to revert all failure-inducing changes?

Yes No

Failure-inducing changes

All changes (200)

Failure-inducing changes (2)

src/java/org/apache/commons/lang/SystemUtils.java (1)

Old start: 1069 | New start: 1069 | Added: 4 | Removed: 3

src/test/org/apache/commons/lang/SystemUtilsTest.java (1)

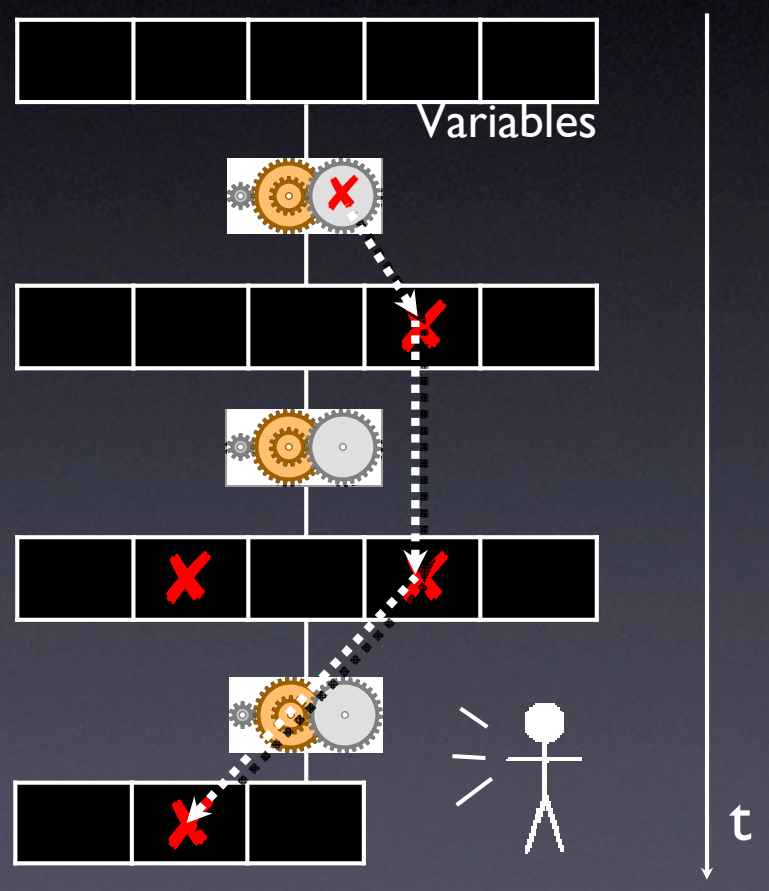
Automated Test

→ Automated Debugging

From Defect to Failure

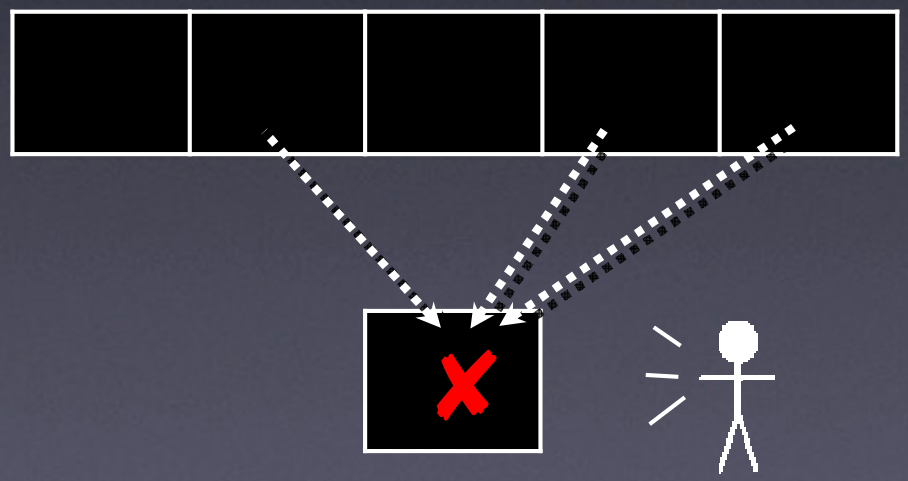
1. The programmer creates a *defect* in the code.
2. When executed, the defect creates an *infection*.
3. The infection *propagates*.
4. The infection causes a *failure*.

This infection chain must be traced back – and broken.

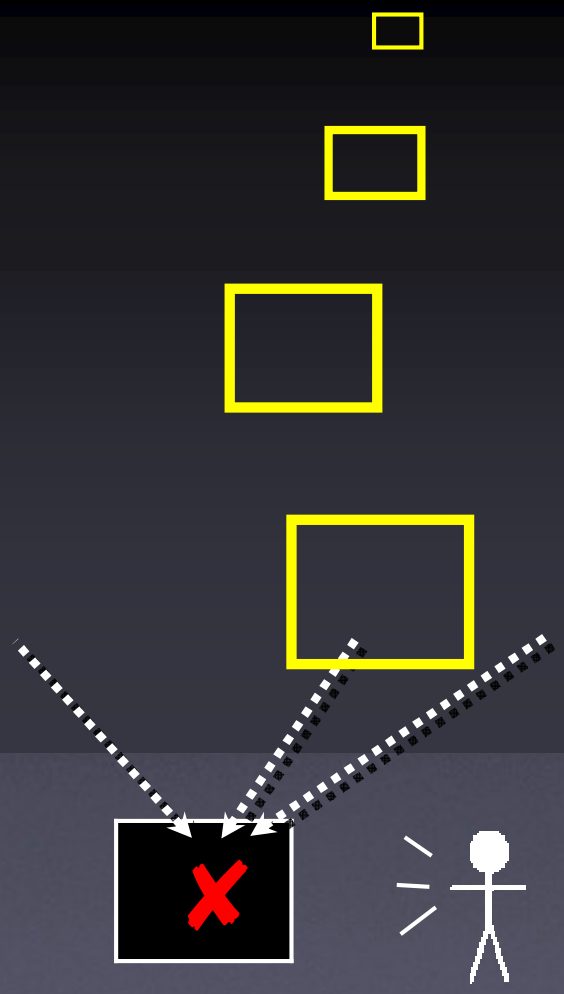


Tracing Infections

- For every infection, we must find the *earlier infection* that *causes* it.
- Program analysis tells us possible causes



