Woher kommen Software-Fehler?

Andreas Zeller • Universität des Saarlandes

1

Woher kommen Software-Fehler?

Jeder Programmierer kennt die Situation: Ein Programm läuft nicht so, wie es soll. Ich stelle Techniken vor, die automatisch

(a) die Ursachen eines Fehlverhaltens finden - indem wir genau die Aspekte isolieren, die das Zustandekommen eines Fehlers verursachen;
(b) Programmfehler finden - indem wir aus dem Code "normale" Anweisungsfolgen lernen und nach Abweichungen suchen; und
(c) vorhersagen, wo in Zukunft Fehler auftreten werden - indem wir maschinell lernen, welche Code- und Prozesseigenschaften bisher mit Fehlern korrelierten.

Fallstudien an echten Programmen mit echten Fehlern, von AspectJ über Firefox zu Windows demonstrieren die Praxistauglichkeit der vorgestellten Verfahren.

Andreas Zeller ist Professor für Softwaretechnik an der Universität des Saarlandes in Saarbrücken. Sein Forschungsgebiet ist die Analyse großer Software-Systeme und deren Fehler. Sein Buch "Why Programs Fail - A Guide to Systematic Debugging" wurde 2006 mit dem Jolt Software Development Productivity Award ausgezeichnet.

Eine F-16

(Nördliche Halbkugel)

2

An F-16 fighter plane on the northern hemisphere.
Why the northern hemisphere, you ask?

Eine F-16

(Südliche Halbkugel)

3

Because this is what an F-16 on the southern hemisphere would look like.
(BTW, interesting effect if you drop a bomb :-)

From risks.digest, volume 3, issue 44:
o Since the F-16 is a fly-by-wire aircraft, the computer keeps the pilot from doing dumb things to himself. So if the pilot jerks hard over on the joystick, the computer will instruct the flight surfaces to make a nice and easy 4 or 5 G flip. But the plane can withstand a much higher flip than that. So when they were 'flying' the F-16 in simulation over the equator, the computer got confused and instantly flipped the plane over, killing the

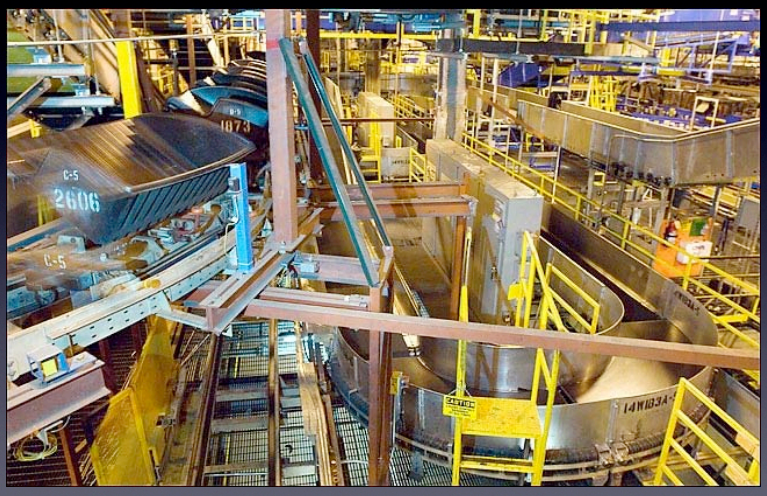
F-16 Fahrgestell

From risks.digest, volume 3, issue 44:
o One of the first things the Air Force test pilots tried on an early F-16 was to tell the computer to raise the landing gear while standing still on the runway. Guess what happened? Scratch one F-16. (my friend says there is a new subroutine in the code called 'wait_on_wheels' now...) [weight?]

(Folklore has it that the programmer checked the height above sea level rather than the height above ground - AZ)

4

Flughafen Denver



5

What camera crews depicted was truly a disaster; carts jammed together, damaged luggage everywhere, some bags literally split in half, and the tattered remains of clothing strewn about causing subsequent carts to derail. Finally, adding insult to injury, half the luggage that survived the ordeal ended up at the wrong terminal.

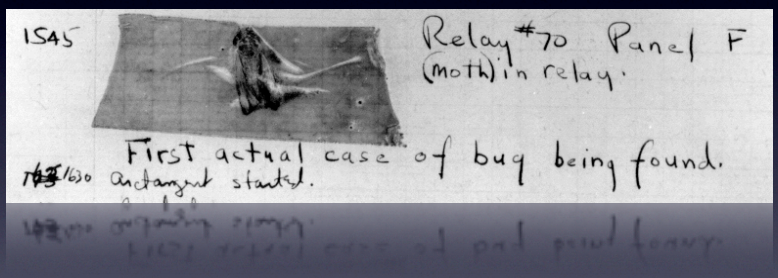
6

http://www.aeroxp.org/2009/01/lesson-on-infinite-loops/
http://www.youtube.com/watch?v=fYTJ9v2vsaE

7

Der erste Bug

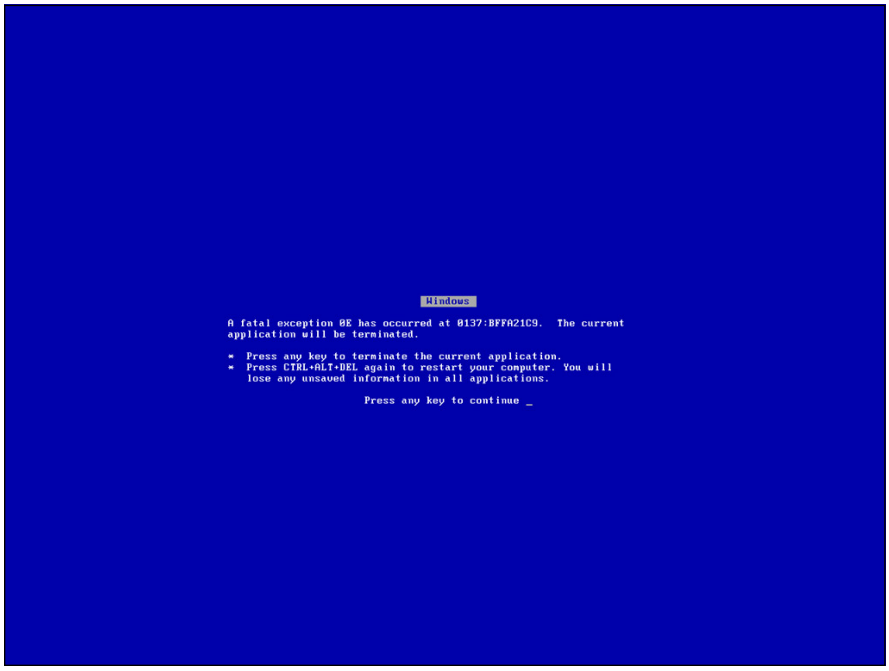
9. September 1947



Retrieved by a technician from the Harvard Mark II machine on September 9, 1947.

Now on display at the Smithsonian, Washington

8



9

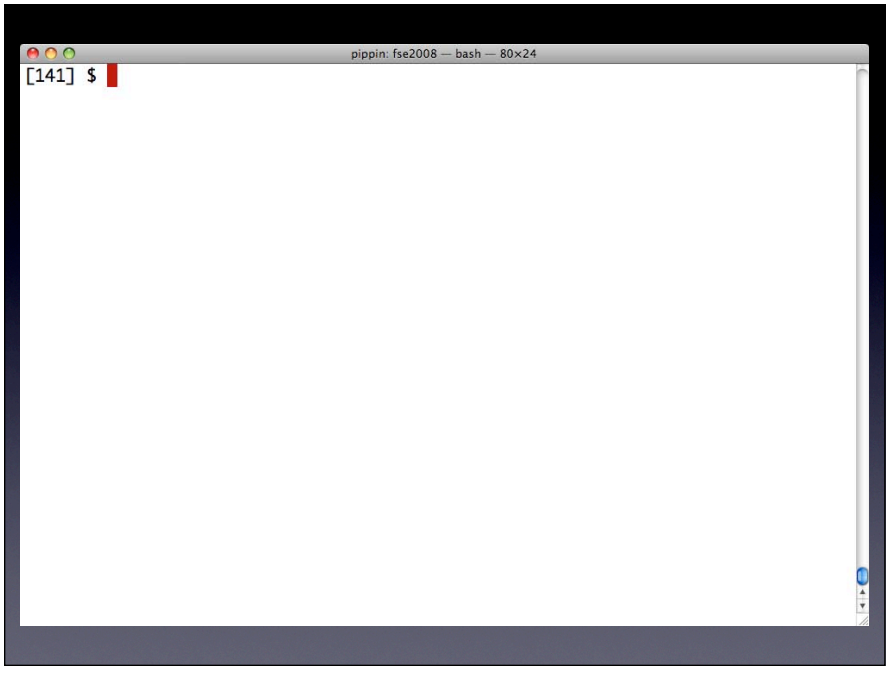
aspectj *crosscutting objects for better modularity*

