

Simplifying Problems

Andreas Zeller



1

Simplifying

- Once one has tracked and reproduced a problem, one must find out *what's relevant*:
 - Does the problem really depend on 10,000 lines of input?
 - Does the failure really require this exact schedule?
 - Do we need this sequence of calls?

2

2

Why simplify?



3

3

<http://www.concordesst.com/accident/accidentindex.html>

Simplifying

- For every circumstance of the problem, check whether it is relevant for the problem to occur.
- If it is not, remove it from the problem report or the test case in question.

4

4

Circumstances

- Any aspect that may influence a problem is a *circumstance*:
 - Aspects of the problem environment
 - Individual steps of the problem history

5

5

Experimentation

- By *experimentation*, one finds out whether a circumstance is relevant or not:
 - Omit the circumstance and try to reproduce the problem.
 - The circumstance is relevant iff the problem no longer occurs.

6

6

Mozilla Bug #24735

Ok the following operations cause mozilla to crash consistently on my machine

- > Start mozilla
- > Go to bugzilla.mozilla.org
- > Select search for bug
- > Print to file setting the bottom and right margins to .50 (I use the file /var/tmp/netscape.ps)
- > Once it's done printing do the exact same thing again on the same file (/var/tmp/netscape.ps)
- > This causes the browser to crash with a segfault

7

7

```
<SELECT NAME="op_sys" MULTIPLE SIZE=7>
<OPTION VALUE="All">All<OPTION VALUE="Windows 3.1">Windows 3.1<OPTION
VALUE="Windows 95">Windows 95<OPTION VALUE="Windows 98">Windows
98<OPTION VALUE="Windows NT">Windows NT<OPTION VALUE="Windows
2000">Windows 2000<OPTION VALUE="Windows ME">Windows ME<OPTION
VALUE="Mac System 7">Mac System 7<OPTION VALUE="Mac System 7.5">Mac
System 7.5<OPTION VALUE="Mac System 7.6.1">Mac System 7.6.1<OPTION
VALUE="Mac System 8.0">Mac System 8.0<OPTION VALUE="Mac System
8.5">Mac System 8.5<OPTION VALUE="Mac System 8.6">Mac System
8.6<OPTION VALUE="Mac System 9.x">Mac System 9.x<OPTION VALUE="MacOS
X">MacOS X<OPTION VALUE="Linux">Linux<OPTION VALUE="BSDI">BSDI<OPTION
VALUE="FreeBSD">FreeBSD<OPTION VALUE="NetBSD">NetBSD<OPTION
VALUE="OpenBSD">OpenBSD<OPTION VALUE="AT&T UNIX">AT&T UNIX<OPTION
VALUE="BeOS">BeOS<OPTION VALUE="IRIX">IRIX<OPTION VALUE="OS/2">OS/2<OPTION
VALUE="OpenVMS">OpenVMS<OPTION VALUE="OS/390">OS/390<OPTION VALUE="OSF/
1">OSF/1<OPTION VALUE="Solaris">Solaris<OPTION
VALUE="SunOS">SunOS<OPTION VALUE="other">other</SELECT>

</td>
<td align=left valign=top>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<OPTION VALUE="--">--<OPTION VALUE="P1">P1<OPTION VALUE="P2">P2<OPTION
VALUE="P3">P3<OPTION VALUE="P4">P4<OPTION VALUE="P5">P5</SELECT>
```

bugzilla.mozilla.org

What's relevant in here?

8

Why simplify?

- **Ease of communication.** A simplified test case is easier to communicate.
- **Easier debugging.** Smaller test cases result in smaller states and shorter executions.
- **Identify duplicates.** Simplified test cases *subsume* several duplicates.

9

9

The Gecko BugAThon

- Download the Web page to your machine.
- Using a text editor, start removing HTML from the page. Every few minutes, make sure it still reproduces the bug.
- Code not required to reproduce the bug can be safely removed.
- When you've cut away as much as you can, you're done.