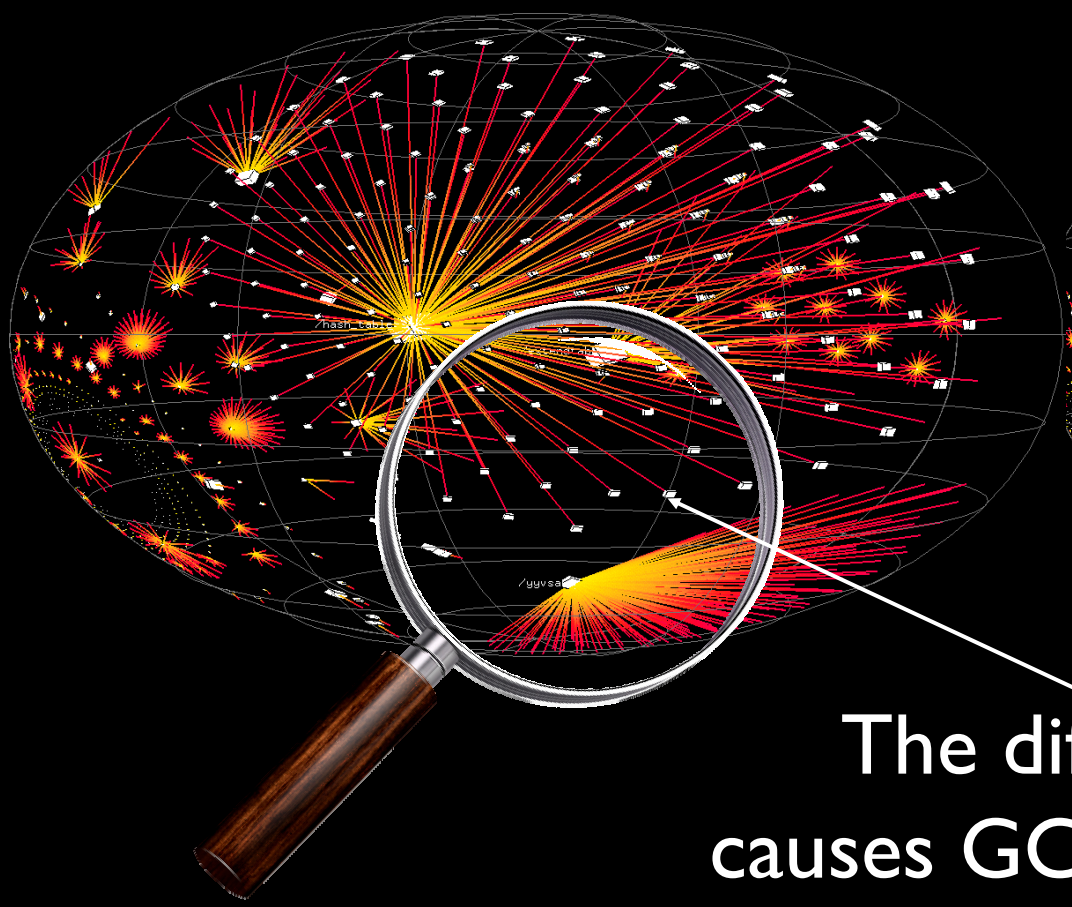
Tracing Infections

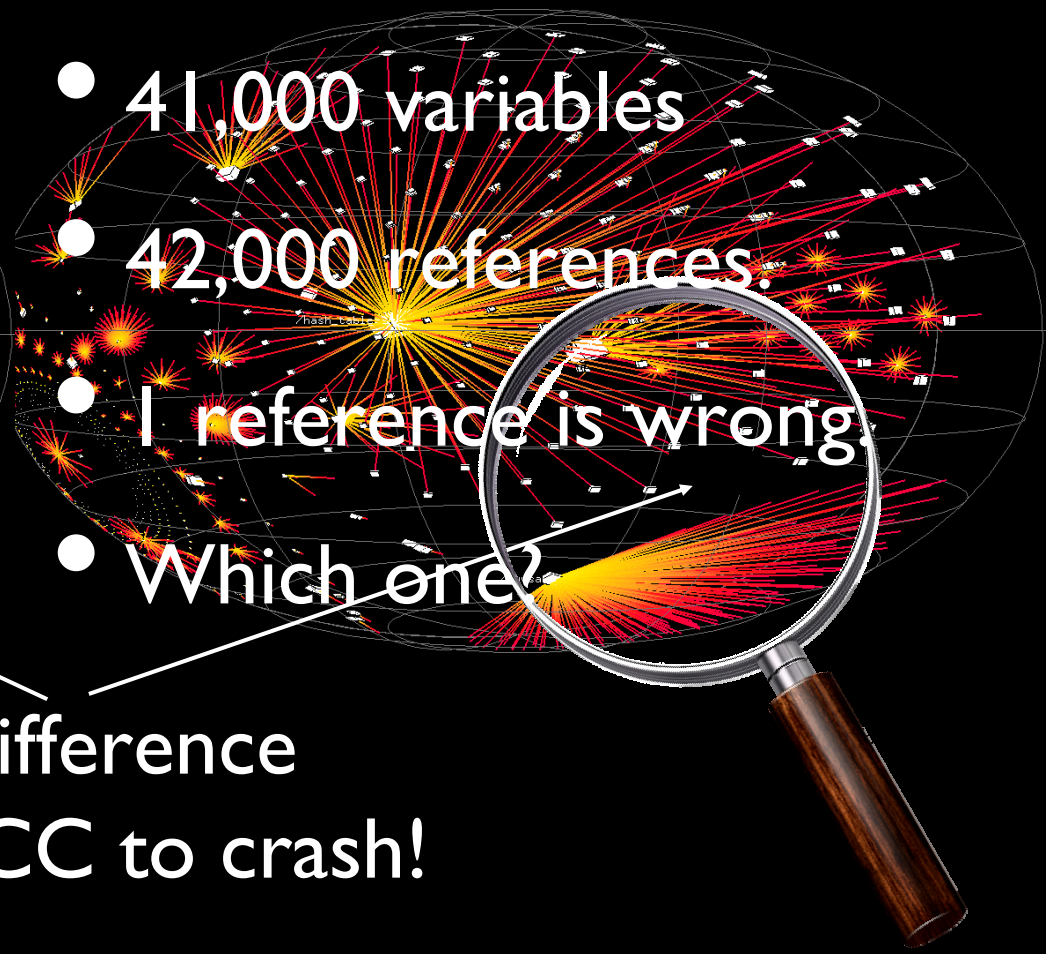


Causes in State

Infected state

Sane state

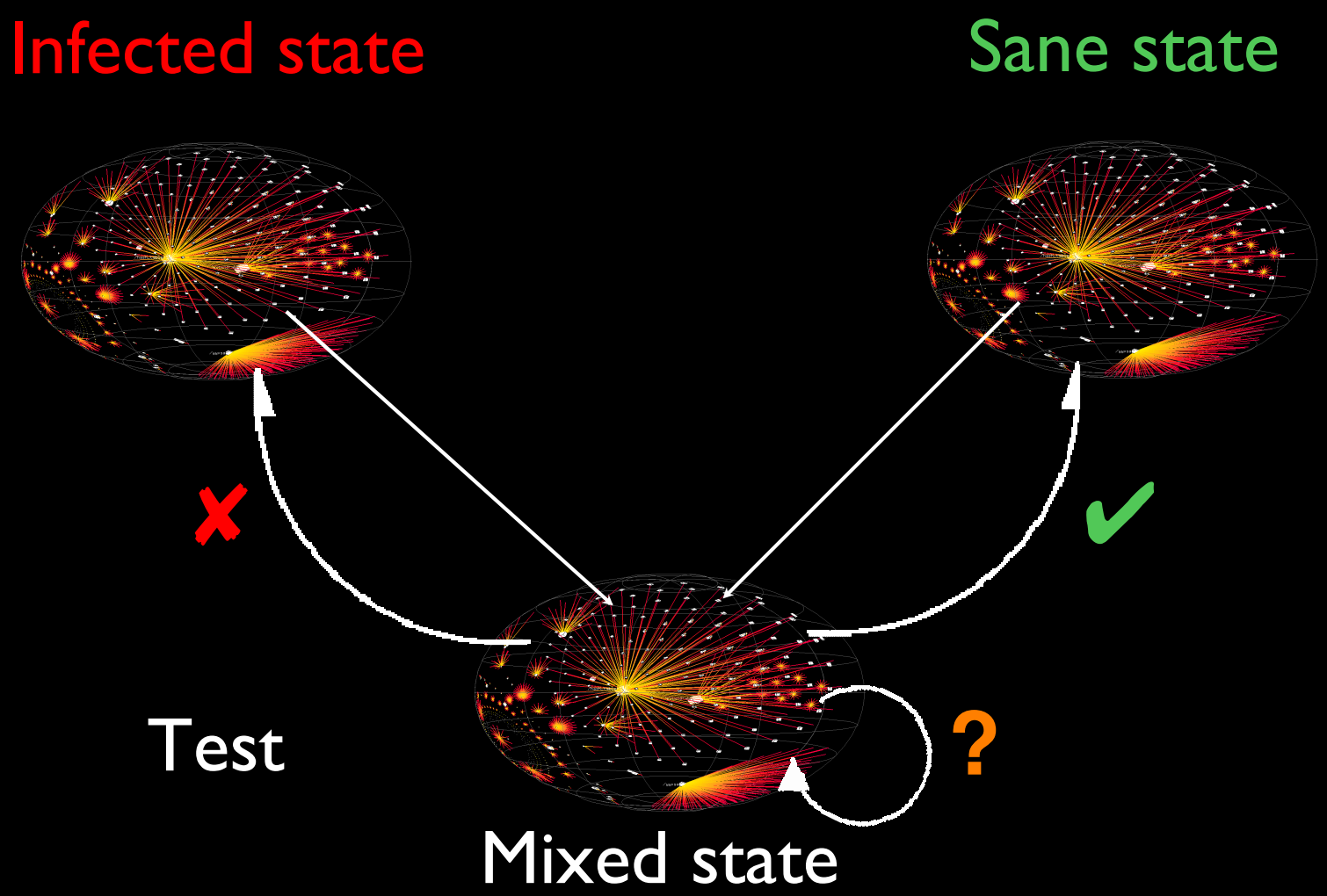




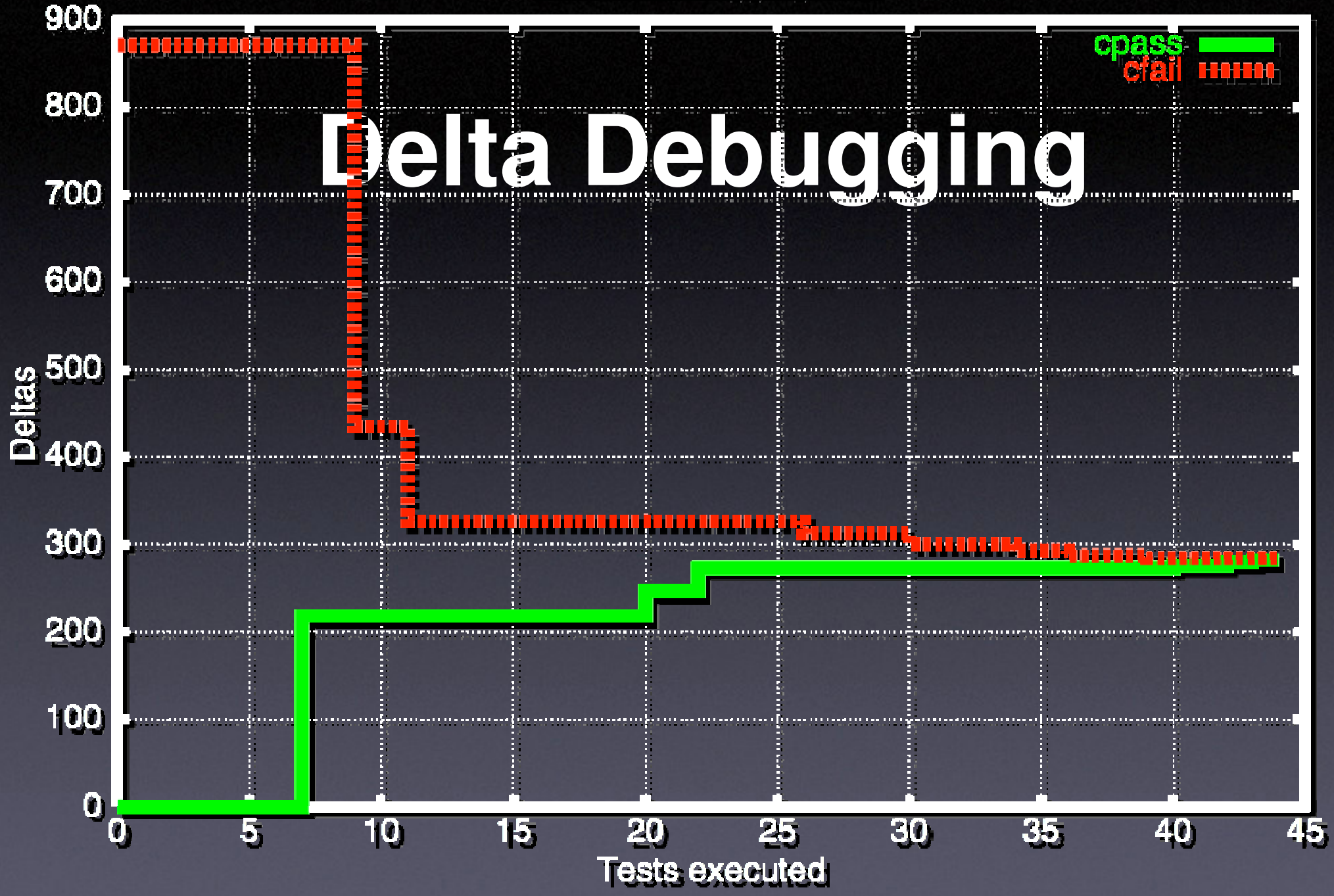
- 41,000 variables
- 42,000 references.
- 1 reference is wrong.
- Which one?

The difference
causes GCC to crash!