10

bug.aj

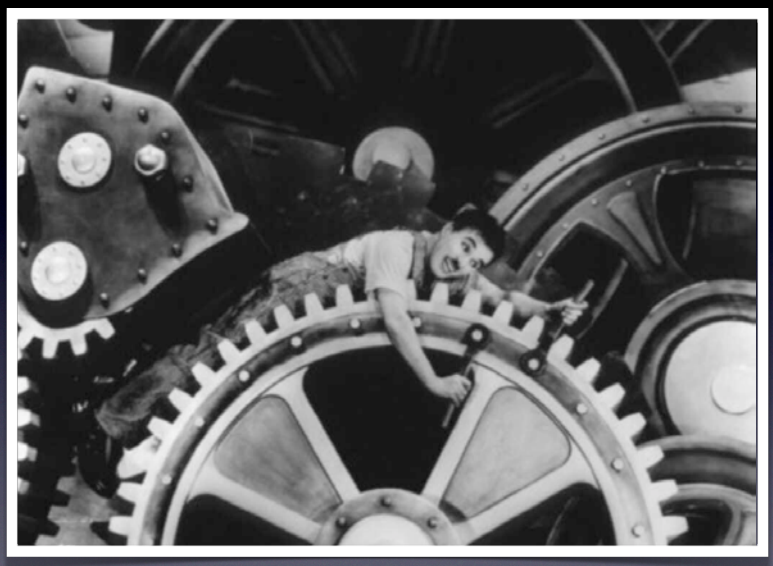
```
@interface A {}

aspect Test {
  declare @field : @A int var* : @A;
  declare @field : int var* : @A;

  interface Subject {}

  public int Subject.vara;
  public int Subject.varb;
}

class X implements Test.Subject {}
```

11



12

13



14

Diagnose

Erkennen

Vorbeugen

15

Diagnose Erkennen Vorbeugen



bug.aj

```
@interface A {}

aspect Test {
  declare @field : @A int var* : @A;
  declare @field : int var* : @A;

  interface Subject {}

  public int Subject.vara;
  public int Subject.varb;
}

class X implements Test.Subject {}
```

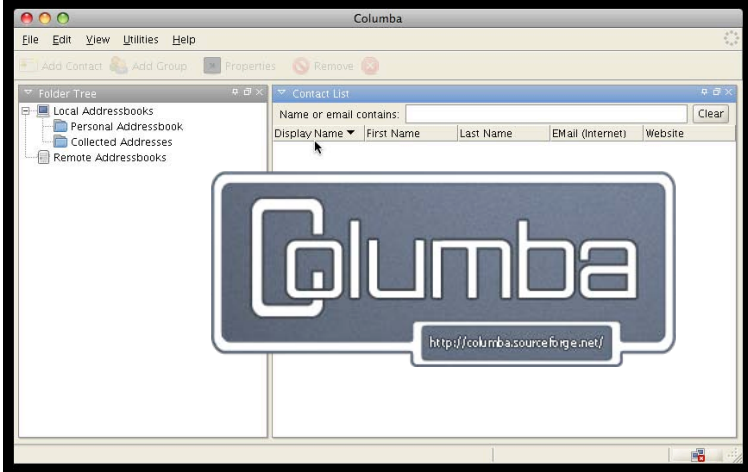

ajc Stack Trace

```
java.util.NoSuchElementException
  at java.util.AbstractList$Itr
     .next(AbstractList.java:427)
  at org.aspectj.weaver.bcel.BcelClassWeaver
     .weaveAtFieldRepeatedly
     (BcelClassWeaver.java:1016)
```


We can fix this by looking at the stack trace.

weaveAtFieldRepeatedly

```
for (Iterator iter = itdFields.iterator();
     iter.hasNext();) {
    ...
    for (Iterator iter2 = worthRetrying.iterator();
         iter.hasNext();) {
        ...
    }
}
```

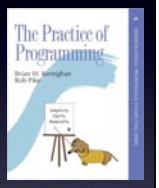



mozilla.csv

Lenhof, Hans-Peter,, +49 681 302-64701, lenhof@cs.uni-sb.de
Lindig, Christian, +49 681 9358406, +49 681 302 5590, lindig@cs.uni-sb.de
Mehlmann, Martin,,, mehlmann@st.cs.uni-sb.de
Meyer zu Tittingdorf, Friedemann, +49 681 302-64026, +49 681 302-58099, meyer@cs.uni-sb.de
Mileva, Yana,, +49 681 302-64020, mileva@cs.uni-sb.de
Müller-Perich, Elisabeth,, +49 681 302-5070, sekr.techfak@rz.uni-sb.de
Nir-Bleimling, Naomi,, +49 68130264011, naomi@wjpserver.cs.uni-sb.de
Offergeld, Thilo,, +49 681 302-6594, t.offergeld@univw.uni-sb.de
PC, CC 2006,,, cc2006pc@st.cs.uni-sb.de
Paul, Wolfgang, +4968171825, +49 6813022436, wjp@cs.uni-sb.de
Premraj, Rahul, +44 7796953511, +49 681 302-64013, premraj@cs.uni-sb.de
Reindel, Erich, +49 6371 912842, +49 681 302-58091, reindel@cs.uni-sb.de
Schuler, David,, +49 681 302-64026, schuler@st.cs.uni-sb.de
Schuler, Erika,, +49 6813025069, schuler@tf.uni-sb.de
Schäfer, Christa, +49 6897 51165, +49 68130264011,
Security, AG,,, security@st.cs.uni-sb.de
Seidel, Raimund, +49 6894 383698, +49 681 302-4513, 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, +1 650 391 9186, +1 408 486 2788, slusallek@cs.uni-sb.de
Smolka, Gert, +49 681 582770, +49 681 302-5311, 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 3710465, +49 681 302-64011, zeller@cs.uni-sb.de
Zimmermann, Tom, +49 851 51542 (Eltern), +1 403 210 9470, zimmerth@cs.uni-sb.de

Vereinfachen

- 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.



mozilla.csv

22

Fehlerursache

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, [illegible] 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

23

Fehlerursache

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, [illegible] 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

24

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

Problem:
Vereinfachen von Hand
ist grausam!

25

Delta Debugging

Delta Debugging *isoliert automatisch Fehlerursachen:*

Eingaben: 1 von 436 Kontakten in Columba

Änderungen: 1 von 8,721 Änderungen in GDB

Threads: 1 von 3.8 bio Thread-Wechsel in Scene.java

➔ *Automatisierte Fehlersuche*

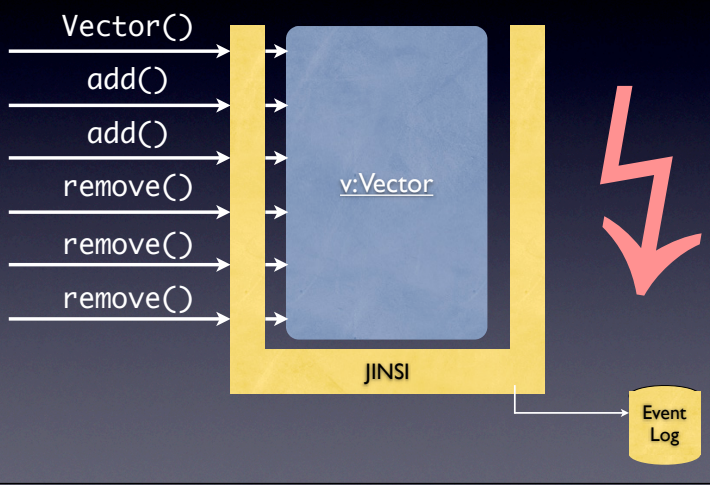
26

Problem:
Benutzeraktionen simulieren
ist umständlich!