10

10

Rewards

- 5 bugs - invitation to the *Gecko* launch party
- 10 bugs - the invitation, plus an attractive *Gecko* stuffed animal
- 12 bugs - the invitation, plus an attractive *Gecko* stuffed animal autographed by Rick Gessner, the Father of *Gecko*
- 15 bugs - the invitation, plus a *Gecko* T-shirt
- 20 bugs - the invitation, plus a *Gecko* T-shirt signed by the whole raptor team

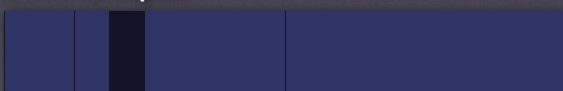
11

11

Binary Search

- Proceed by binary search. Throw away half the input and see if the output is still wrong.
- If not, go back to the previous state and discard the other half of the input.

HTML input



12

12

Simplified Input

```
<SELECT NAME="priority" MULTIPLE SIZE=7>
```

- Simplified from 896 lines to one single line
- Required 12 tests only

13

13

Benefits

- **Ease of communication.** All one needs is “Printing <SELECT> crashes”.
- **Easier debugging.** We can directly focus on the piece of code that prints <SELECT>.
- **Identify duplicates.** Check other test cases whether they’re <SELECT>-related, too.

14

14

Why automate?

- Manual simplification is *tedious*.
- Manual simplification is *boring*.
- We have machines for tedious and boring tasks.

15

15

Basic Idea

- We set up an *automated test* that checks whether the failure occurs or not (= Mozilla crashes when printing or not)
- We implement a *strategy* that realizes the binary search.

16

16

Automated Test

1. Launch Mozilla
2. Replay (previously recorded) steps from problem report
3. Wait to see whether
 - Mozilla crashes (= the test *fails*)
 - Mozilla still runs (= the test *passes*)
4. If neither happens, the test is *unresolved*

17

17

Binary Search

<SELECT NAME="priority" MULTIPLE SIZE=7> ✗

<SELECT NAME="priority" MULTIPLE SIZE=7> ✓

What do we do if both halves pass? ✓

<SELECT NAME="priority" MULTIPLE SIZE=7> ✓

<SELECT NAME="priority" MULTIPLE SIZE=7> ✓

<SELECT NAME="priority" MULTIPLE SIZE=7> ✗

<SELECT NAME="priority" MULTIPLE SIZE=7> ✗

<SELECT NAME="priority" MULTIPLE SIZE=7> ✓

18

18

Configuration

Circumstance

δ

All circumstances

$$C = \{\delta_1, \delta_2, \dots\}$$

Configuration $c \subseteq C$

$$c = \{\delta_1, \delta_2, \dots, \delta_n\}$$

19

19

Tests

Testing function

$$test(c) \in \{\checkmark, \times, ?\}$$

Failure-inducing configuration

$$test(c_x) = \times$$

Relevant configuration $c'_x \subseteq c_x$

$$\forall \delta_i \in c'_x \cdot test(c'_x \setminus \{\delta_i\}) \neq \times$$

20

20

Binary Strategy

Split input

$$c_x = c_1 \cup c_2$$

If removing first half fails...

$$test(c_x \setminus c_1) = \times \implies c'_x = c_x \setminus c_1$$

If removing second half fails...

$$test(c_x \setminus c_2) = \times \implies c'_x = c_x \setminus c_2$$

Otherwise, increase granularity:

$$c_x = c_1 \cup c_2 \cup c_3 \cup c_4$$

$$c_x = c_1 \cup c_2 \cup c_3 \cup c_4 \cup c_5 \cup c_6 \cup c_7 \cup c_8$$

21

21

General Strategy

Split input into n parts (initially 2)

$$c_x = c_1 \cup c_2 \cup \dots \cup c_n$$

If some removal fails...

$$\exists i \in \{1, \dots, n\} \cdot \text{test}(c_x \setminus c_i) = \text{X} \implies \begin{cases} c_x' = c_x \setminus c_i \\ n' = \max(n - 1, 2) \end{cases}$$