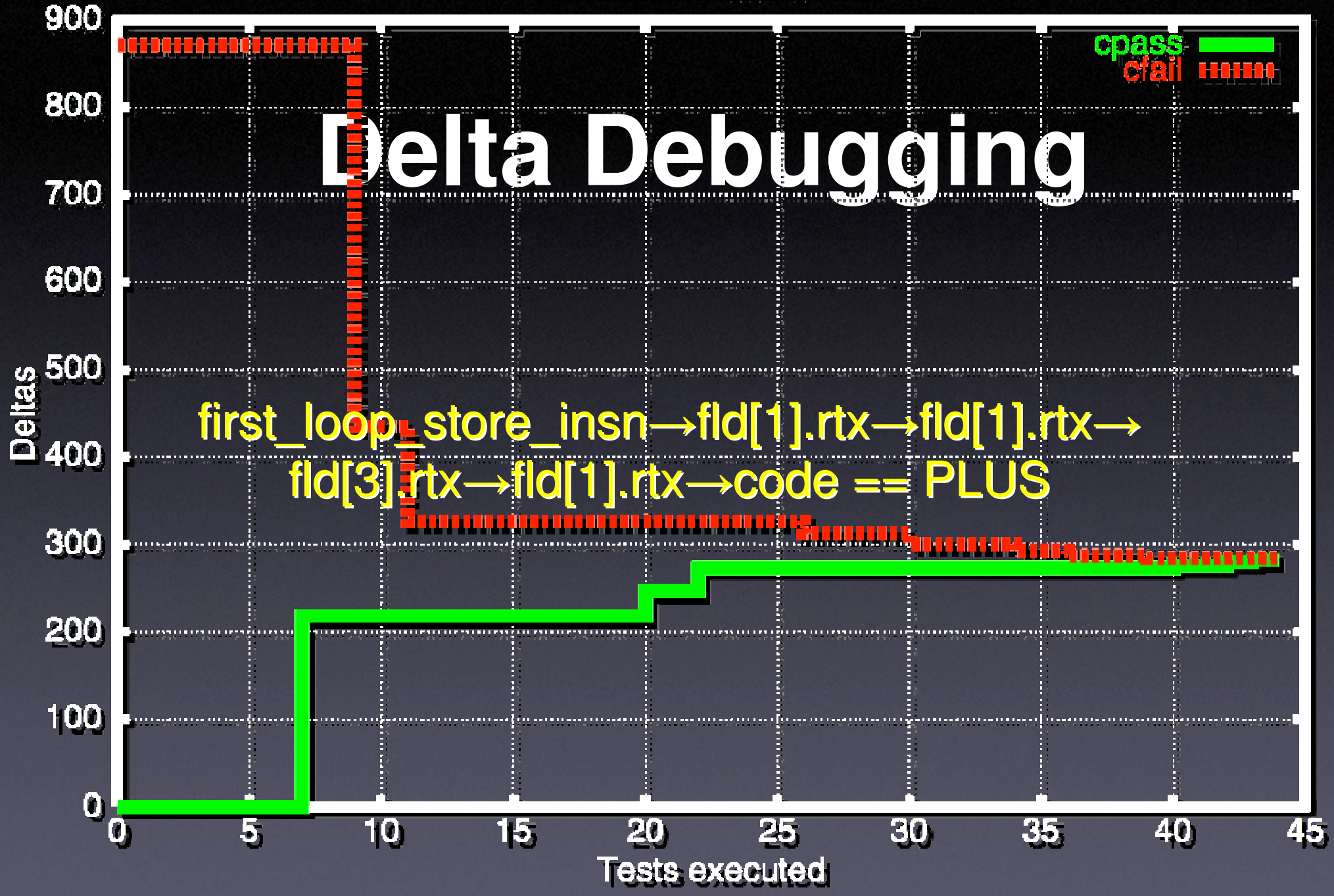
Search in Space



Delta Debugging

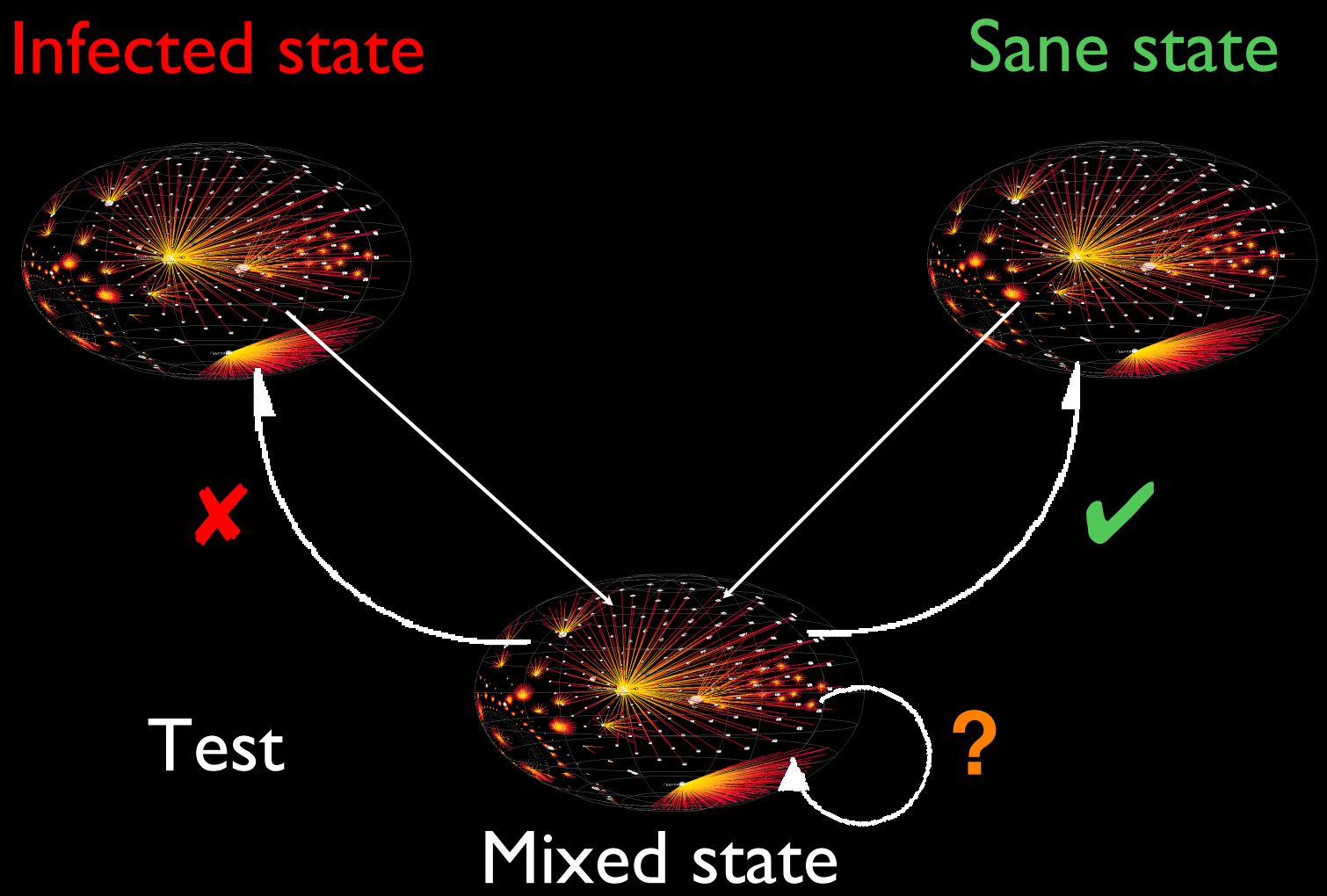


Delta Debugging

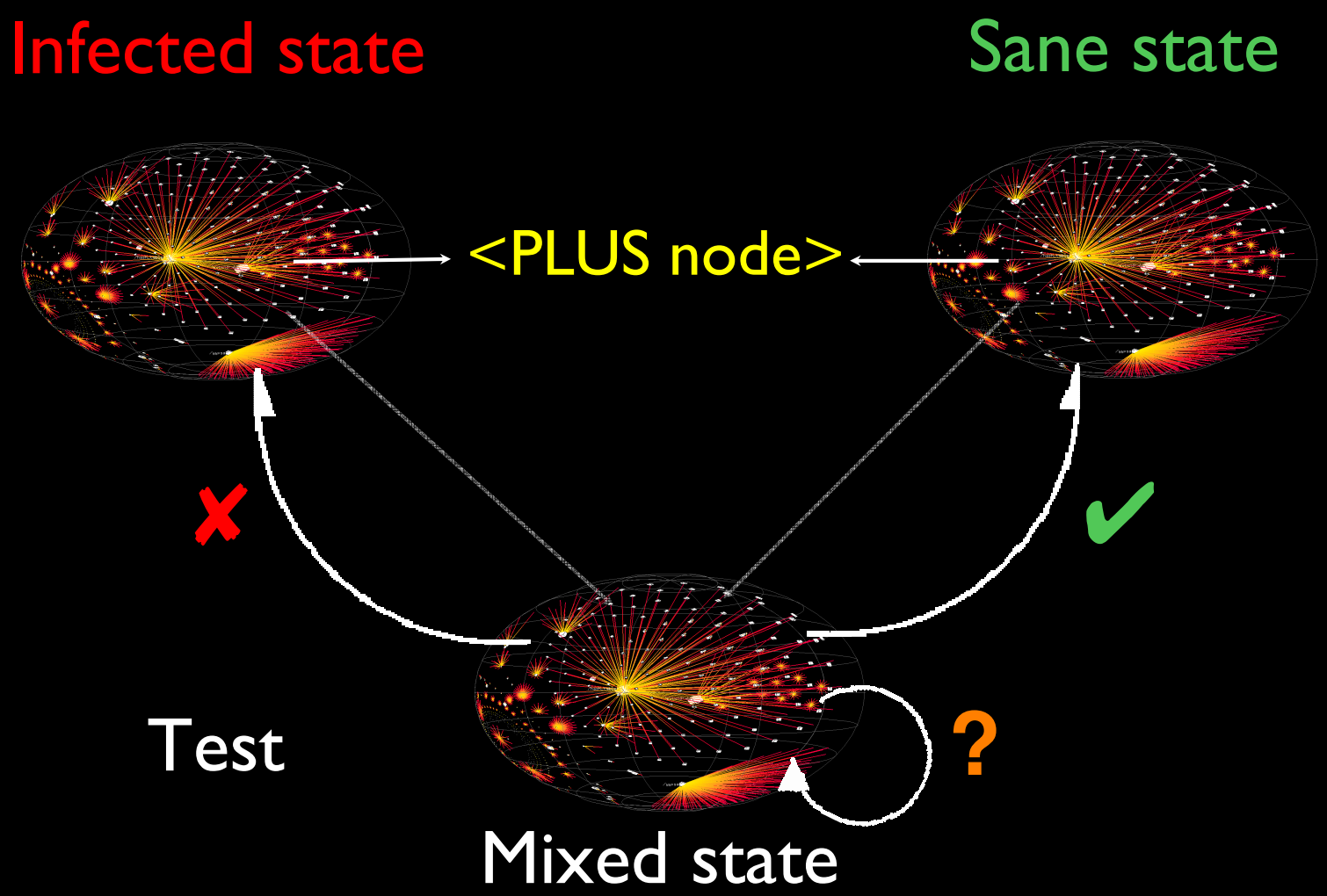


first_loop_store_insn→fld[1].rtx→fld[1].rtx→
fld[3].rtx→fld[1].rtx→code == PLUS

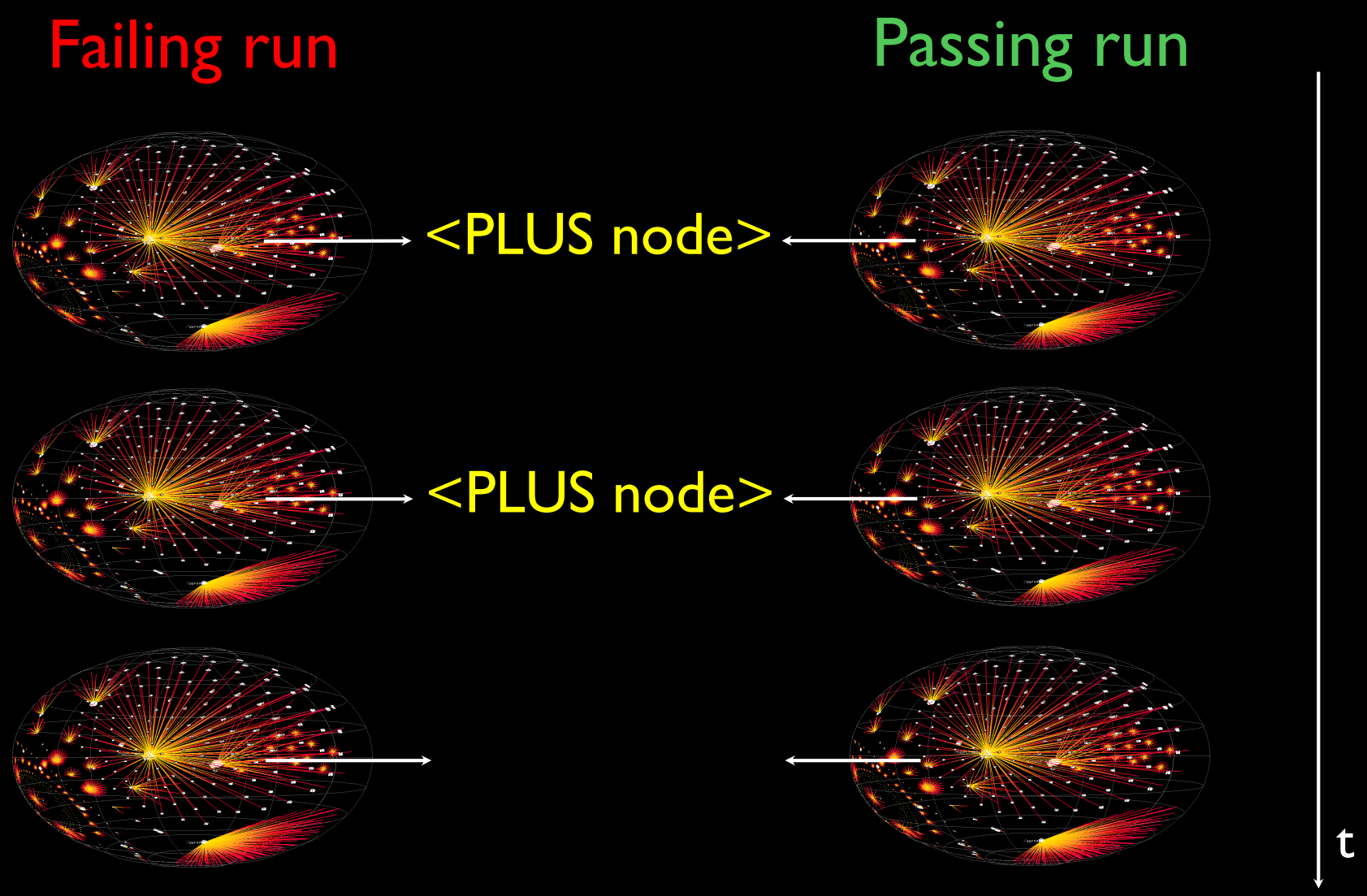
Search in Space



Search in Space



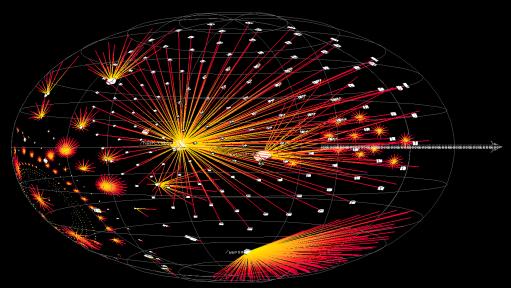
Search in Time

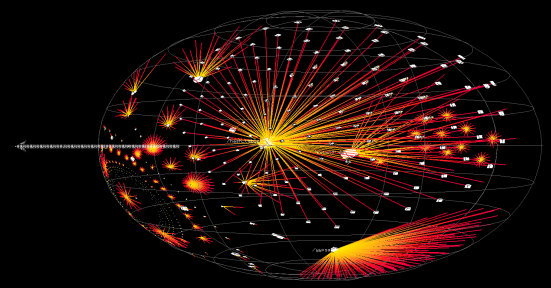


Search in Time

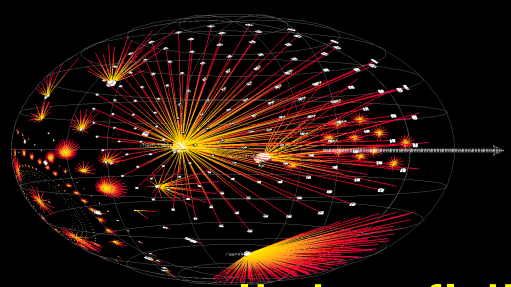
Failing run

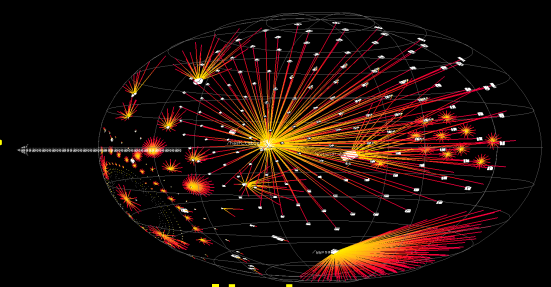
Passing run





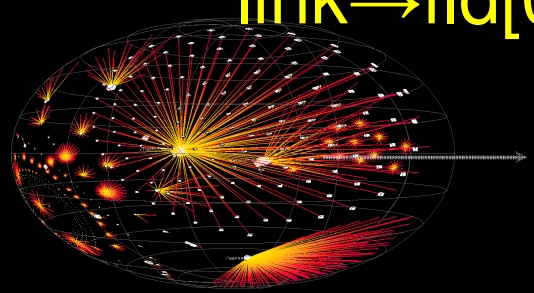
<PLUS node>

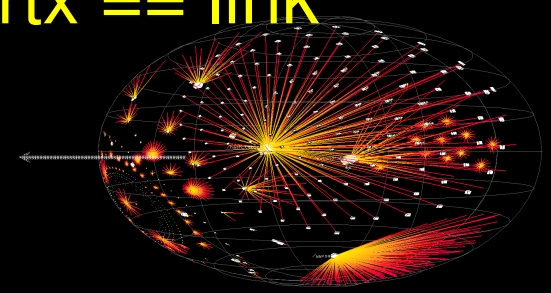




<PLUS node>

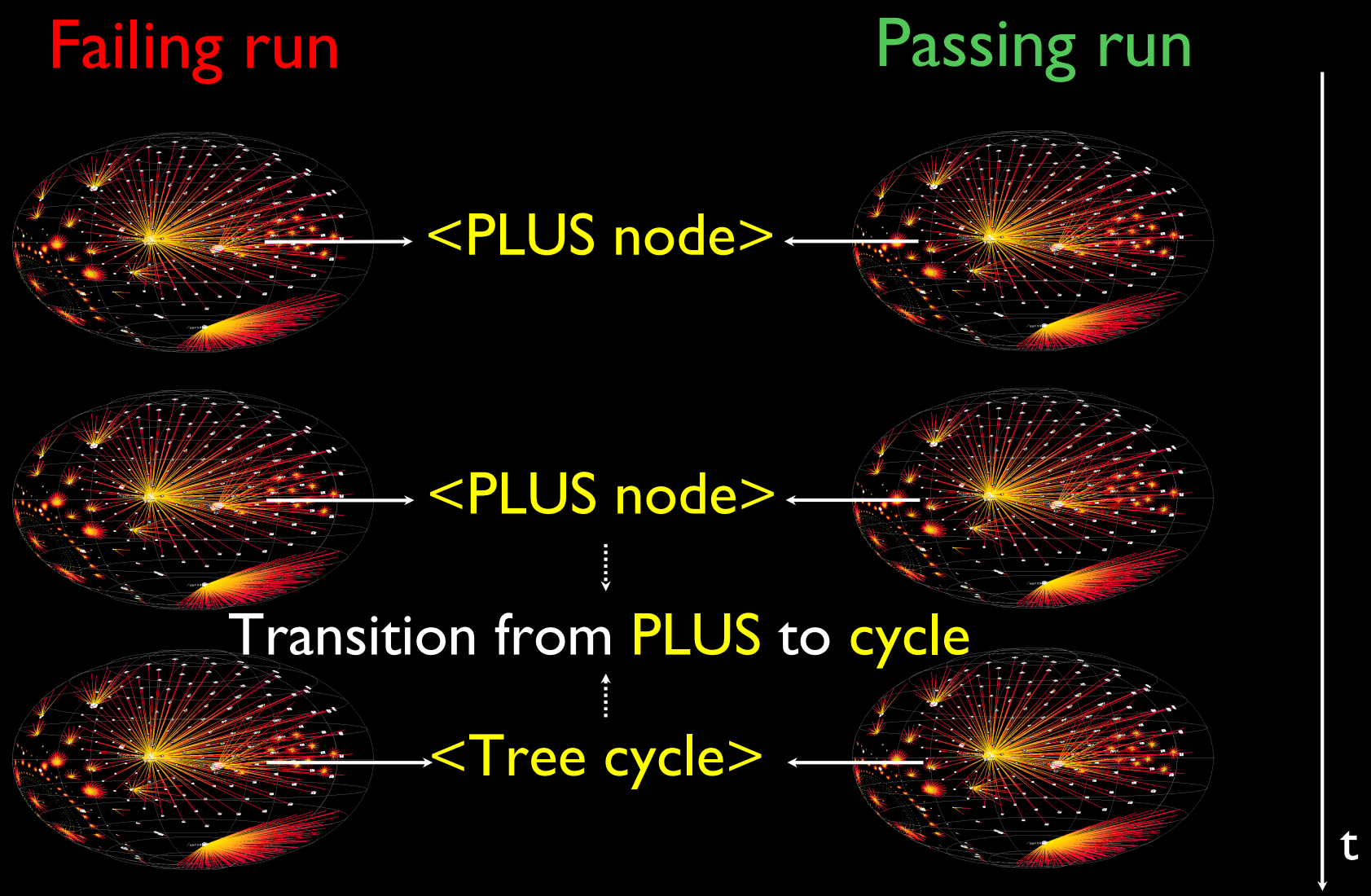
link→fld[0].rtx→fld[0].rtx == link





t

Search in Time



Why Transitions?