27

Isolieren Relevanter Aufrufe

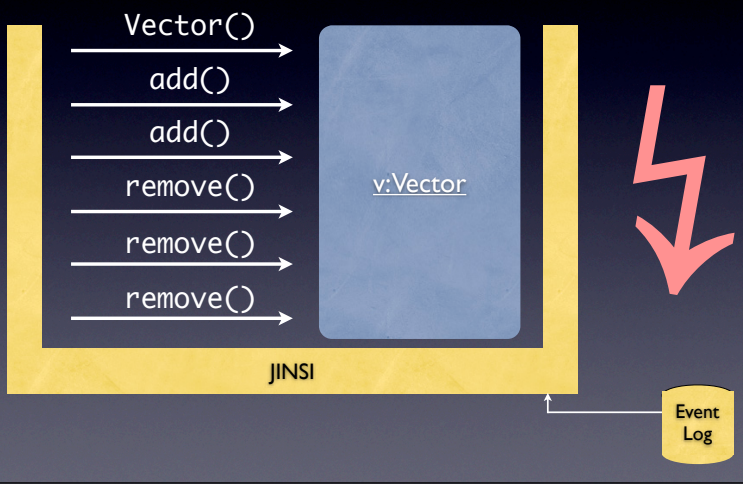
Schritt 1: Aufzeichnen



28

Isolieren Relevanter Aufrufe

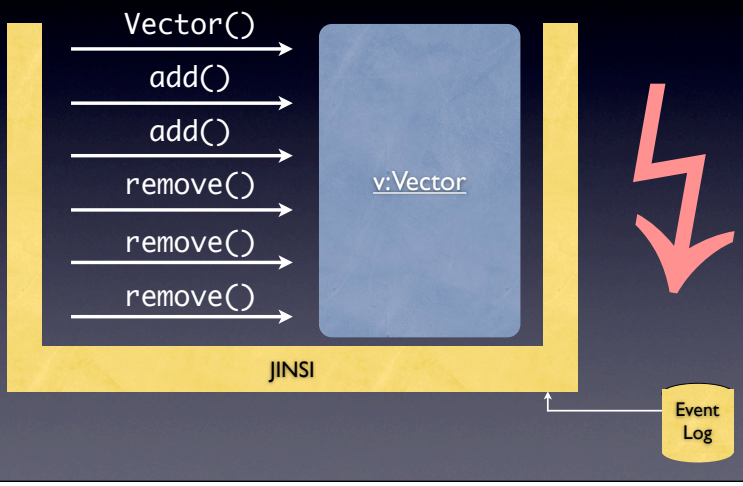
Schritt 2: Abspielen



29

Isolieren Relevanter Aufrufe

Schritt 3: Vereinfachen



30

Isolieren Relevanter Aufrufe

Schritt 4: Testfall erzeugen

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

JINSI

Columba ContactModel

ContactModel()
setSortString()
setFormattedName()
setNickName()
setFamilyName()
setGivenName()

und 18732 weitere…

c: ContactModel

Columba ContactModel

ContactModel()
getPreferredEmail()

c: ContactModel

Unit Test

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


getPreferredEmail

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

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

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

   return model.getAddress();
}
```


Delta Debugging

Delta Debugging *isoliert automatisch Fehlerursachen:*

Eingaben: 1 von 436 Kontakten in Columba

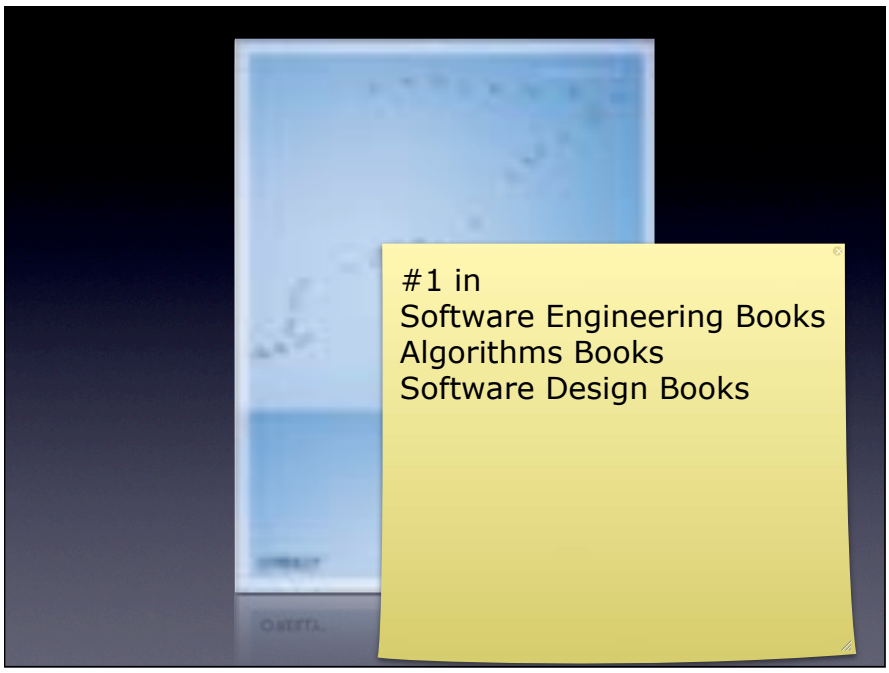
Änderungen: 1 von 8,721 Änderungen in CVDB

Threads: 1 von 3.8 bio Thread-Wech… …ava

Aufrufe: 2 von 18738 Methodenau…

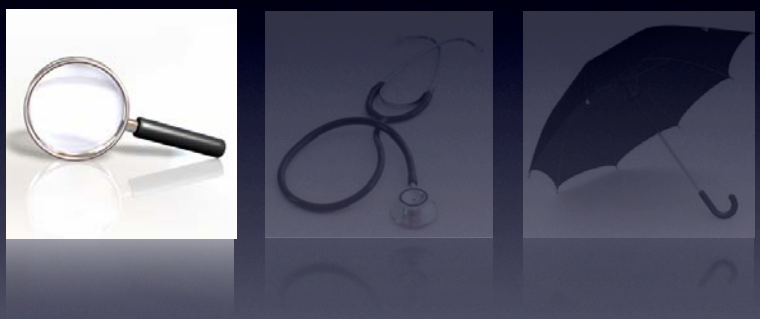
➔ *Automatisierte Fehlersuche*

99,8%
Verringerung des Suchraums



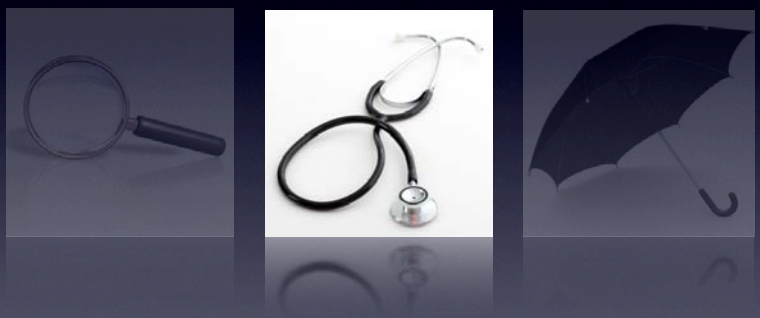
37

Diagnose Erkennen Vorbeugen



38

Diagnose Erkennen Vorbeugen



39

Kann man spezifizieren – eleganter ist aber das Extrahieren aus Code

weaveAtFieldRepeatedly

```
for (Iterator iter = itdFields.iterator();
     iter.hasNext();) {
    ...
    for (Iterator iter2 = worthRetrying.iterator();
         iter.hasNext();) {
        ...
    }
}
```

AspectJ

getPreferredEmail

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

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

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

  return model.getAddress();
}
```

Columba

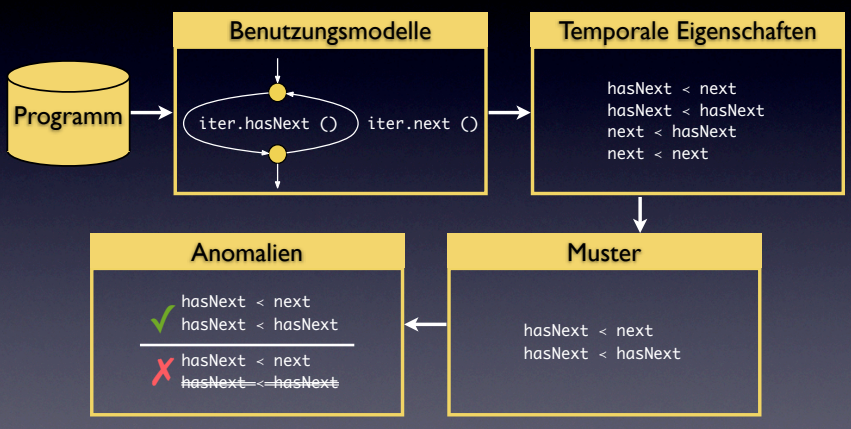
- Ungültige Benutzung von Iteratoren: vor next() sollte hasNext() kommen
- hasNext() ist *operationale Vorbedingung*

40

Problem:
Vorbedingungen spezifizieren ist aufwändig!