Otherwise, increase granularity

$$c_x' = c_x \quad n' = 2n$$

22

22

ddmin in a Nutshell

$c_x' = \text{ddmin}(c_x)$ is a relevant configuration

$\text{ddmin}(c_x) = \text{ddmin}'(c_x', 2)$ with $\text{ddmin}'(c_x', n) =$

$$\begin{cases} c_x' & \text{if } |c_x'| = 1 \\ \text{ddmin}'(c_x' \setminus c_i, \max(n - 1, 2)) & \text{else if } \exists i \in \{1..n\} \cdot \text{test}(c_x' \setminus c_i) = \text{X} \\ & \text{("some removal fails")} \\ \text{ddmin}'(c_x', \min(2n, |c_x'|)) & \text{else if } n < |c_x'| \text{ ("increase granularity")} \\ c_x' & \text{otherwise} \end{cases}$$

where $c_x' = c_1 \cup c_2 \cup \dots \cup c_n$

$$\forall c_i, c_j \cdot c_i \cap c_j = \emptyset \wedge |c_i| \approx |c_j|$$

23

23

```
def _ddmin(circumstances, n):
    while len(circumstances) >= 2:
        subsets = split(circumstances, n)

        some_complement_is_failing = 0
        for subset in subsets:
            complement = listminus(circumstances, subset)
            if test(complement) == FAIL:
                circumstances = complement
                n = max(n - 1, 2)
                some_complement_is_failing = 1
                break

        if not some_complement_is_failing:
            if n == len(circumstances):
                break
            n = min(n * 2, len(circumstances))

    return circumstances
```

24

24

ddmin at Work

Input: <SELECT NAME="priority" MULTIPLE SIZE=7> (40 characters) ✗
<SELECT NAME="priority" MULTIPLE SIZE=7> (0 characters) ✓

```
1 <SELECT NAME="priority" MULTIPLE SIZE=7> (20) ✓ 25 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
2 <SELECT NAME="priority" MULTIPLE SIZE=7> (20) ✓ 26 <SELECT NAME="priority" MULTIPLE SIZE=7> (8) ✓
3 <SELECT NAME="priority" MULTIPLE SIZE=7> (30) ✓ 27 <SELECT NAME="priority" MULTIPLE SIZE=7> (9) ✓
4 <SELECT NAME="priority" MULTIPLE SIZE=7> (30) ✗ 28 <SELECT NAME="priority" MULTIPLE SIZE=7> (9) ✓
5 <SELECT NAME="priority" MULTIPLE SIZE=7> (20) ✓ 29 <SELECT NAME="priority" MULTIPLE SIZE=7> (9) ✓
6 <SELECT NAME="priority" MULTIPLE SIZE=7> (20) ✗ 30 <SELECT NAME="priority" MULTIPLE SIZE=7> (9) ✓
7 <SELECT NAME="priority" MULTIPLE SIZE=7> (10) ✓ 31 <SELECT NAME="priority" MULTIPLE SIZE=7> (8) ✓
8 <SELECT NAME="priority" MULTIPLE SIZE=7> (10) ✓ 32 <SELECT NAME="priority" MULTIPLE SIZE=7> (9) ✓
9 <SELECT NAME="priority" MULTIPLE SIZE=7> (15) ✓ 33 <SELECT NAME="priority" MULTIPLE SIZE=7> (8) ✗
10 <SELECT NAME="priority" MULTIPLE SIZE=7> (15) ✓ 34 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
11 <SELECT NAME="priority" MULTIPLE SIZE=7> (15) ✗ 35 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
12 <SELECT NAME="priority" MULTIPLE SIZE=7> (10) ✓ 36 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
13 <SELECT NAME="priority" MULTIPLE SIZE=7> (10) ✓ 37 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
14 <SELECT NAME="priority" MULTIPLE SIZE=7> (10) ✓ 38 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
15 <SELECT NAME="priority" MULTIPLE SIZE=7> (12) ✓ 39 <SELECT NAME="priority" MULTIPLE SIZE=7> (6) ✓
16 <SELECT NAME="priority" MULTIPLE SIZE=7> (13) ✓ 40 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
17 <SELECT NAME="priority" MULTIPLE SIZE=7> (12) ✓ 41 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
18 <SELECT NAME="priority" MULTIPLE SIZE=7> (13) ✗ 42 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
19 <SELECT NAME="priority" MULTIPLE SIZE=7> (10) ✓ 43 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
20 <SELECT NAME="priority" MULTIPLE SIZE=7> (10) ✓ 44 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
21 <SELECT NAME="priority" MULTIPLE SIZE=7> (11) ✓ 45 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
22 <SELECT NAME="priority" MULTIPLE SIZE=7> (10) ✗ 46 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
23 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓ 47 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
24 <SELECT NAME="priority" MULTIPLE SIZE=7> (8) ✓ 48 <SELECT NAME="priority" MULTIPLE SIZE=7> (7) ✓
```