- Each failure cause in the program state is caused by some statement
- These statements are executed
- at cause transitions
- Cause transitions thus are statements that cause the failure

All GCC Transitions

Location	New cause at transition
<Start>	argv[3]
toplev.c:4755	name
toplev.c:2909	dump_base_name
c-lex.c:187	finput→_IO_buf_base
c-lex.c:1213	nextchar
c-lex.c:1213	yyssa[41]
c-typeck.c:3615	yyssa[42]
c-lex.c:1213	last_insn→fld[1].rtx→…→fld[1].rtx.code
c-decl.c:1213	sequence_result[2]→…→fld[1].rtx.code
combine.c:4271	x→fld[0].rtx→fld[0].rtx

combine.c

```
if (GET_CODE (XEXP (x, 0)) == PLUS {
    x = apply_distributive_law
          (gen_binary (PLUS, mode,
                gen_
                g
                    XEXP (x, 1))));

    if (GET_CODE (x) != MULT)
          return x;
}
```

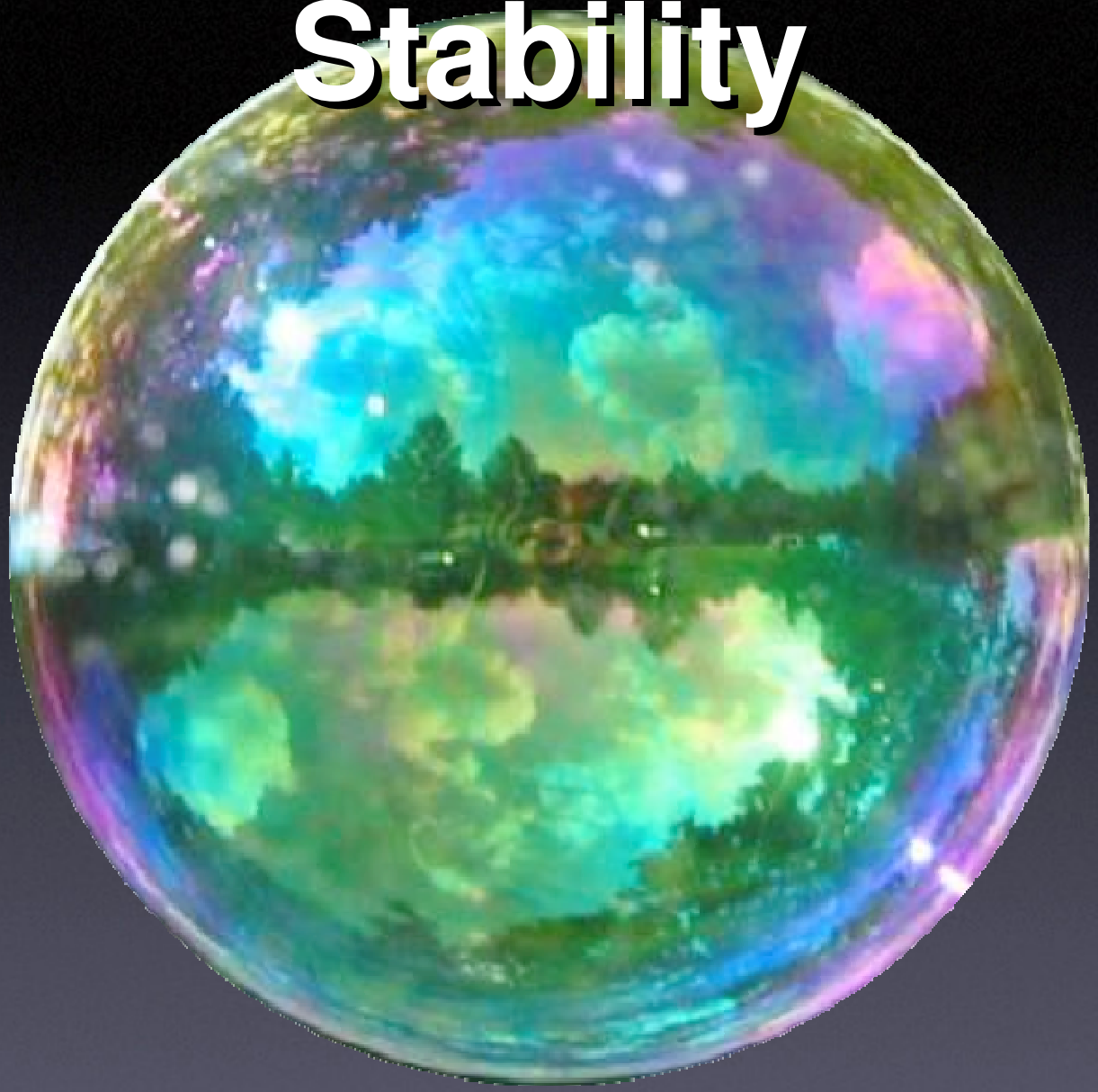
2 lines out of 338,000

Should be copy_rtx()

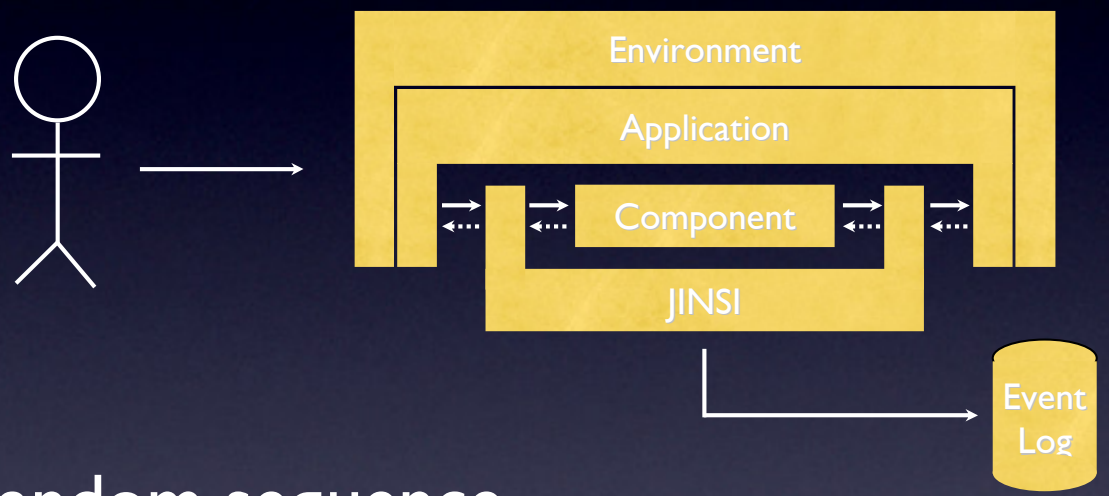
Implementations

C	Java	Python
Web service + command line	Eclipse plug-in	Module
24 months	12 months	2 days

Stability



Isolating Relevant Calls



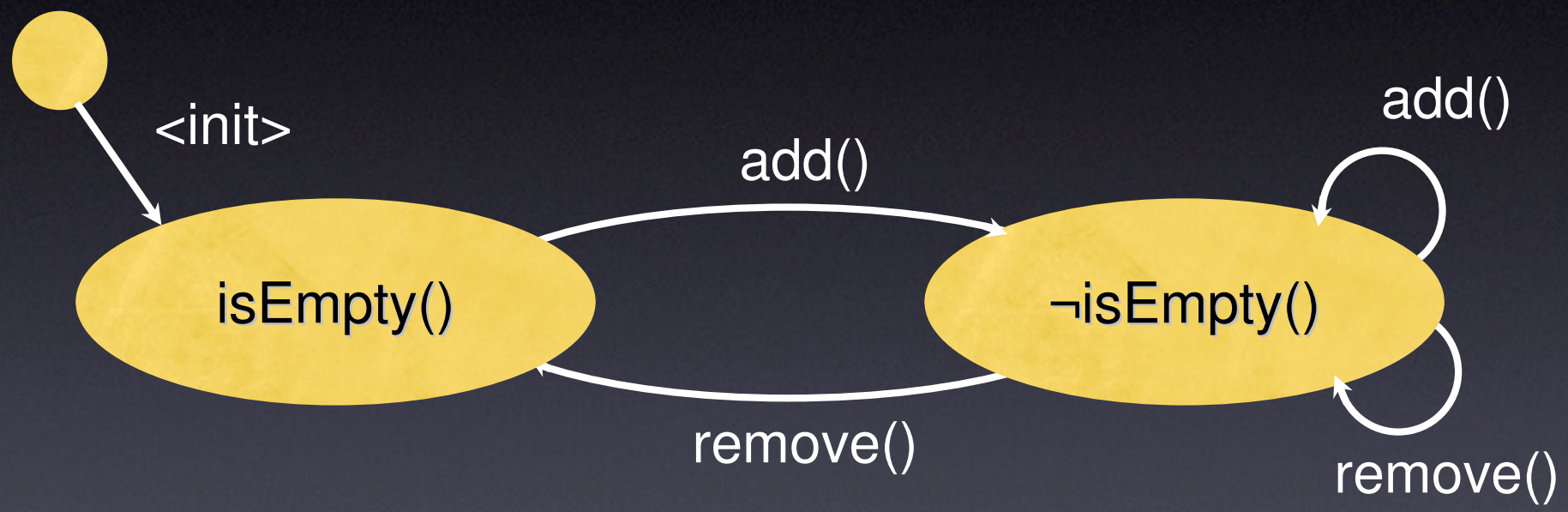
Random sequence of 28 operations

Event log contains 32 interactions

```
<init>
insert(Coin(25))
insert(Coin(10))
insert(Coin(25))
insert(Coin(25))
vend()
vend()
```

Minimal set to reproduce the failure: 7 interactions

Mining Call Sequences



How are these related to failures?