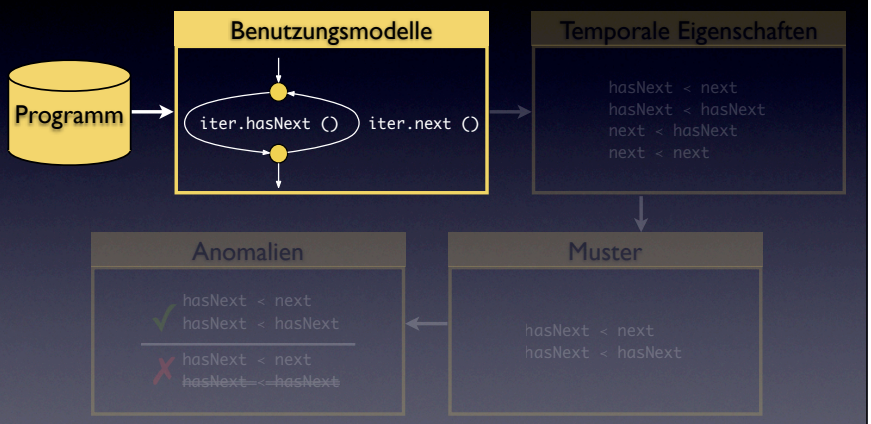
41

OP-Miner



42

OP-Miner



Methodenmodelle

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```

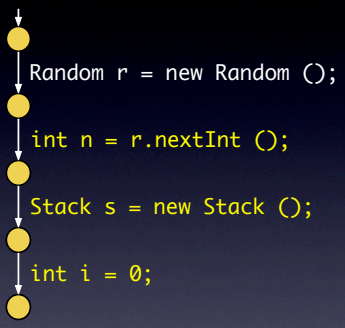

Methodenmodelle

Random r = new Random ();

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```

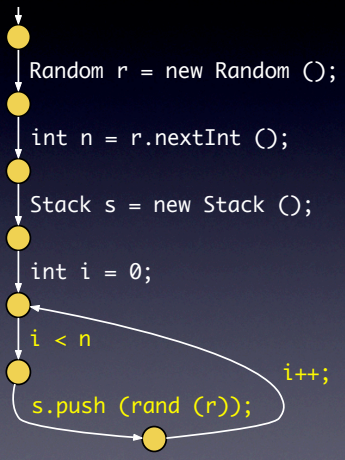
Methodenmodelle

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```



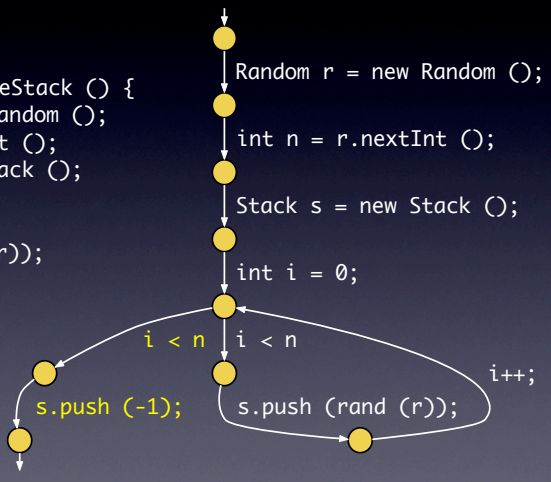
Methodenmodelle

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```



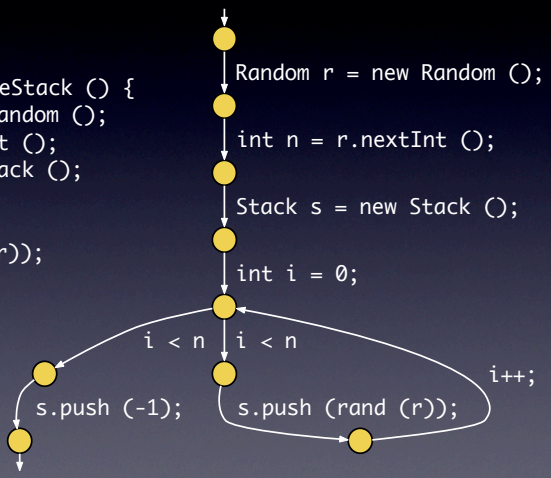
Methodenmodelle

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```

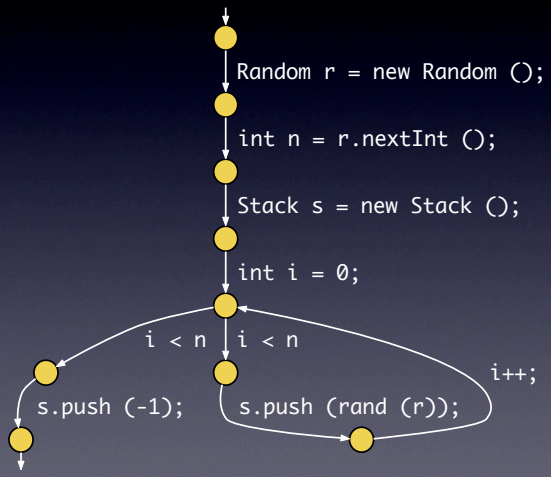


Methodenmodelle

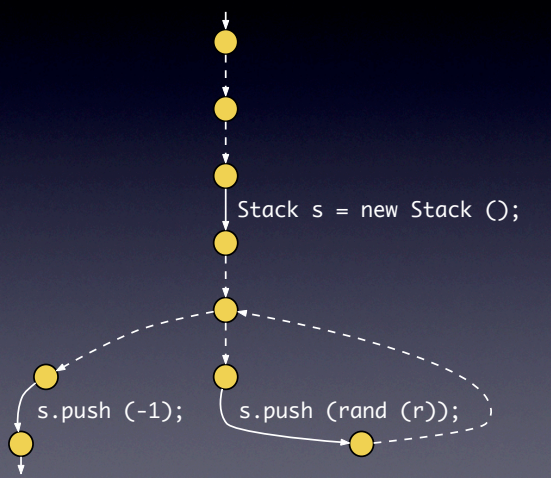
```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```



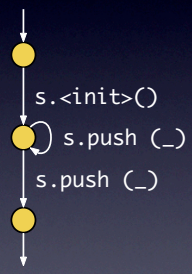
Benutzungsmodelle



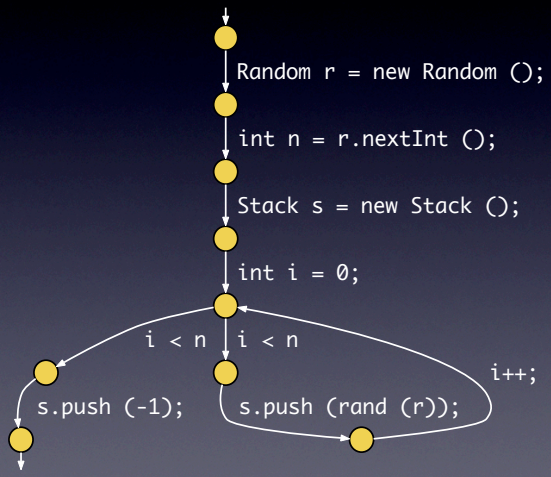
Benutzungsmodelle



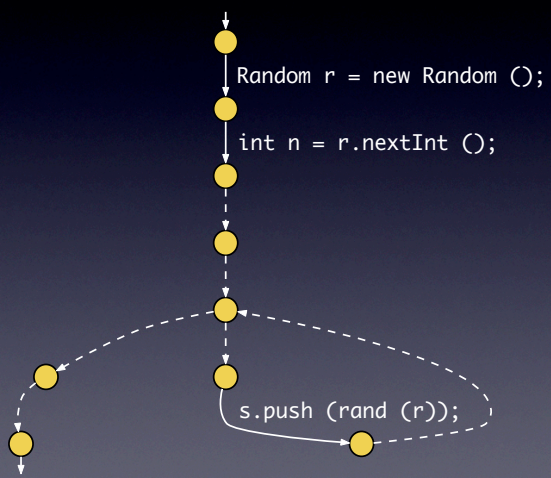
Benutzungsmodelle



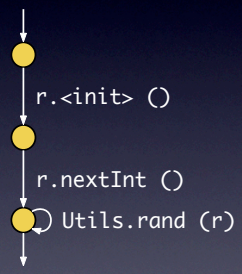
Benutzungsmodelle



Benutzungsmodelle

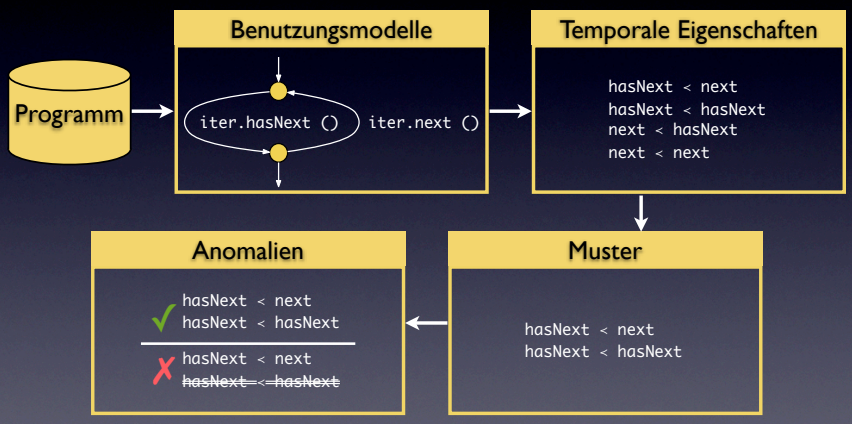


Benutzungsmodelle



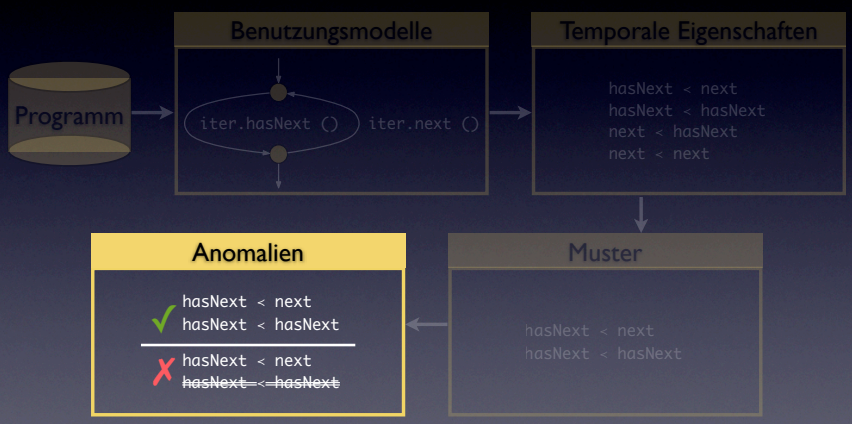
55

OP-Miner



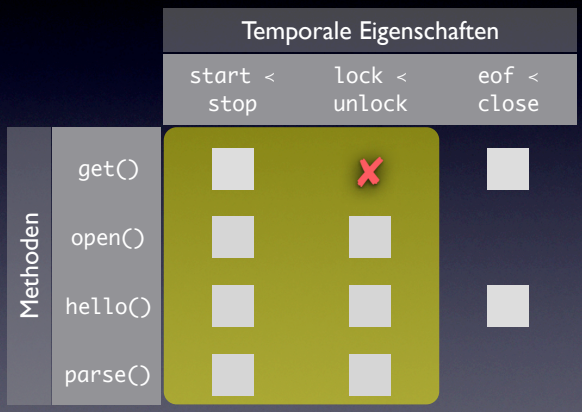
56

OP-Miner



57

Anomalien entdecken



58

This would be a pattern, if it were not for the missing element

Ein Defekt