Result: <SELECT>

25

25

Complexity

- The maximal number of *ddmin* tests is

$$\frac{(|c_x|^2 + 7|c_x|)}{2}$$

26

26

Worst Case Details

First phase: every test is unresolved

$$t = 2 + 4 + 8 + \dots + 2|c_x| \\ = 2|c_x| + |c_x| + \frac{|c_x|}{2} + \frac{|c_x|}{4} + \dots = 4|c_x|$$

Second phase: testing *last* set always fails

$$t' = (|c_x| - 1) + (|c_x| - 2) + \dots + 1 \\ = 1 + 2 + 3 + \dots + (|c_x| - 1) \\ = \frac{|c_x|(|c_x| - 1)}{2} = \frac{|c_x|^2 - |c_x|}{2}$$

27

27

Binary Search

If

- there is only one failure-inducing circumstance, and
- all configurations that include this circumstance fail,

the number of tests is $t \leq \log_2(|C_n|)$

28

28

More Simplification

Simplified failure-inducing *fuzz* input:

- FLEX crashes on 2,121 or more non-newline characters
- NROFF crashes on “\D^J%0F” or “\302\n”
- CRTPLOT crashes on “t”

29

29

Minimal Interaction

Ok the following operations cause mozilla to crash consistently on my machine

- > Start mozilla
- > Go to bugzilla.mozilla.org
- > Select search for bug
- > Print to file setting the bottom and right margins to .50 (I use the file `/var/tmp/netscape.ps`)
- > Once it's done printing do the exact same thing again on the same file (`/var/tmp/netscape.ps`)
- > This causes the browser to crash with a segfault

30

30

Minimal Interaction

Basic idea:

Apply *ddmin* to recorded user interaction

- To reproduce the Mozilla printing crash:
 - Press *P* while holding *Alt*
 - Press *mouse button I*
 - Release *mouse button I*

31

31

Optimization

- Caching
- Stop Early
- Syntactic Simplification
- Isolate Differences, not Circumstances

32

32

Caching

- Basic idea: store the results of earlier test()
- Saves 8 out of 48 tests in `<SELECT>` example

33

33

Stop Early

One may stop simplification when

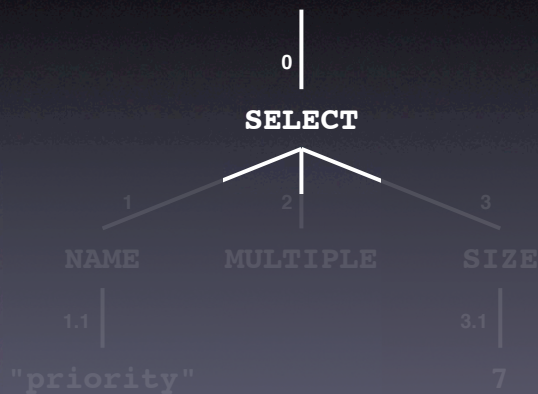
- a certain *granularity* has been reached
- no *progress* has been made
- a certain *amount of time* has elapsed

34

34

Syntactic Simplification

```
<SELECT NAME="priority" MULTIPLE SIZE=7>
```



35

Differences

```
<SELECT NAME="priority" MULTIPLE SIZE=7>
```

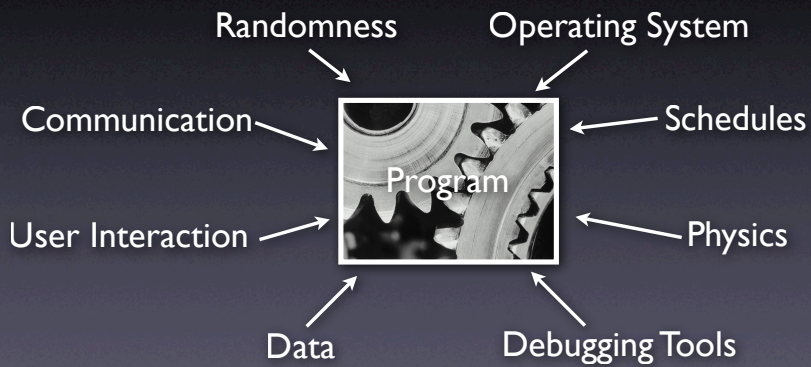
The extra "<" is failure-inducing!

```
<SELECT NAME="priority" MULTIPLE SIZE=7>
```