The Traffic Principle

T rack the problem

R eproduce

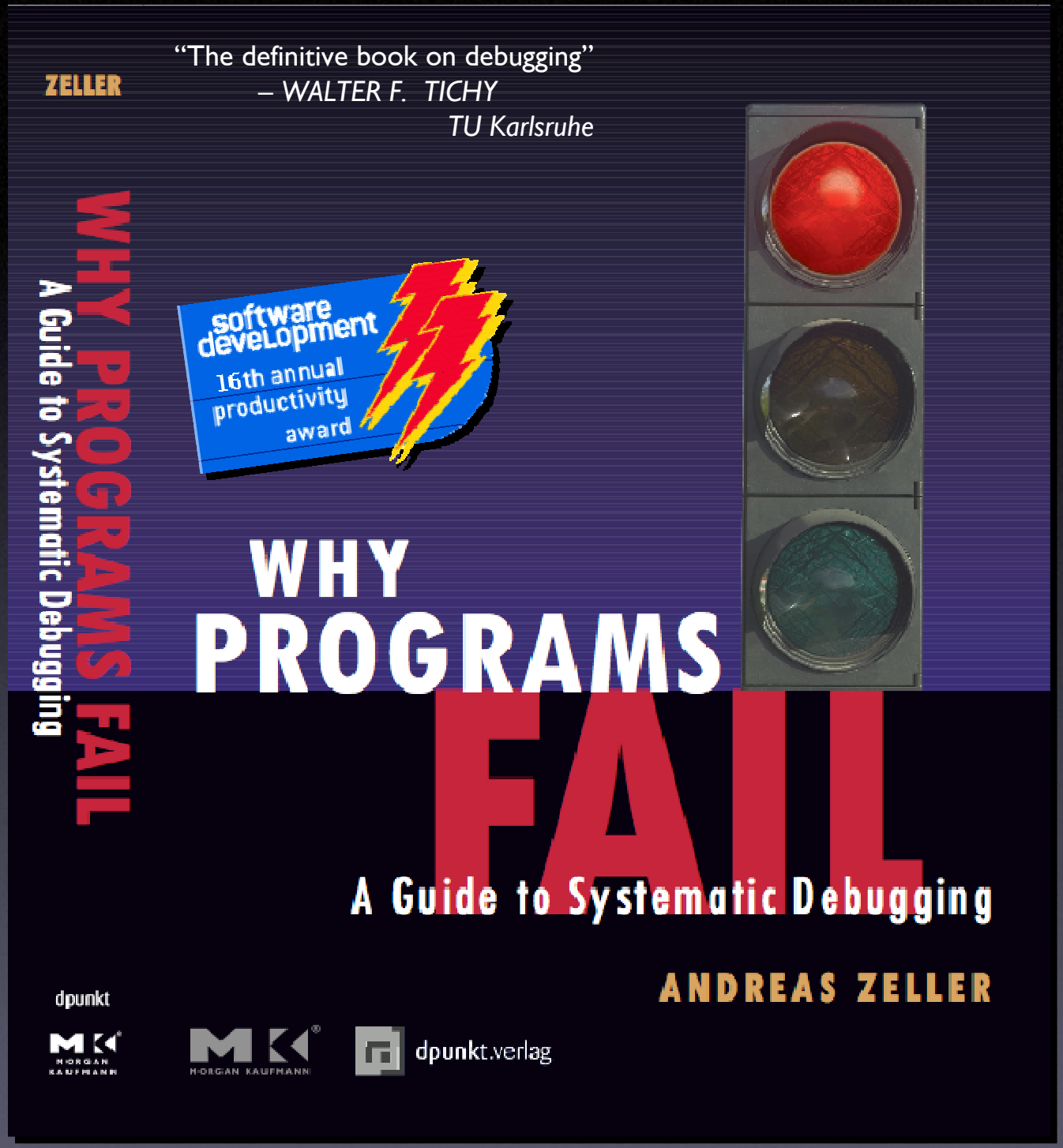
A utomate

F ind Origins

F ocus

I solate

C orrect

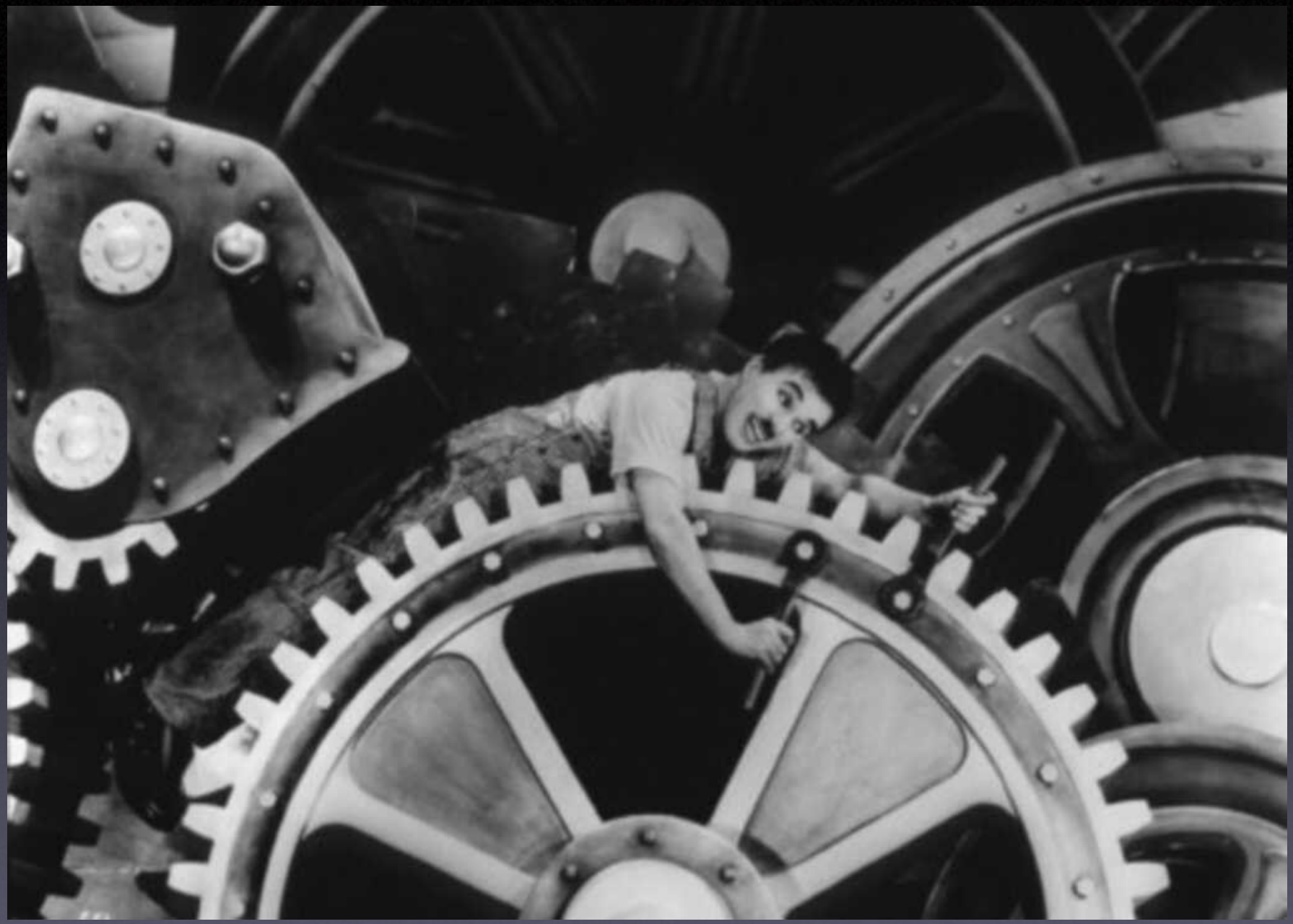

"The definitive book on debugging"
– WALTER F. TICHY
TU Karlsruhe
ZELLER
WHY PROGRAMS FAIL
A Guide to Systematic Debugging
software development
16th annual
productivity
award
WHY
PROGRAMS
FAIL
A Guide to Systematic Debugging
ANDREAS ZELLER
dpunkt
MORGAN KAUFMANN
dpunkt.verlag

combine.c

```
if (GET_CODE (XEXP (x, 0)) == PLUS {
    x = apply_distributive_law
          (gen_binary (PLUS, mode,
                gen_
                g
                    XEXP (x, 1))));

    if (GET_CODE (x) != MULT)
          return x;
}
```

What is the cause of this error?

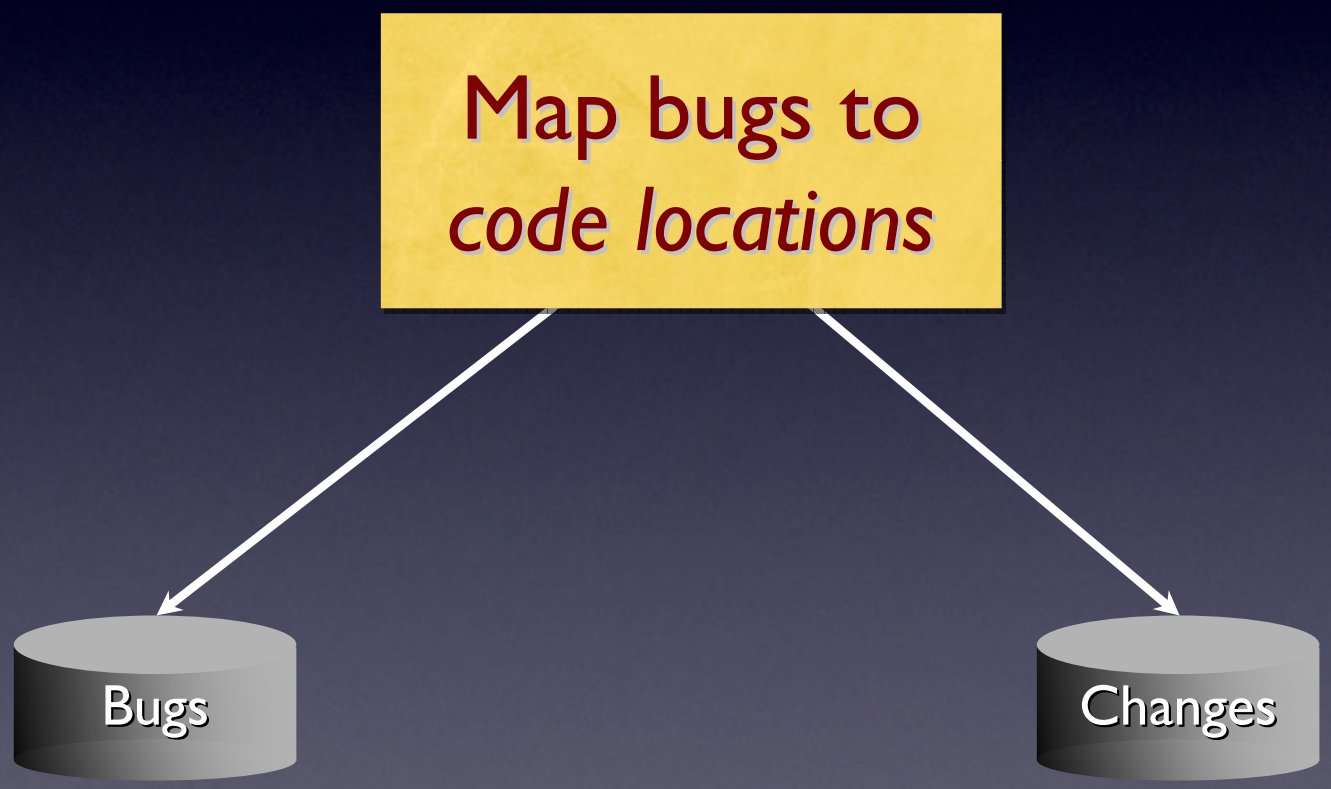
Should be copy_rtx()

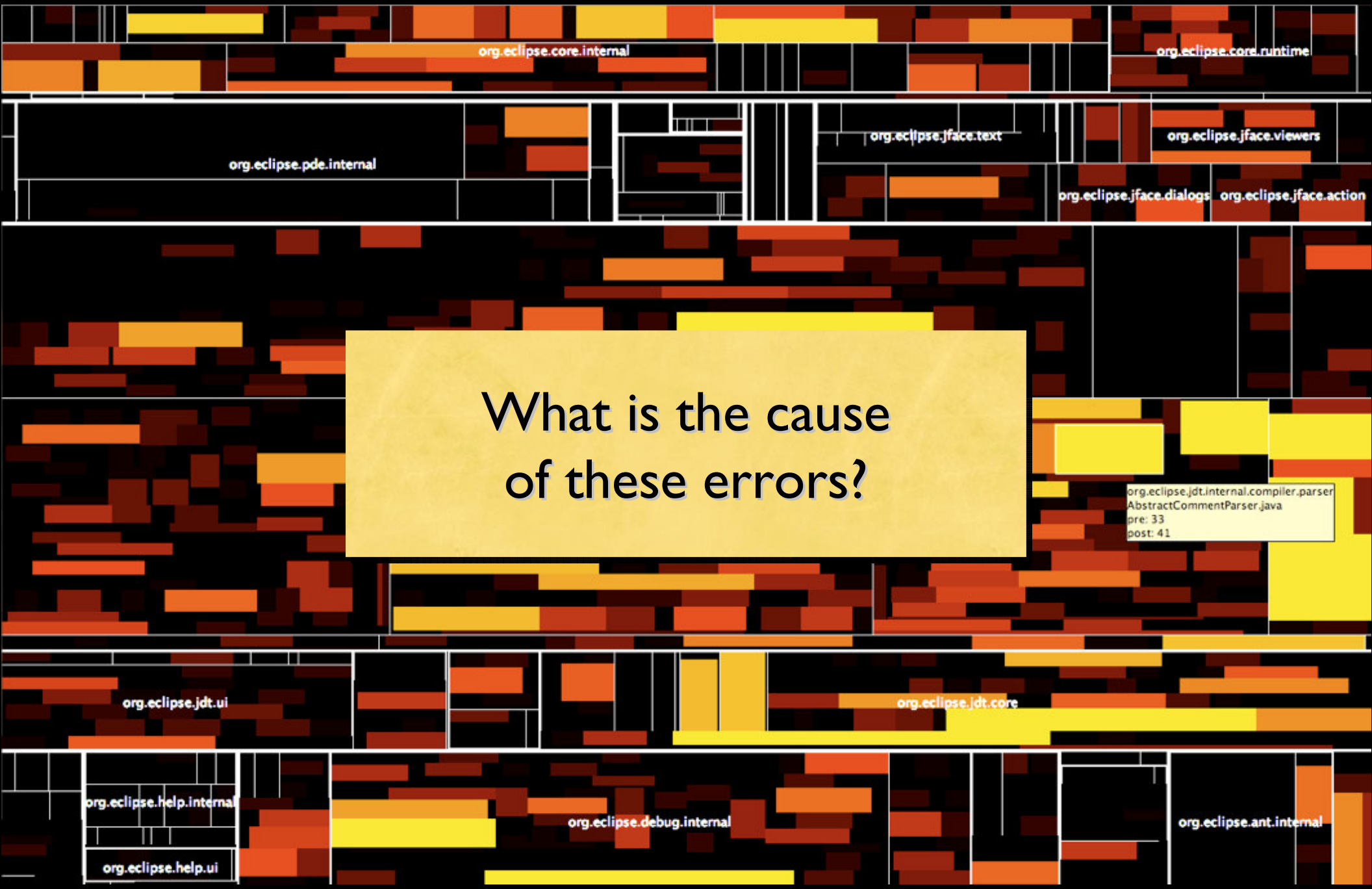


Problem Report for Keynote
Problem and system information:
Date/Time: 2006-03-07 23:35:25.516 +0100
OS Version: 10.4.5 (Build 8H14)
Report Version: 4
Command: Keynote
Path: /Applications/iWork '06/Keynote.app/Contents/MacOS/Keynote
Parent: WindowServer [79]
Version: 3.0.0 (423)
Build Version: 1
Project 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


Map bugs to
code locations
Bugs
Changes



What is the cause of these errors?

Is it the Developers?

Does experience matter?

Bug density correlates with experience!