```
for (Iterator iter = itdFields.iterator();
     iter.hasNext();) {
    ...
    for (Iterator iter2 = worthRetrying.iterator();
         iter.hasNext();) {
        ... sollte iter2 sein
    }
}
```

bug.aj

```
@interface A {}

aspect Test {
  declare @field : @A int var* : @A;
  declare @field : int var* : @A;

  interface Subject {}

  public int Subject.vara;
  public int Subject.varb;
}

class X implements Test.Subject {}
```

59

Noch ein Defekt

```
public void visitNEWARRAY (NEWARRAY o) {
   byte t = o.getTypecode ();
   if (!((t == Constants.T_BOOLEAN) ||
         (t == Constants.T_CHAR) ||
          ...
         (t == Constants.T_LONG))) {
      constraintViolated (o, "(...)'+t+'(...)");
   }
}
```

sollte "+t+" sein

60

On encountering a wrong typecode, \<visitNEWARRAY()> should report the typecode to the user. However, it fails to do so, as it uses \<'+t+'> instead of \<"+t+"> when constructing the second parameter to the \<constraintViolated()> method, causing the string \<'+t+'> to be interpreted verbatim---the message contains \<'+t+'> rather than the typecode in \<t>. \OPMiner{} reports this as an OP violation: the second parameter of \<constraintViolated()> should be the result of a \<StringBuffer.toString()> method call---i.e.\ a constructed string rather than a constant string. The

Ein Fehlalarm

```
Name internalNewName (String[] identifiers)
  ...
  for (int i = 1; i < count; i++) {
    SimpleName name = new SimpleName(this);
    name.internalSetIdentifier(identifiers[i]);
    ...
  }
  ...
}
```

sollte unverändert bleiben

In 48 cases: argument comes from String() constructor;
only in 3 cases: from array

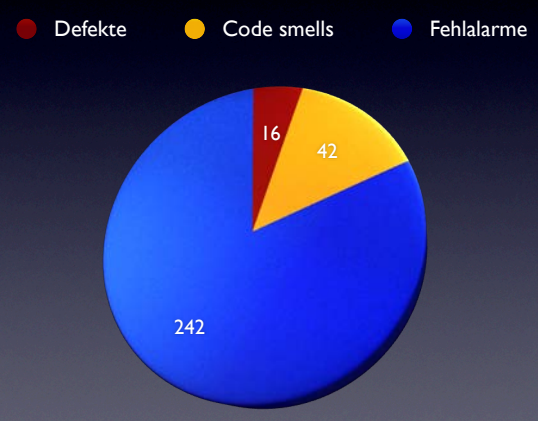
Ein Code Smell