36

36

More Circumstances



37

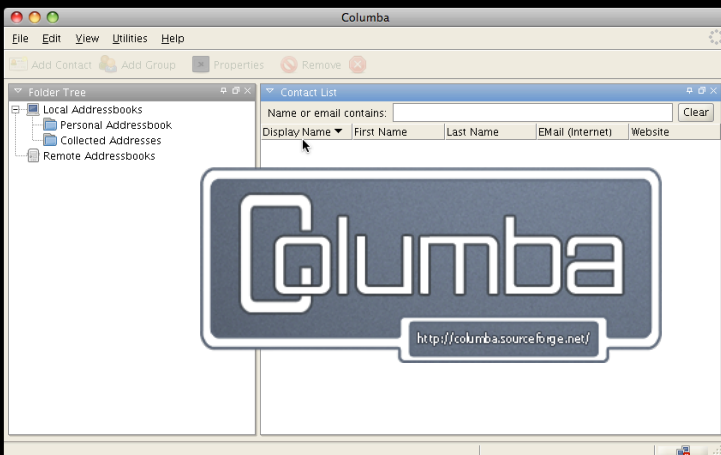
37

More Automation

- Failure-Inducing Input
- Failure-Inducing Code Changes
- Failure-Inducing Schedules
- Failure-Inducing Program States
- **Failure-Inducing Method Calls**

38

38



39

Lenhof, Hans-Peter,, +49 681 302-64701, lenhof@cs.uni-sb.de
Lindig, Christian, +49 681 9378406, +49 681 302 7790, lindig@cs.uni-sb.de
Mehlmann, Martin,, mehlmann@st.cs.uni-sb.de
Meyer zu Tittingdorf, +49 681 302-78099, meyer@cs.uni-sb.de
Mileva, Yana,, +49 681 302-64020, mileva@cs.uni-sb.de
Müller-Perich, Elisabeth,, +49 681 302-7070, sekr.techfak@rz.uni-sb.de
Nir-Bleimling, Naomi,, +49 68130264011, naomi@wjpserver.cs.uni-sb.de
Offergeld, Thilo,, +49 681 302-6794, t.offergeld@univw.uni-sb.de
PC, CC 2006,, cc2006pc@st.cs.uni-sb.de
Paul, Wolfgang, +4968171827, +49 6813022436, wjp@cs.uni-sb.de
Premraj, Rahul, +44 7796973711, +49 681 302-64013, premraj@cs.uni-sb.de
Reindel, Erich, +49 6371 912842, +49 681 302-78091, reindel@cs.uni-sb.de
Schuler, David,, +49 681 302-64026, schuler@st.cs.uni-sb.de
Schuler, Erika,, +49 6813027069, schuler@tf.uni-sb.de
Schäfer, Christa, +49 6897 71167, +49 68130264011,
Security, AG,, security@st.cs.uni-sb.de
Seidel, Raimund, +49 6894 383698, +49 681 302-4713, rseidel@cs.uni-sb.de
Sekretariat, Sekretariat,, +49 681 302-64011, office@st.cs.uni-sb.de
Sliwerski, Jacek, +491741333208,, sliwers@st.cs.uni-sb.de
Slusallek, Philipp, +49 6826 1 88 71 32, +49 681 302-3830, slusallek@cs.uni-sb.de
Slusallek USA, Philipp, +1 670 391 9186, +1 408 486 2788, slusallek@cs.uni-sb.de
Smolka, Gert, +49 681 782770, +49 681 302-7311, smolka@ps.uni-sb.de
Software-Evolution, AG,, softevo@st.cs.uni-sb.de
Thiel, Frank,, hausmeister@cs.uni-sb.de
Weiß, Cathrin,, weiss@st.cs.uni-sb.de
Wilhelm, Reinhard,, +49 681 302-4399, wilhelm@cs.uni-sb.de
Zeller, Andreas,, zeller@cs.uni-sb.de
Zeller, Andreas, +49 681 3710467, +49 681 302-64011, zeller@cs.uni-sb.de
Zimmermann, Tom, +49 871 71742 (Eltern), +1 403 210 9470, zimmerth@cs.uni-sb.de