Is it History?

I found lots of bugs here. Will there be more?

Yes! (But where did these come from?)

How about metrics?

Do *code metrics* correlate with bug density?

Sometimes!

Uh. Coverage?

Does *test coverage* correlate with bug density?

Yes – the more coverage, the more bugs!

Ah! Language features?

Are gotos harmful?

No correlation!

Ok. Problem Domain?

Which tokens do matter?

import • extends • implements

Eclipse Imports

71% of all components importing compiler show a post-release defect

```
import org.eclipse.jdt.internal.compiler.lookup.*;
import org.eclipse.jdt.internal.compiler.*;
import org.eclipse.jdt.internal.compiler.ast.*;
import org.eclipse.jdt.internal.compiler.util.*;
...
import org.eclipse.pde.core.*;
import org.eclipse.jface.wizard.*;
import org.eclipse.ui.*;
```

14% of all components importing ui show a post-release defect

Joint work with Adrian Schröter • Tom Zimmermann

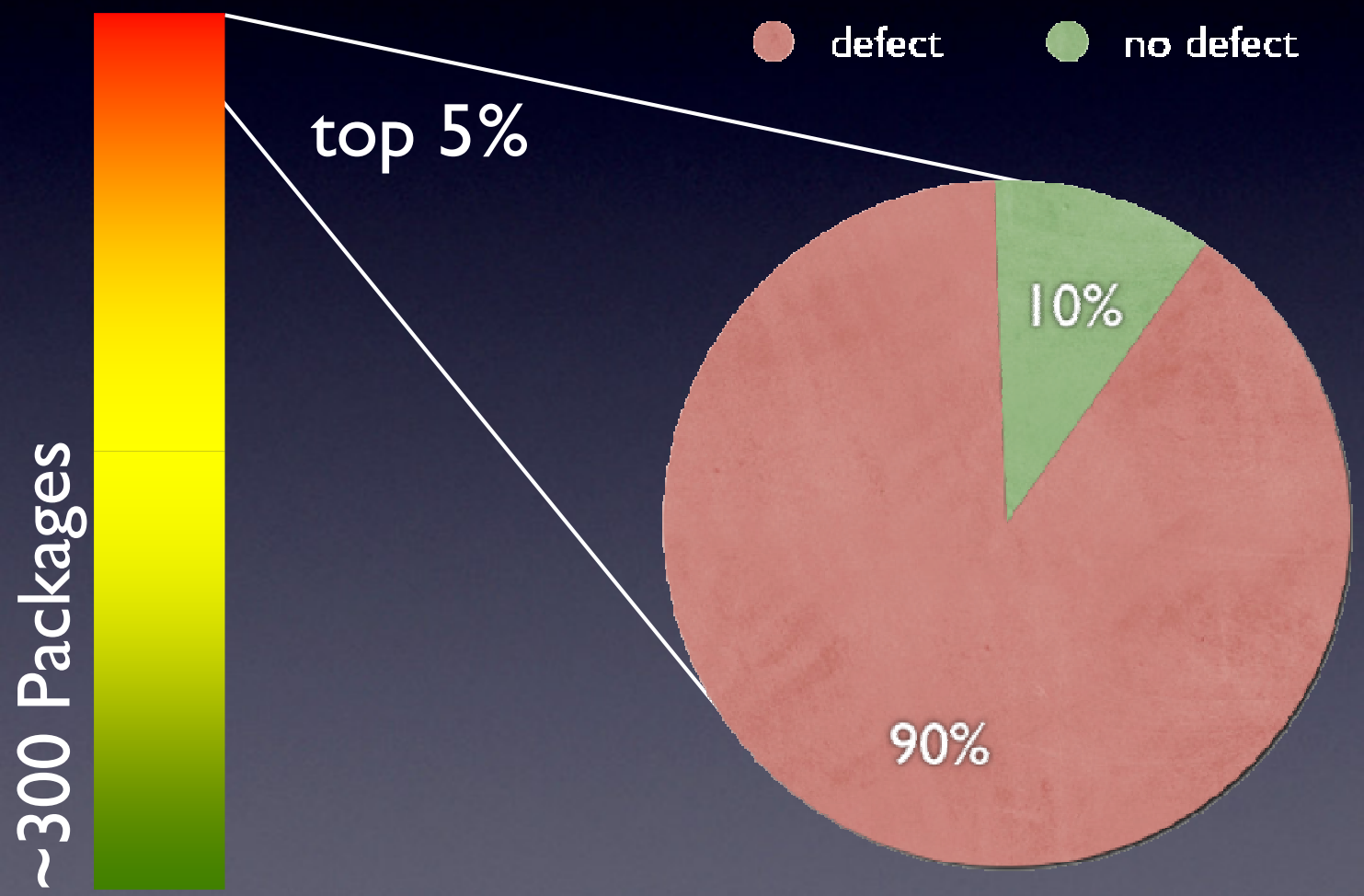
Eclipse Imports

Correlation with failure

```
import org.eclipse.jdt.internal.compiler.lookup.*;
import org.eclipse.jdt.internal.compiler.*;
import org.eclipse.jdt.internal.compiler.ast.*;
import org.eclipse.jdt.internal.compiler.util.*;
...
import org.eclipse.pde.core.*;
import org.eclipse.jface.wizard.*;
import org.eclipse.ui.*;
```