```
public String getRetentionPolicy ()
{
  ...
  for (Iterator it = ...; it.hasNext();)
  {
      ... = it.next();
      ...
      return retentionPolicy;
  }
  ...
}
```

sollte repariert werden

Hint → if fixed, would improve program
Code smell → does not result in errors, but may cause maintainability problems
Defects → reported & verified

AspectJ



1 out of 5 is a defect or code smell
2.5 minutes per violation – one new defect after 10 minutes
Defects → reported & confirmed

Fehler finden

Program	# Violations Total	Investigated	# Defects	# Code smells	# False positives	Efficiency
Act-Rbot 0.8.2	25	25	2	13	10	**60%**
Apache Tomcat 6.0.16	55	55	0	9	46	**16%**
ArgoUML 0.24	305	28	0	12	16	**43%**
AspectJ 1.5.3	300	300	16	42	242	**19%**
Azureus 2.5.0.0	315	85	1	26	58	**32%**
Columba 1.2	57	57	4	15	38	**33%**
jEdit 4.2	11	11	0	4	7	**36%**
	1,068	**562**	**23**	**121**	**417**	**26%**

All in all, 1 out of 4 violations is a problem
Lots of subtle defects in production code
Unclear whether these would be found by other means

64

OP-Miner

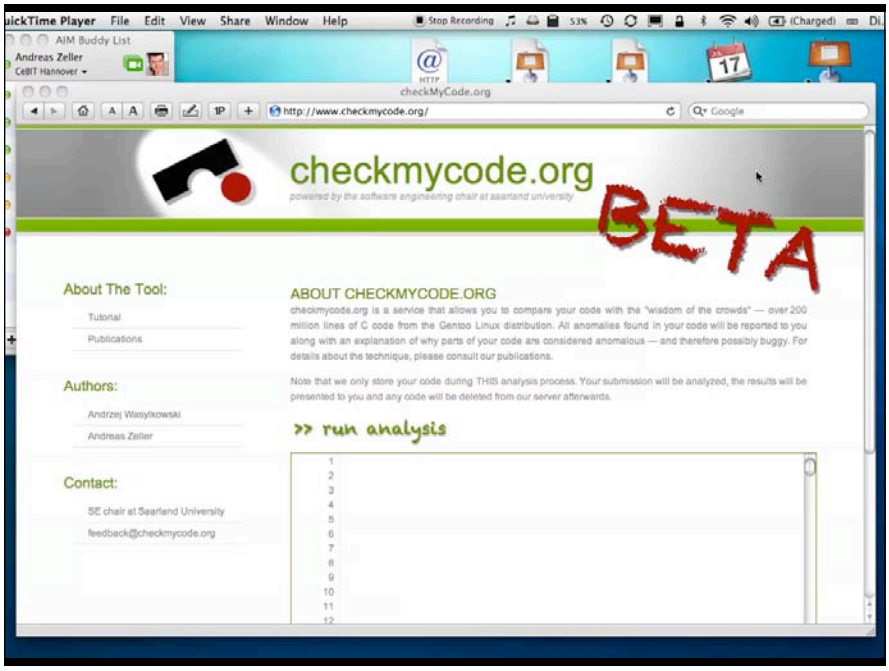
- ★ OP-Miner lernt *operationale Vorbedingungen* – wie man Argumente gewöhnlich konstruiert
- ★ Lernt aus gewöhnlicher Benutzung – allgemein und projektspezifisch
- ★ Vollautomatisch
- ★ Dutzende bestätigter Fehler gefunden

65

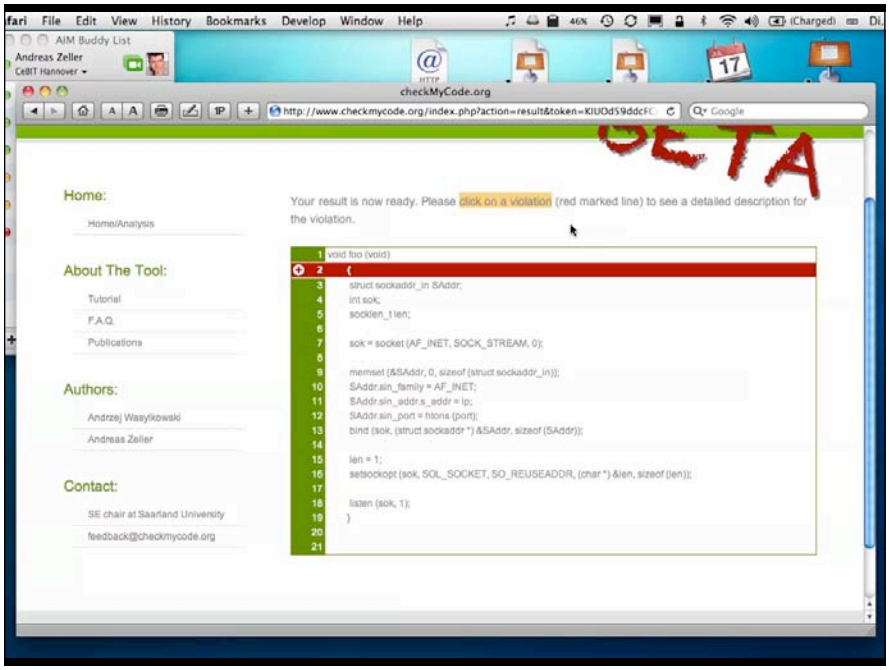
We have 6097 projects in our reference database. Their size ranges from 7 (for openssl-blacklist_0.4.2 and openvpn-blacklist_0.3) to 5,491,951 (for linux-2.6.29) SLOC (generated using David A. Wheeler's 'SLOCCount'; includes only .c files). Some other statistics:

[first quartile]:	1093
[third quartile]:	16160
[median]:	4162
[mean]:	33020

66



67



68

Ein Defekt

conspire-0.20 (IRC client)