Failure Cause

40

Lenhof, Hans-Peter,, +49 681 302-64701, lenhof@cs.uni-sb.de
Lindig, Christian, +49 681 9378406, +49 681 302 7790, lindig@cs.uni-sb.de
Mehlmann, Martin,, mehlmann@st.cs.uni-sb.de
Meyer zu Tittingdorf, +49 681 302-78099, meyer@cs.uni-sb.de
Mileva, Yana,, +49 681 302-64020, mileva@cs.uni-sb.de
Müller-Perich, Elisabeth,, +49 681 302-7070, sekr.techfak@rz.uni-sb.de
Nir-Bleimling, Naomi,, +49 68130264011, naomi@wjpserver.cs.uni-sb.de
Offergeld, Thilo,, +49 681 302-6794, t.offergeld@univw.uni-sb.de
PC, CC 2006,, cc2006pc@st.cs.uni-sb.de
Paul, Wolfgang, +4968171827, +49 6813022436, wjp@cs.uni-sb.de
Premraj, Rahul, +44 7796973711, +49 681 302-64013, premraj@cs.uni-sb.de
Reindel, Erich, +49 6371 912842, +49 681 302-78091, reindel@cs.uni-sb.de
Schuler, David,, +49 681 302-64026, schuler@st.cs.uni-sb.de
Schuler, Erika,, +49 6813027069, schuler@tf.uni-sb.de
Schäfer, Christa, +49 6897 71167, +49 68130264011,
Security, AG,, security@st.cs.uni-sb.de
Seidel, Raimund, +49 6894 383698, +49 681 302-4713, rseidel@cs.uni-sb.de
Sekretariat, Sekretariat,, +49 681 302-64011, office@st.cs.uni-sb.de
Sliwerski, Jacek, +491741333208,, sliwers@st.cs.uni-sb.de
Slusallek, Philipp, +49 6826 1 88 71 32, +49 681 302-3830, slusallek@cs.uni-sb.de
Slusallek USA, Philipp, +1 670 391 9186, +1 408 486 2788, slusallek@cs.uni-sb.de
Smolka, Gert, +49 681 782770, +49 681 302-7311, smolka@ps.uni-sb.de
Software-Evolution, AG,, softevo@st.cs.uni-sb.de
Thiel, Frank,, hausmeister@cs.uni-sb.de
Weiß, Cathrin,, weiss@st.cs.uni-sb.de
Wilhelm, Reinhard,, +49 681 302-4399, wilhelm@cs.uni-sb.de
Zeller, Andreas,, zeller@cs.uni-sb.de
Zeller, Andreas, +49 681 3710467, +49 681 302-64011, zeller@cs.uni-sb.de
Zimmermann, Tom, +49 871 71742 (Eltern), +1 403 210 9470, zimmerth@cs.uni-sb.de

Failure Cause

Now, the idea is that we can easily automate the whole process.

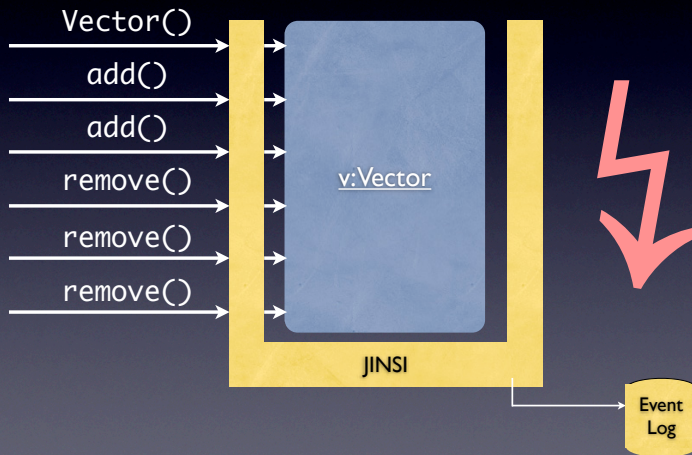
41

Problem:
Simulating user interaction
is cumbersome.