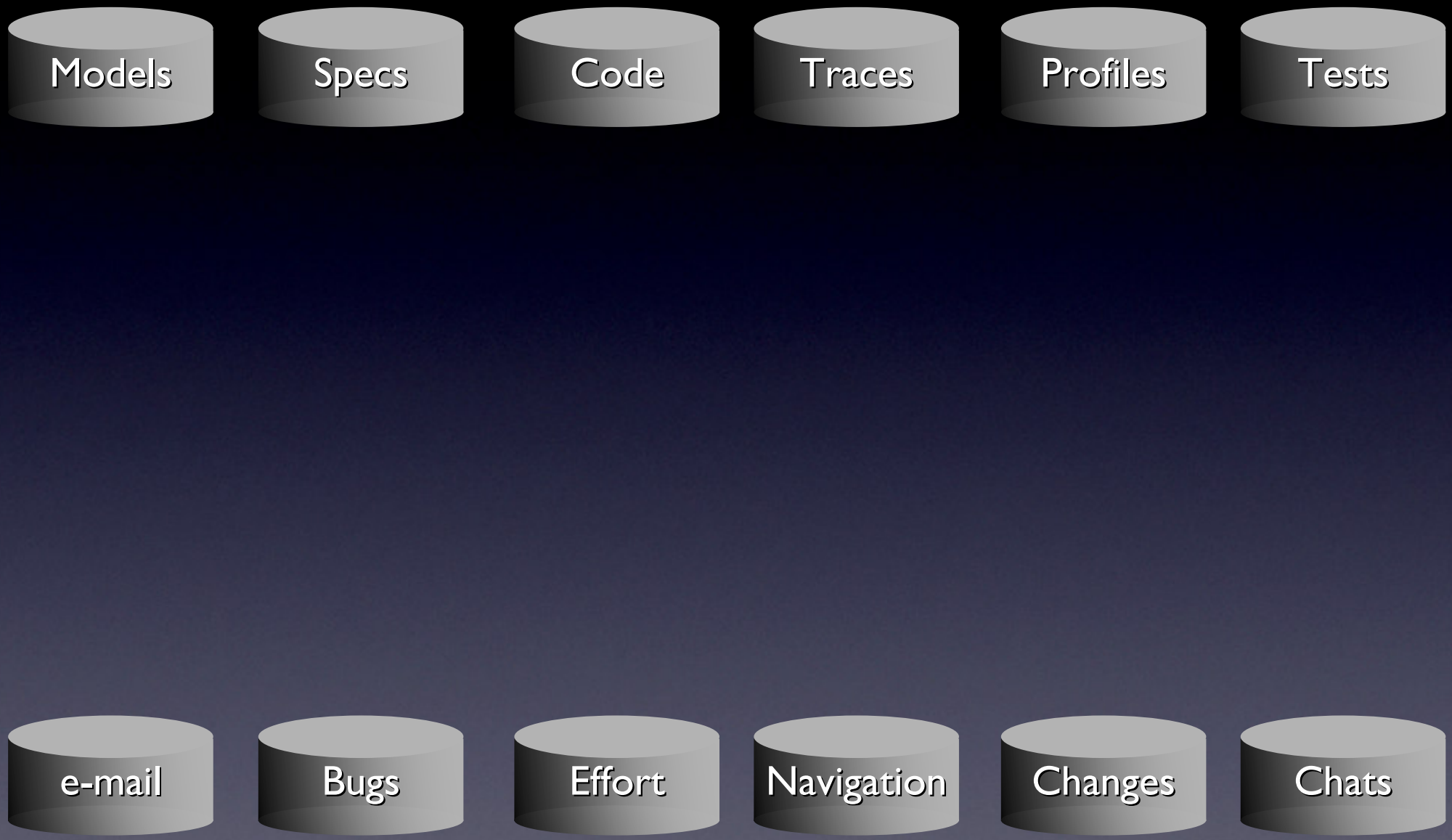
Correlation with success

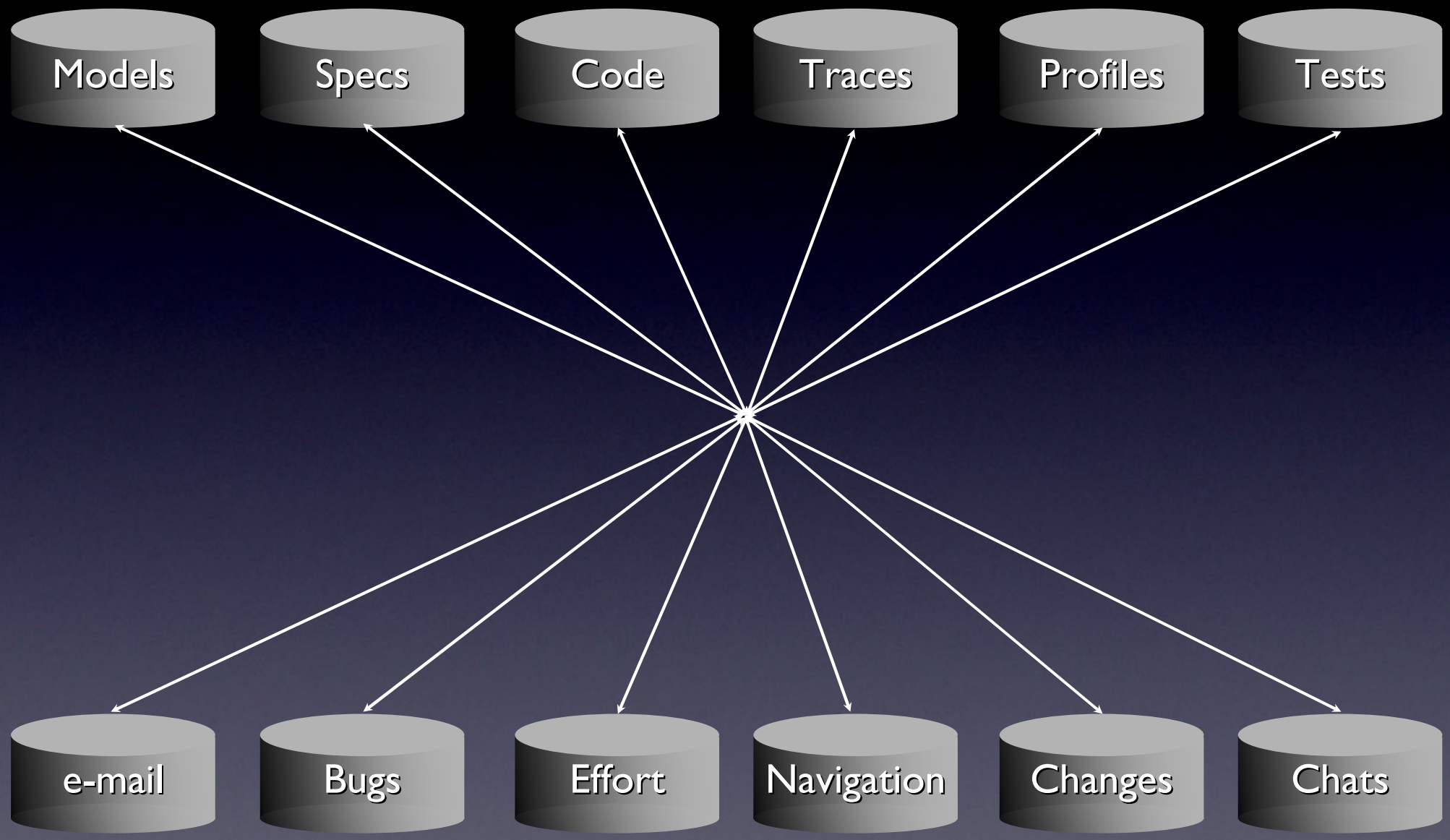
Prediction

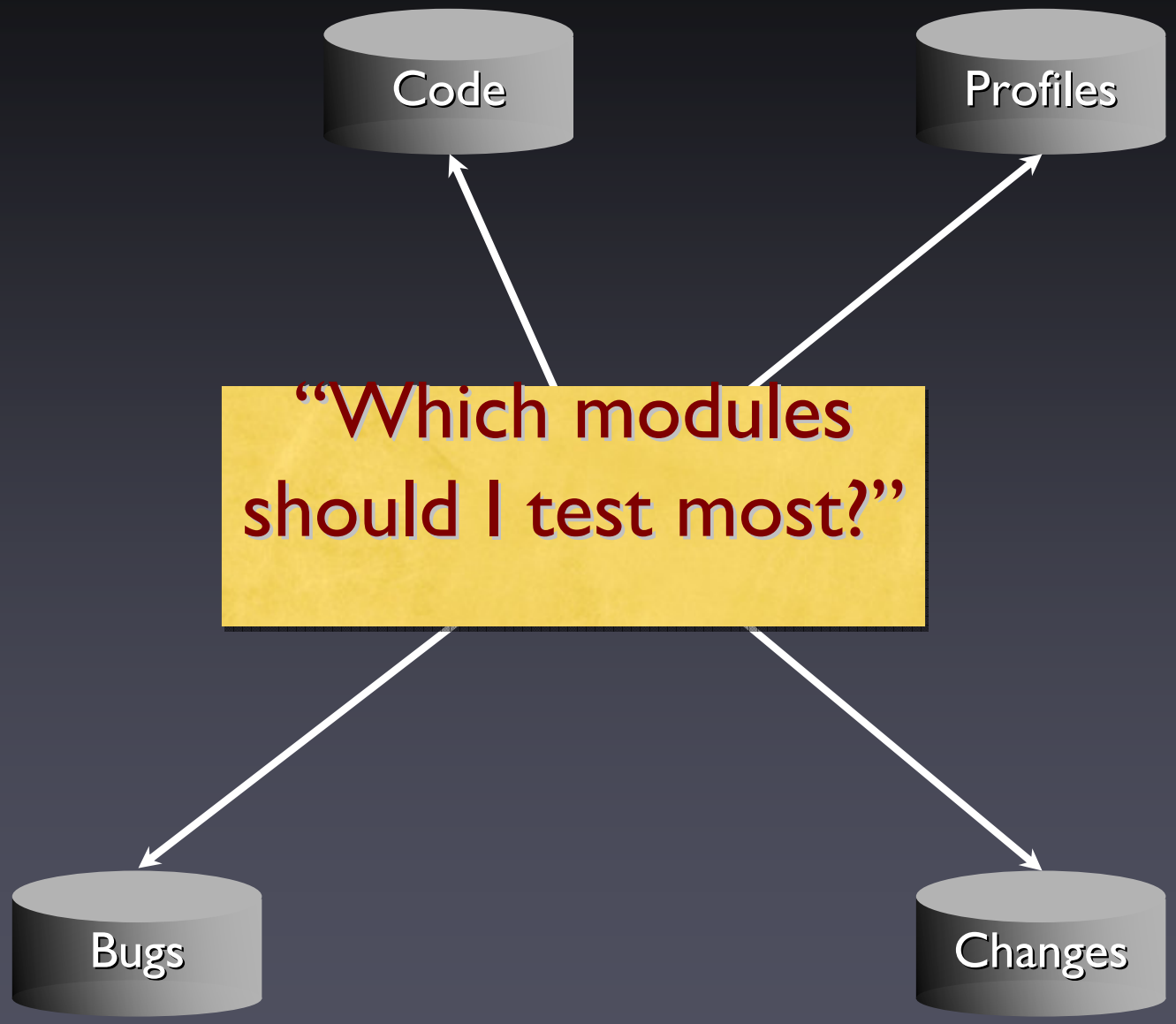


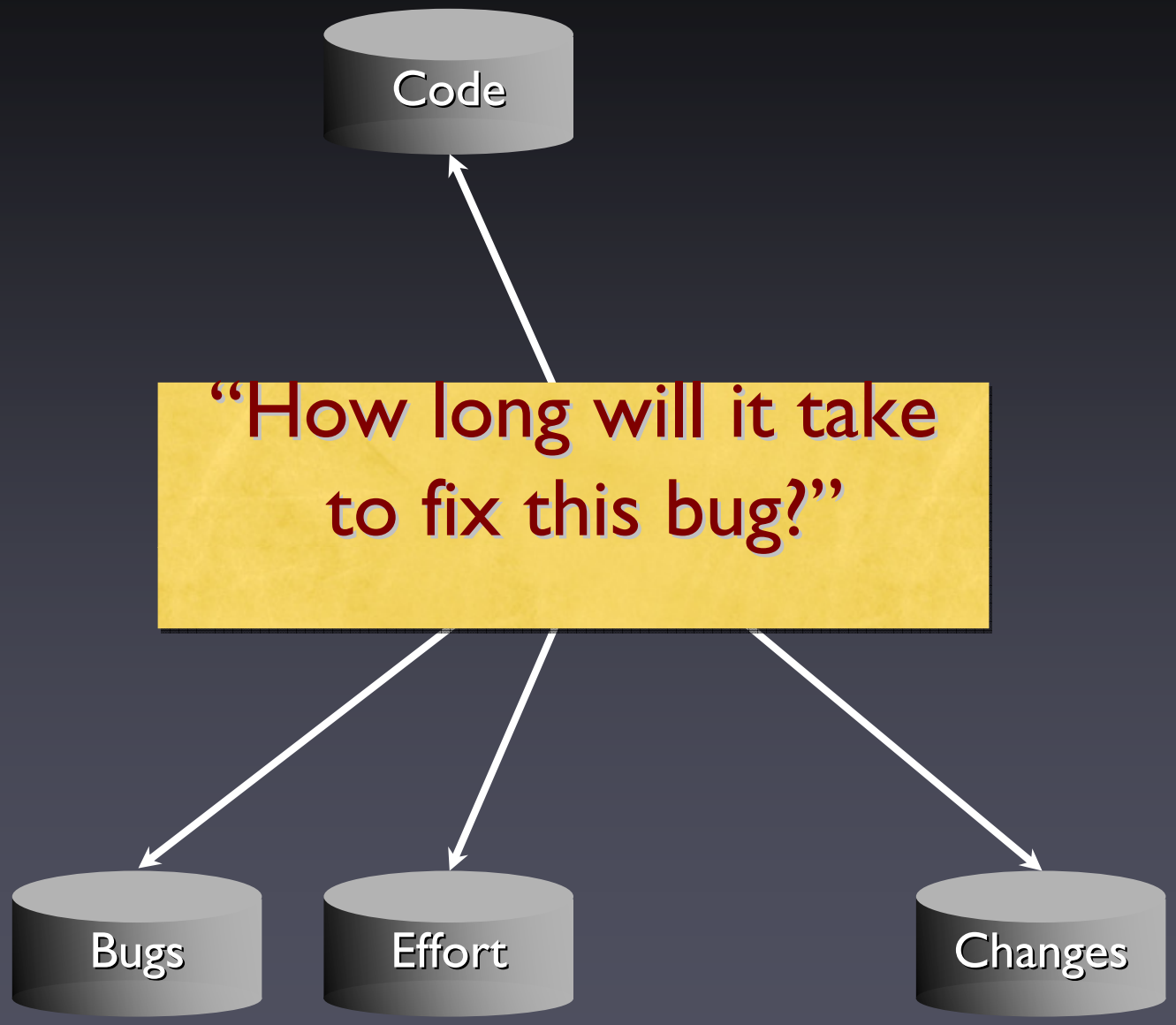
Software Archives

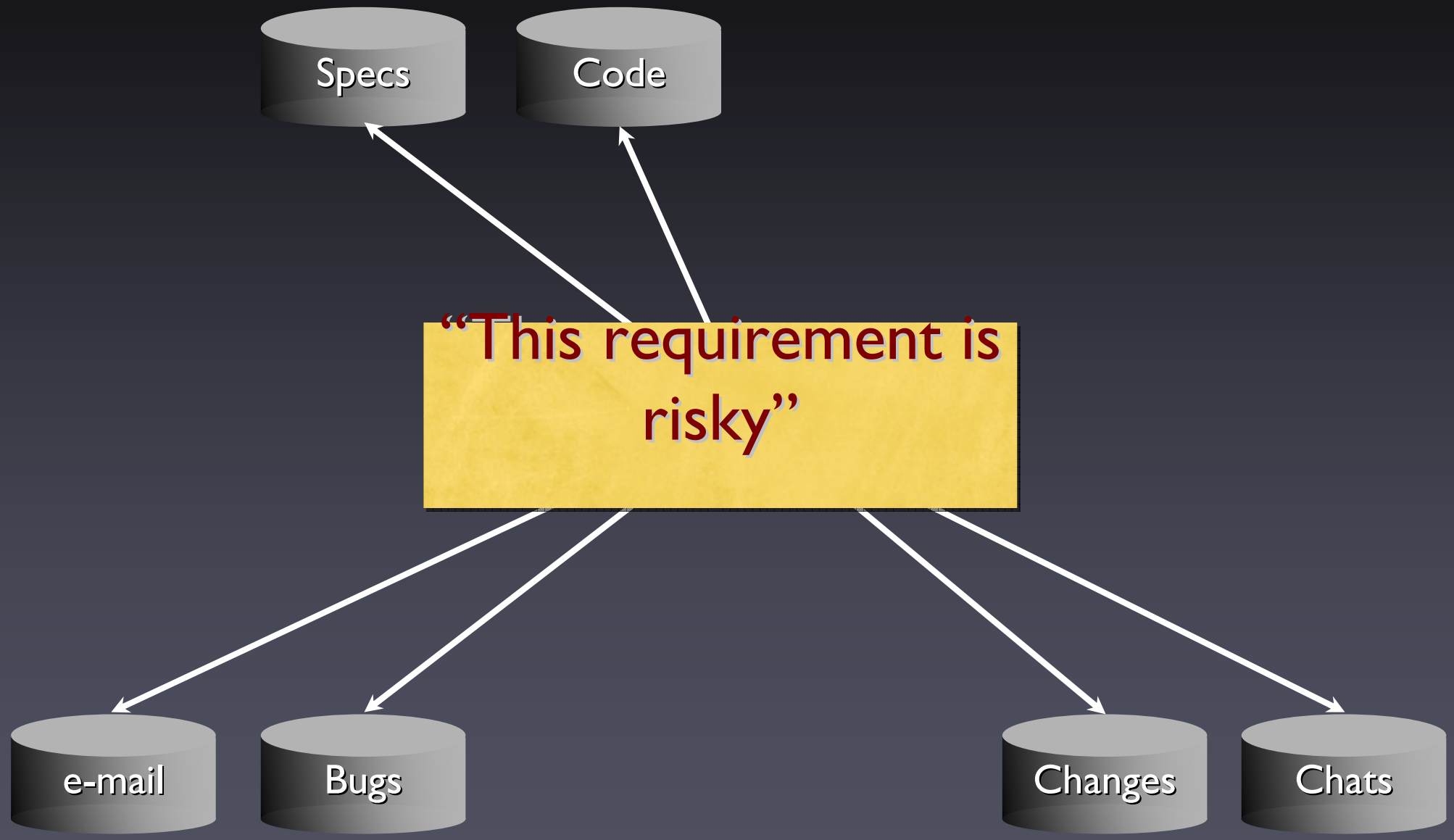
- contain full record of project history
- maintained via programming environments
- automatic maintenance and access
- freely accessible in open source projects

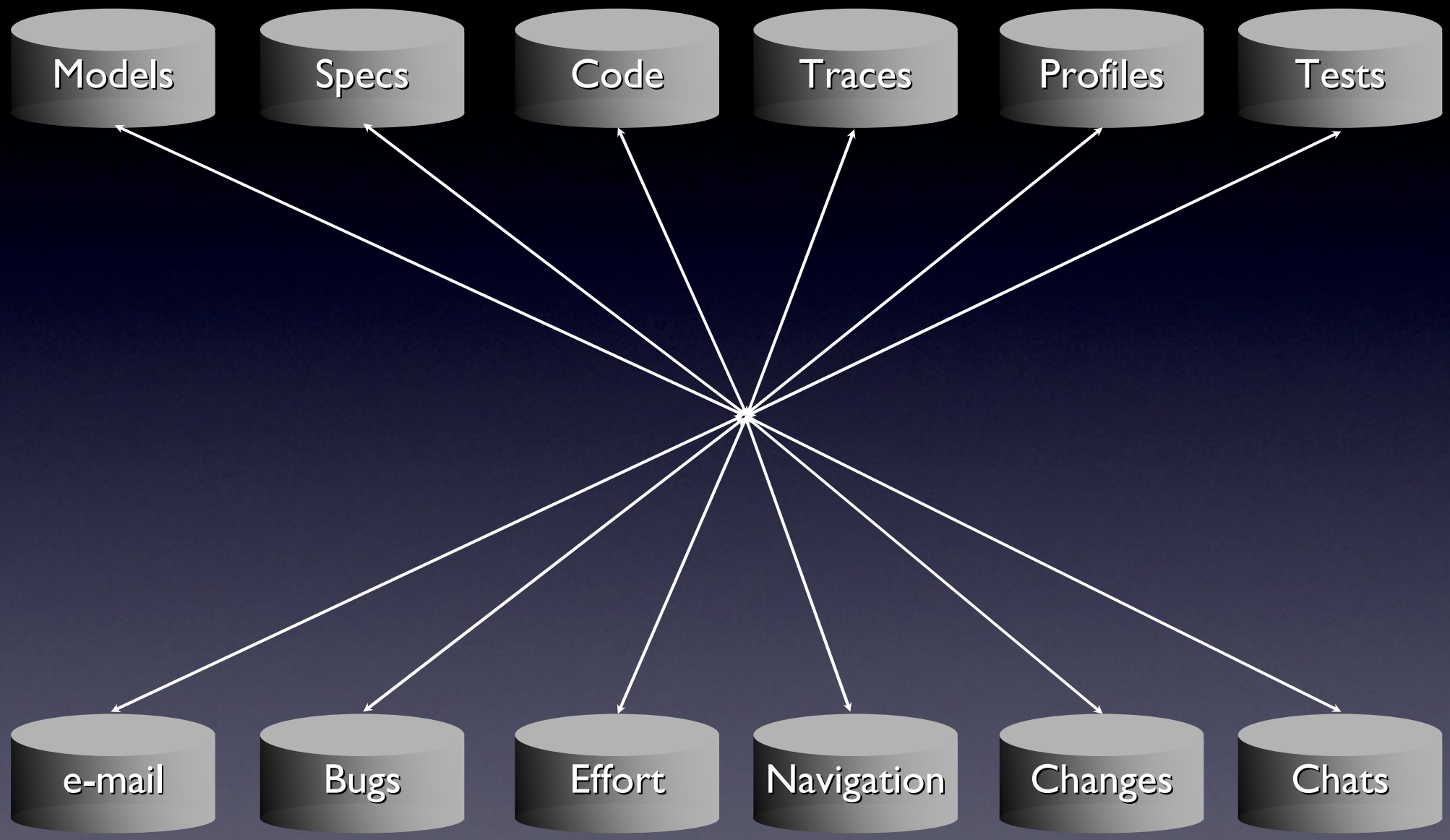

Models
Specs
Code
Traces
Profiles
Tests
e-mail
Bugs
Effort
Navigation
Changes
Chats