```
static int dcc_listen_init (…) {
  dcc->sok = socket (…);
  if (…) {
    while (…) {
      … = bind (dcc->sok, …);
    }
    /* with a small port range, reuseAddr is needed */
    setsockopt (dcc->sok, …, SO_REUSEADDR, …);
  }
  listen (dcc->sok, …);
}
```

muss vor bind() stehen

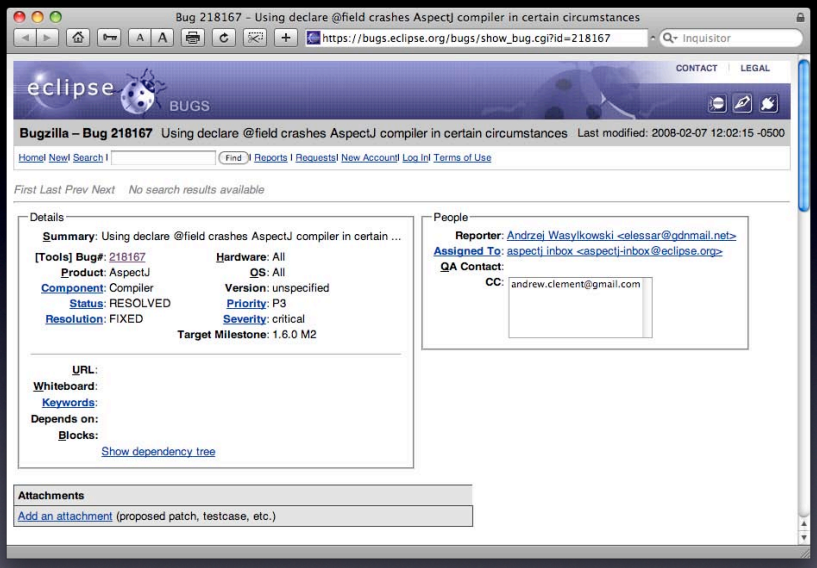
69

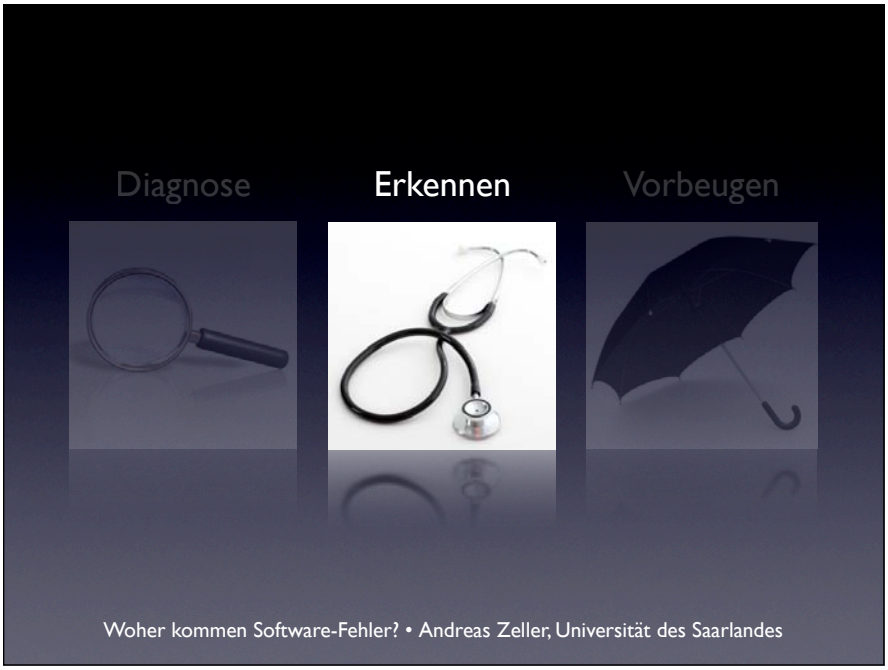
This comes from the conspire-0.20 project. Just the relevant code is shown here.

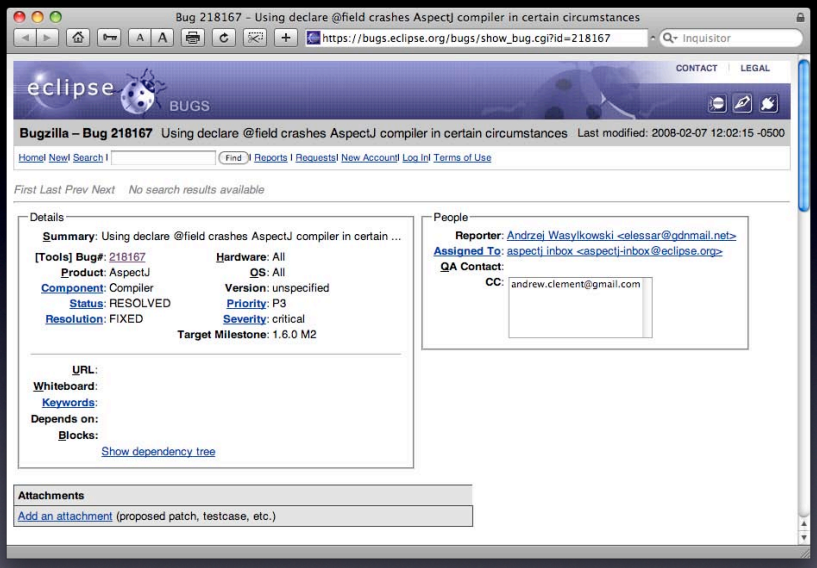
The defect: setsockopt with SO_REUSEADDR option must be called BEFORE bind to have any effect. Here it's just wrong.

Why is this a good example? There are six

Programmfehler sind eingereicht und bestätigt



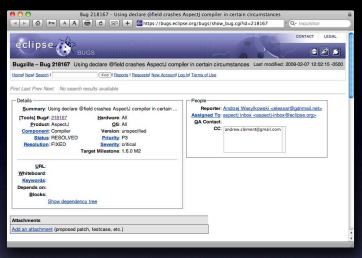




73

Problem:
Wie können wir aus unseren Fehlern lernen?

74

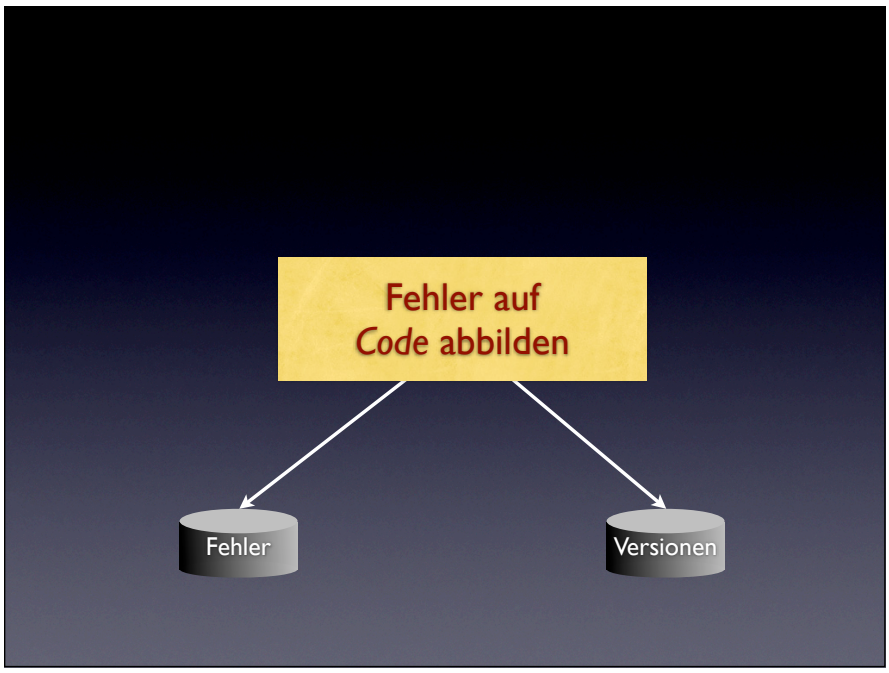


Fehler

Versionen

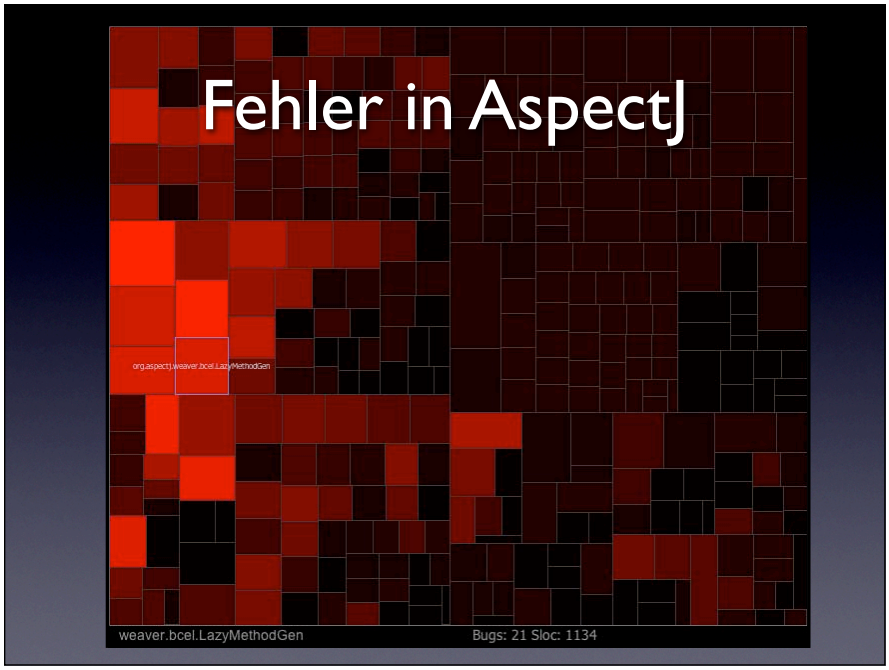
Such software archives are being used in practice all the time. If you file a bug, for instance, the report is stored in a bug database, and the resulting fix is stored in the version archive.

75



These databases can then be mined to extract interesting information. From bugs and changes, for instance, we can tell how many bugs were fixed in a particular location.

76



77

78

Sind es die Entwickler?

Macht Erfahrung einen Unterschied?

Je mehr Erfahrung, desto mehr Fehler!

79

Oder die Geschichte?

Wir haben hier viele Fehler gefunden…

Dann sind dort noch mehr!

80

Wie steht es mit Metriken?

Korrelieren Metriken mit Fehlerdichte?

Manchmal!

81

Programmiersprache?

Sind *gotos* schädlich?

Keine Korrelation!

82

Ok. Problembereich?

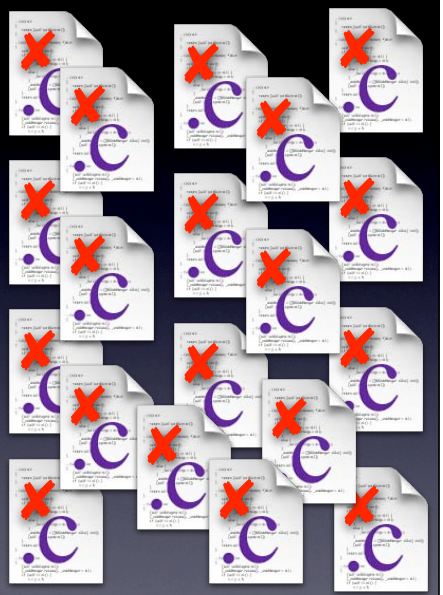
Welche Tokens sind relevant?

import • extends • implements

83

nsIPrivateDOMEvent.h

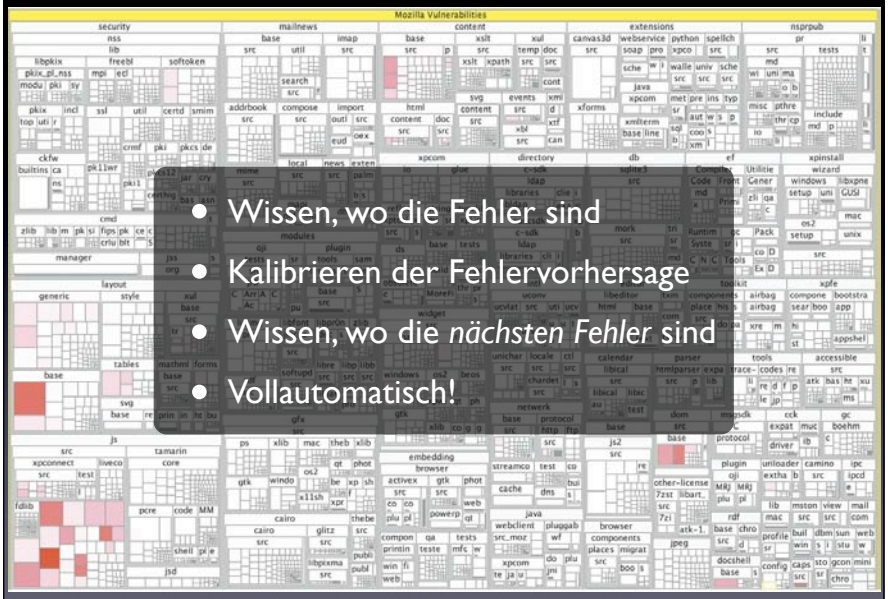
nsReadableUtils.h



84

Vorhersage	Komponente	Tatsache
1	nsDOMClassInfo	3
2	SGridRowLayout	95
3	xpcprivate	6
4	jsxml	2
5	nsGenericHTMLElement	8
6	jsgc	3
7	nsISEnvironment	12
8	jsfun	1
9	nsHTMLLabelElement	18
10	nsHttpTransaction	35

85



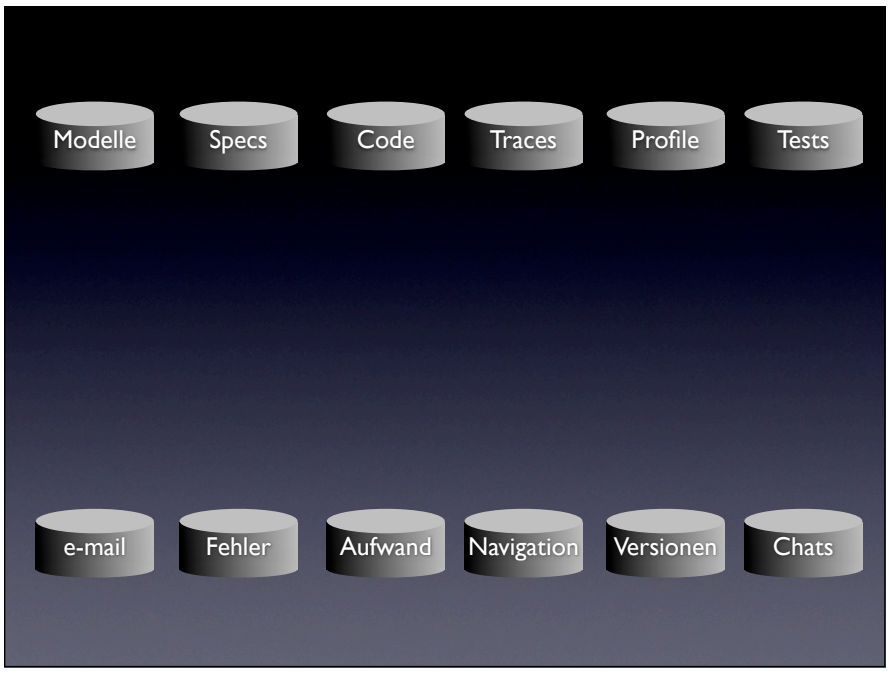
86

Software-Archive

- enthalten komplette Projektgeschichte
- gepflegt durch Programmierumgebungen
- automatischer Zugang
- frei verfügbar durch Open-Source Projekte