42

Isolating Relevant Calls

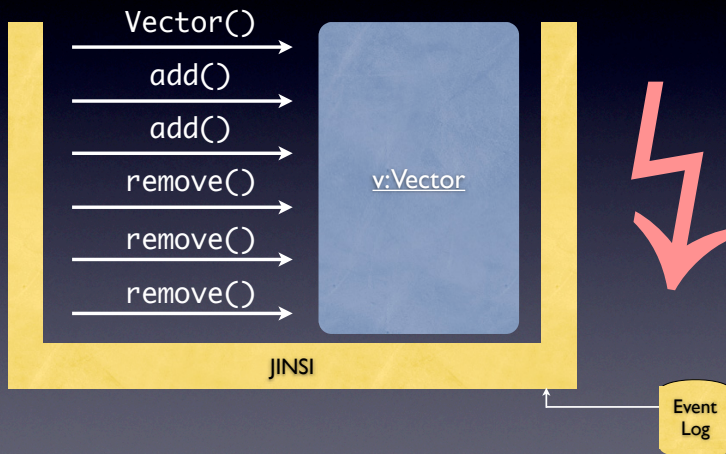
Step 1: Record



43

Isolating Relevant Calls

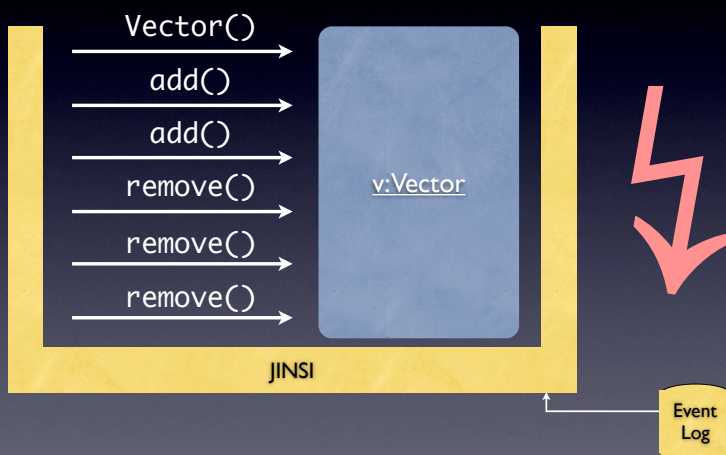
Step 2: Replay



44

Isolating Relevant Calls

Step 3: Simplify



45

Isolating Relevant Calls

Step 4: Create Unit Test

```
testVector()  
{  
    Vector v = new Vector();  
    v.remove(obj);  
}
```

JINSI



46

Columba ContactModel

ContactModel()
setSortString()
setFormattedName()
setNickName()
setFamilyName()
setGivenName()
and 18732 more...



47

Columba ContactModel

ContactModel()
getPreferredEmail()



48

Unit Test

```
testContactModel()
{
    ContactModel c = new ContactModel();
    String s = c.getPreferredEmail();
}
```

49

getPreferredEmail

```
public String getPreferredEmail() {
    Iterator it = getEmailIterator();

    // get first item
    IEmailModel model = (IEmailModel) it.next();

    // backwards compatibility
    // -> its not possible anymore to create a
    // contact model without email address
    if (model == null)
        return null;

    return model.getAddress();
}
```

50

Concepts

- ★ The aim of simplification is to create a simple *test case* from a problem report.
- ★ Simplified test cases...
 - are easier to communicate
 - facilitate debugging
 - identify duplicate problem reports

51

Concepts (2)

- ★ To simplify a test case, remove all irrelevant circumstances.
- ★ A circumstance is irrelevant if the problem occurs regardless of whether the circumstance is present or not.

52

52

Concepts (3)

- ★ To automate simplification, set up
 - an *automated test*
 - a *strategy* to determine the relevant circumstances
- ★ One such strategy is the *dmin* delta debugging algorithm

53

53

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.

54

54