Models
Specs
Code
Traces
Profiles
Tests
e-mail
Bugs
Effort
Navigation
Changes
Chats


Code
Profiles
"Which modules should I test most?"
Bugs
Changes

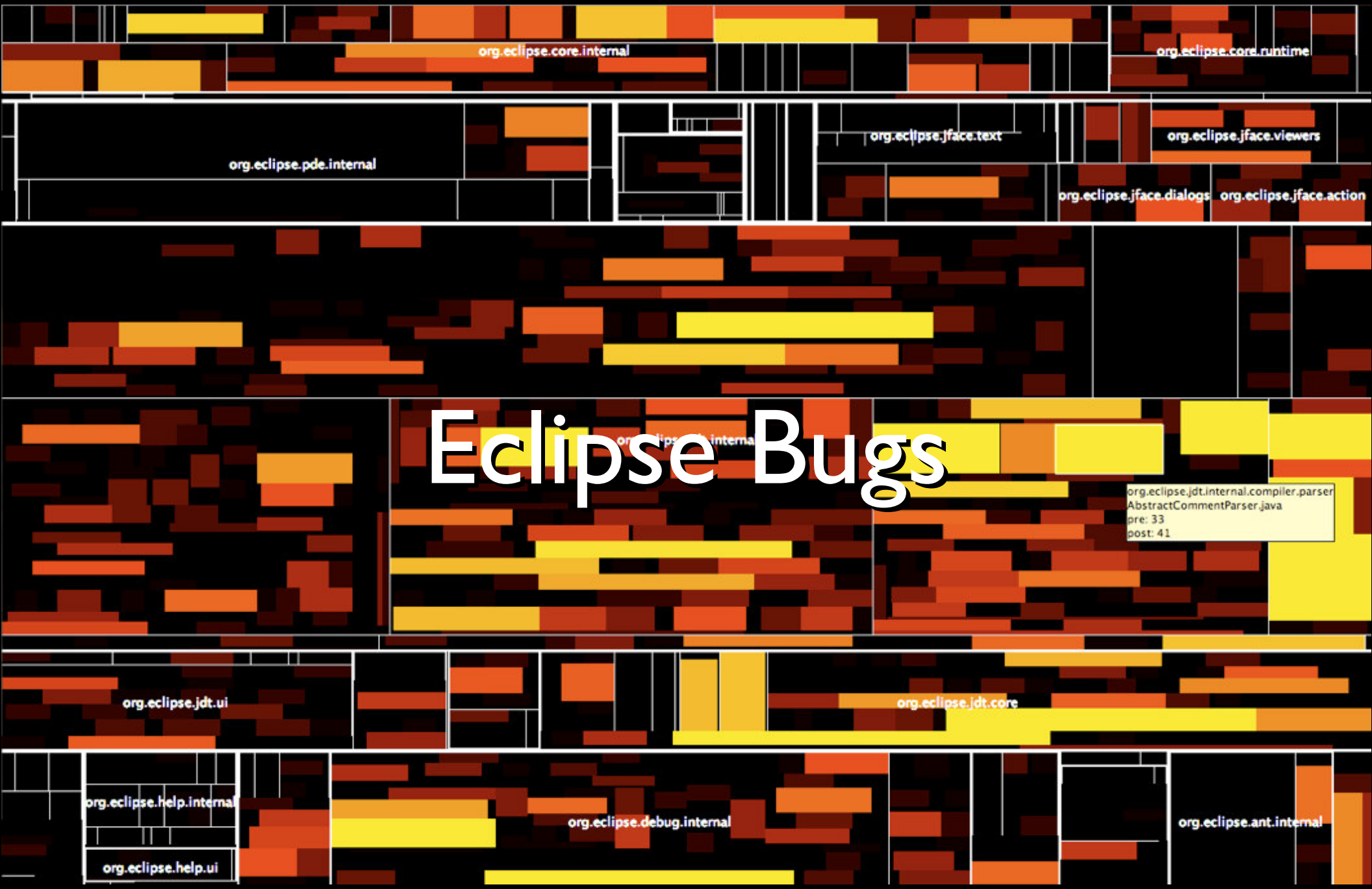

Code
"How long will it take to fix this bug?"
Bugs
Effort
Changes


Specs
Code
"This requirement is risky"
e-mail
Bugs
Changes
Chats

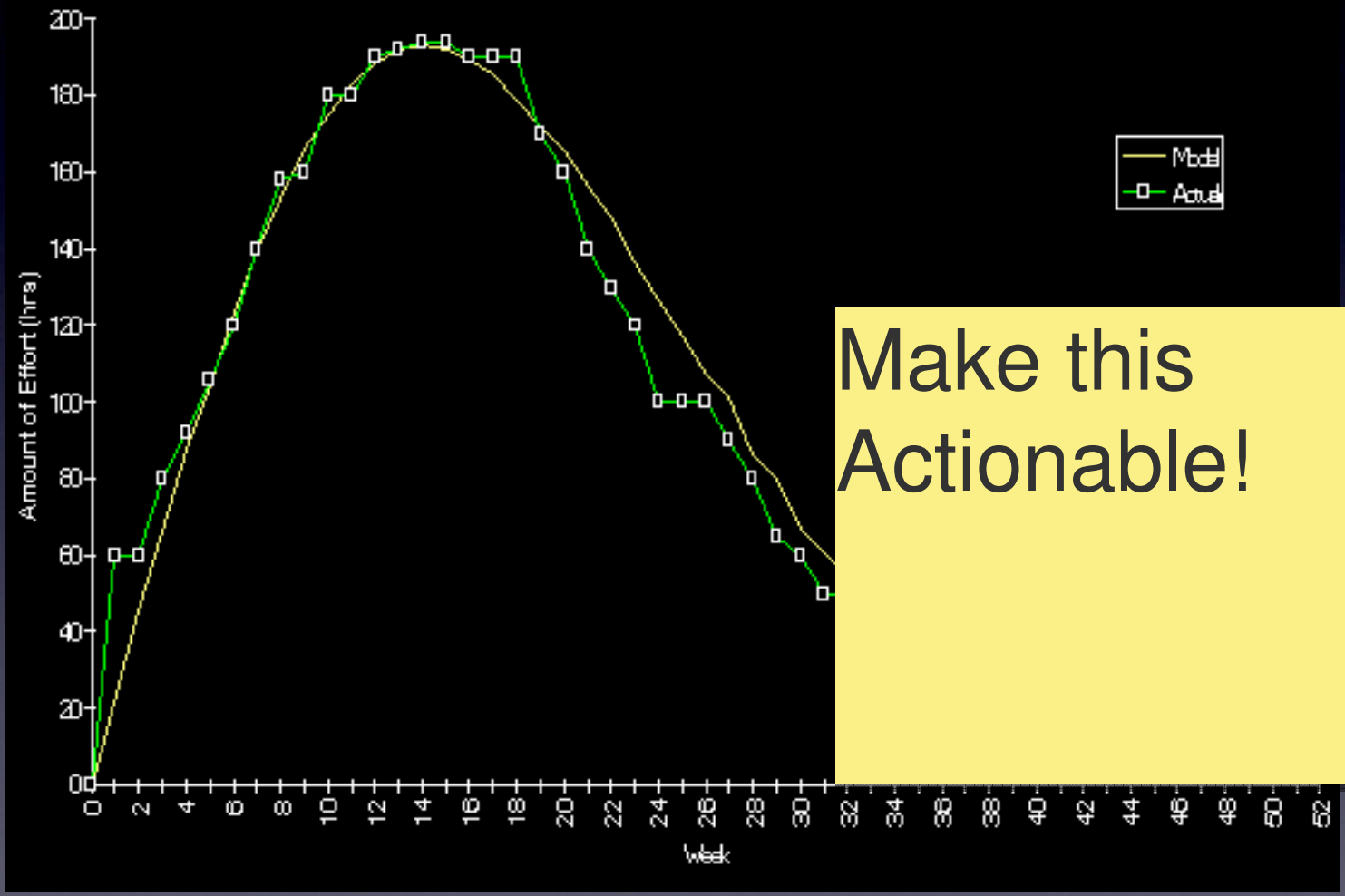

Models
Specs
Code
Traces
Profiles
Tests
e-mail
Bugs
Effort
Navigation
Changes
Chats

Very often
Often
Sometimes

Eclipse Bugs



Studies



Make this Actionable!

Rosenberg, L. and Hyatt, L. "Developing An Effective Metrics Program"
European Space Agency Software Assurance Symposium, Netherlands, March, 1996

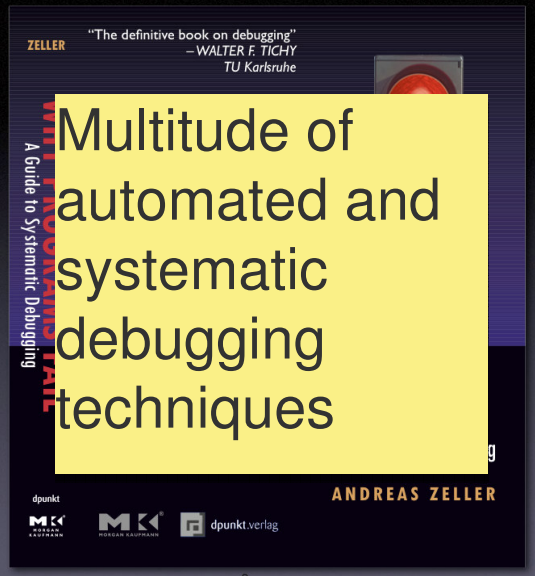
Bridge-City-Hall Station
Theatre-Aly
Church St
240
yards
0.30 ld
33 mph
10:13 a
10:15 a
Broadway
tomtom

Assistance

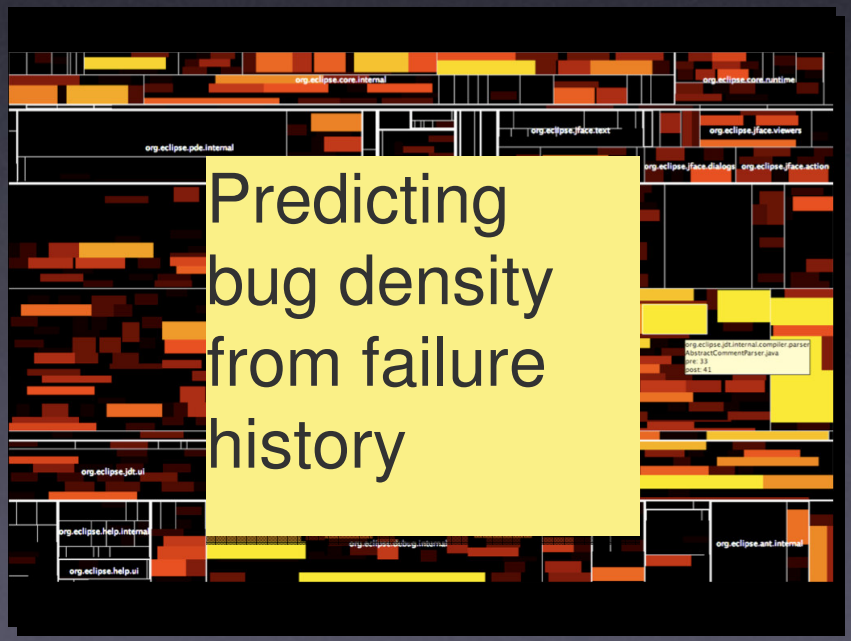
Future environments will

- mine patterns from program + process
- apply rules to make predictions
- provide assistance in all development decisions
- adapt advice to project history

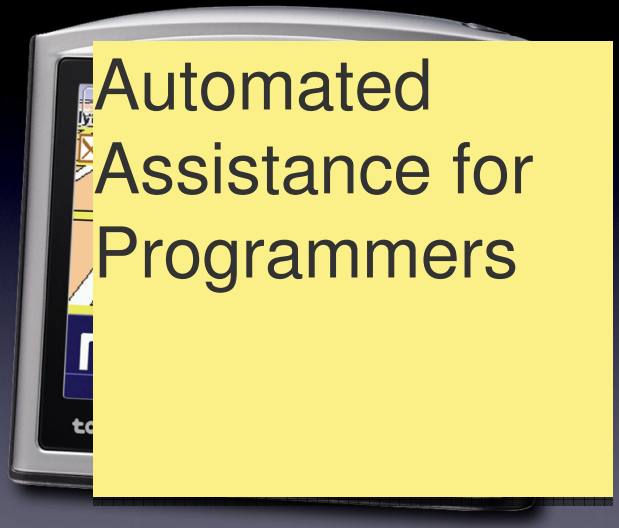
Locating Errors automatically



Multitude of automated and systematic debugging techniques



Predicting bug density from failure history



Automated Assistance for Programmers

This work is licensed under the Creative Commons Attribution License. To view a copy of this license, visit

http://creativecommons.org/licenses/by/1.0

or send a letter to Creative Commons, 559 Abbott Way, Stanford, California 94305, USA.