This was just a simple example. So, the most important aspect that software archives give you is automation. They are maintained automatically ("The data comes to you"), and they can be evaluated automatically ("Instantaneous results"). For researchers, there are plenty open source archives available, allowing us to test, compare, and evaluate our tools.

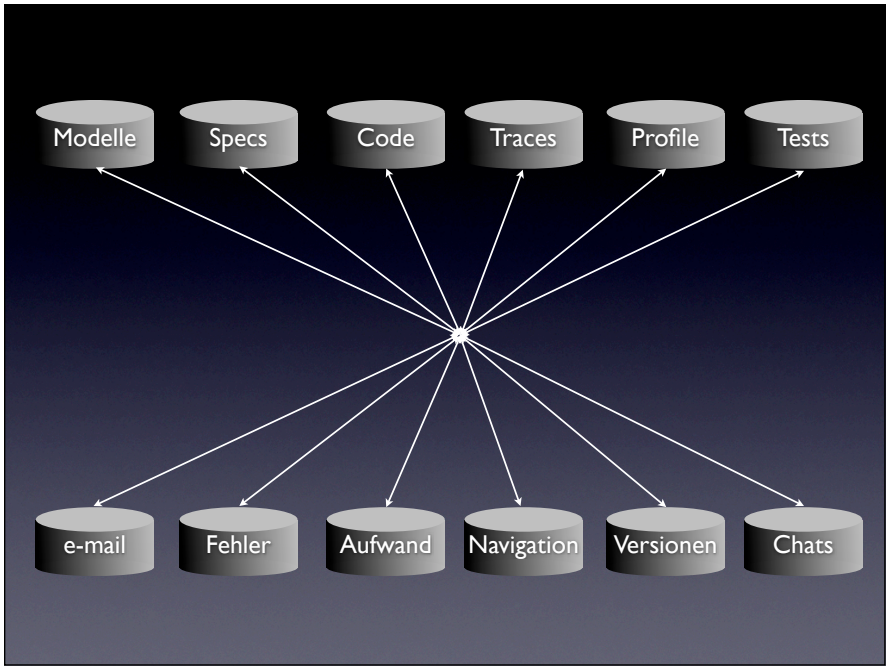
87



Tools can only work together if they draw on different artefacts

What are we working on in SE - we are constantly producing and analyzing artefacts: code, specs, etc.

88



Combining these sources will allow us to get this "waterfall effect" – that is, being submerged by data; having more data than we could possibly digest.

89

The dirty story about this data is that it is frequently collected manually. In fact, the company phone book is among the most important tools of an empirical software engineering researchers. One would phone one developer after the other, and question them – say, "what was your effort", or "how often did you test module 'foo'?", and tick in the appropriate form. In other words, data is scarce, and as it is being collected from humans after the fact, is prone to errors, and prone to bias.

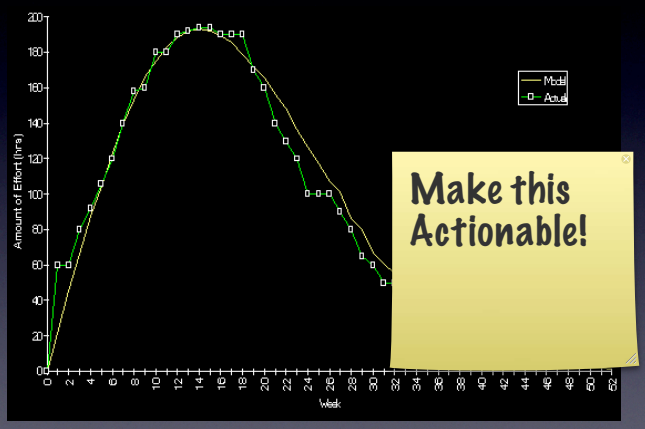
90

91

Combining these sources will allow us to get this "waterfall effect" – that is, ~~being submerged by data; having more data than we could possibly digest.~~

92

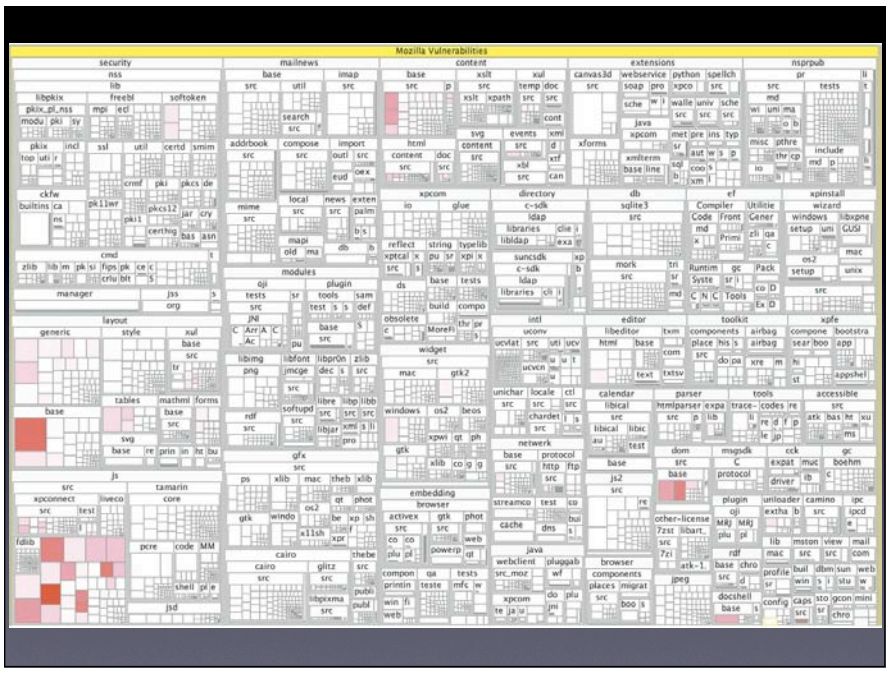
Studien



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

Let's now talk about results. What should our tools do? Should they come up with nice reports, and curves like ~~this one?~~

93


Mozilla Vulnerabilities

94


Church-St
240
yards
0.30 M
33 mph
10:13 a
10:15 a
Broadway
tomtom

95


Tom Mens
Serge Demeyer (Eds.)
Software Evolution
Springer
Windows Server System
DIRECTX 9
Windows NetMeeting
eclipse
aspectj

96

Google™ IBM® SAP® Microsoft®

97

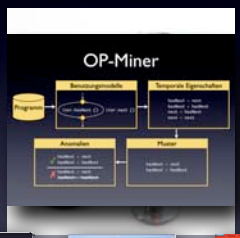
Diagnose Erkennen Vorbeugen

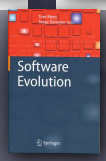


Woher kommen Software-Fehler? • Andreas Zeller, Universität des Saarlandes

98

Diagnose Erkennen Vorbeugen



Woher kommen Software-Fehler? • Andreas Zeller, Universität des Saarlandes

99