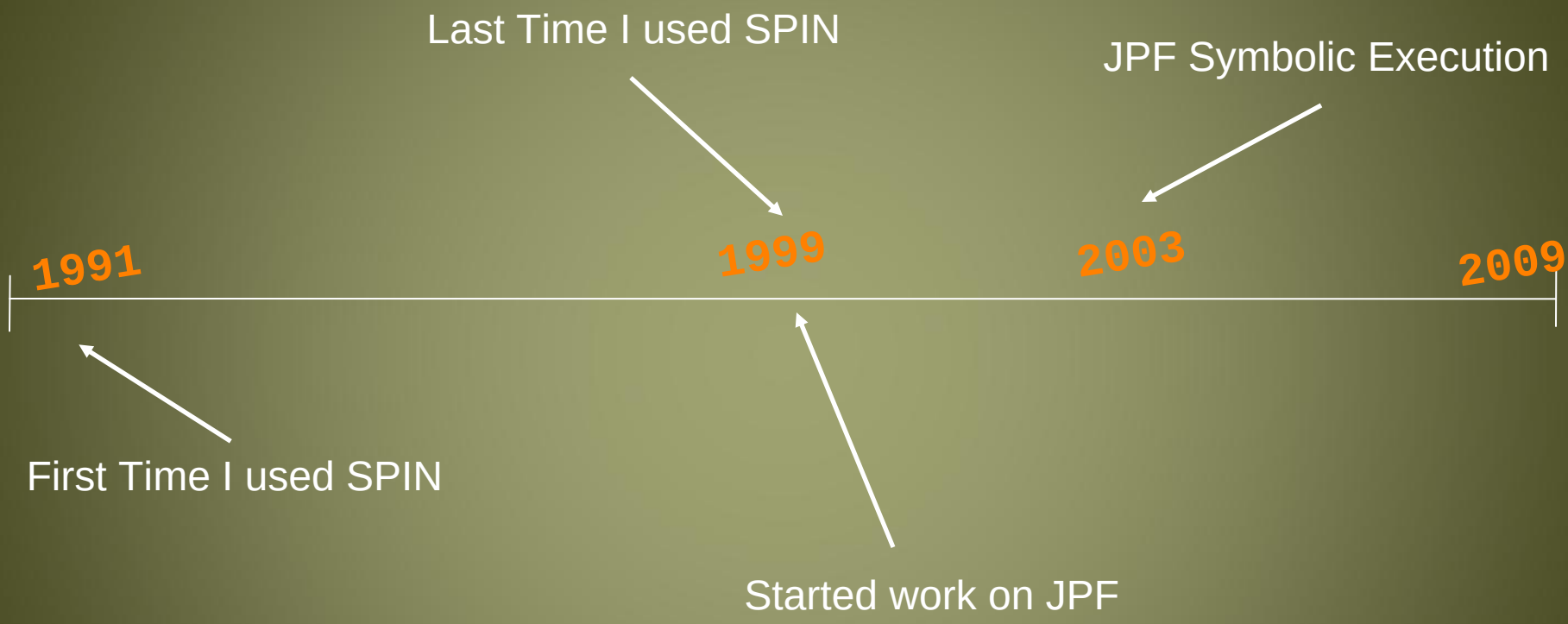
Who Really Cares if the Program Crashes?

Willem Visser

Stellenbosch University

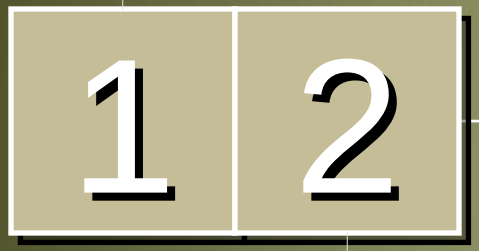

Last Time I used SPIN
JPF Symbolic Execution
1991
1999
2003
2009
First Time I used SPIN
Started work on JPF

Introduction

1

End

Introduction



End

A Brief History of Everything


Introduction
What I’ve learned away from research
1
2
3
End
A Brief History of Everything


Introduction
What I’ve learned away from research
1
2
3
4
A Brief History of Everything
Conclusions


Introduction
SEVEN (2007-2009)
1
2
3
4
NASA (1998 – 2007)
STELLENBOSCH
2009 - …

How many

previously unknown bugs

have you found

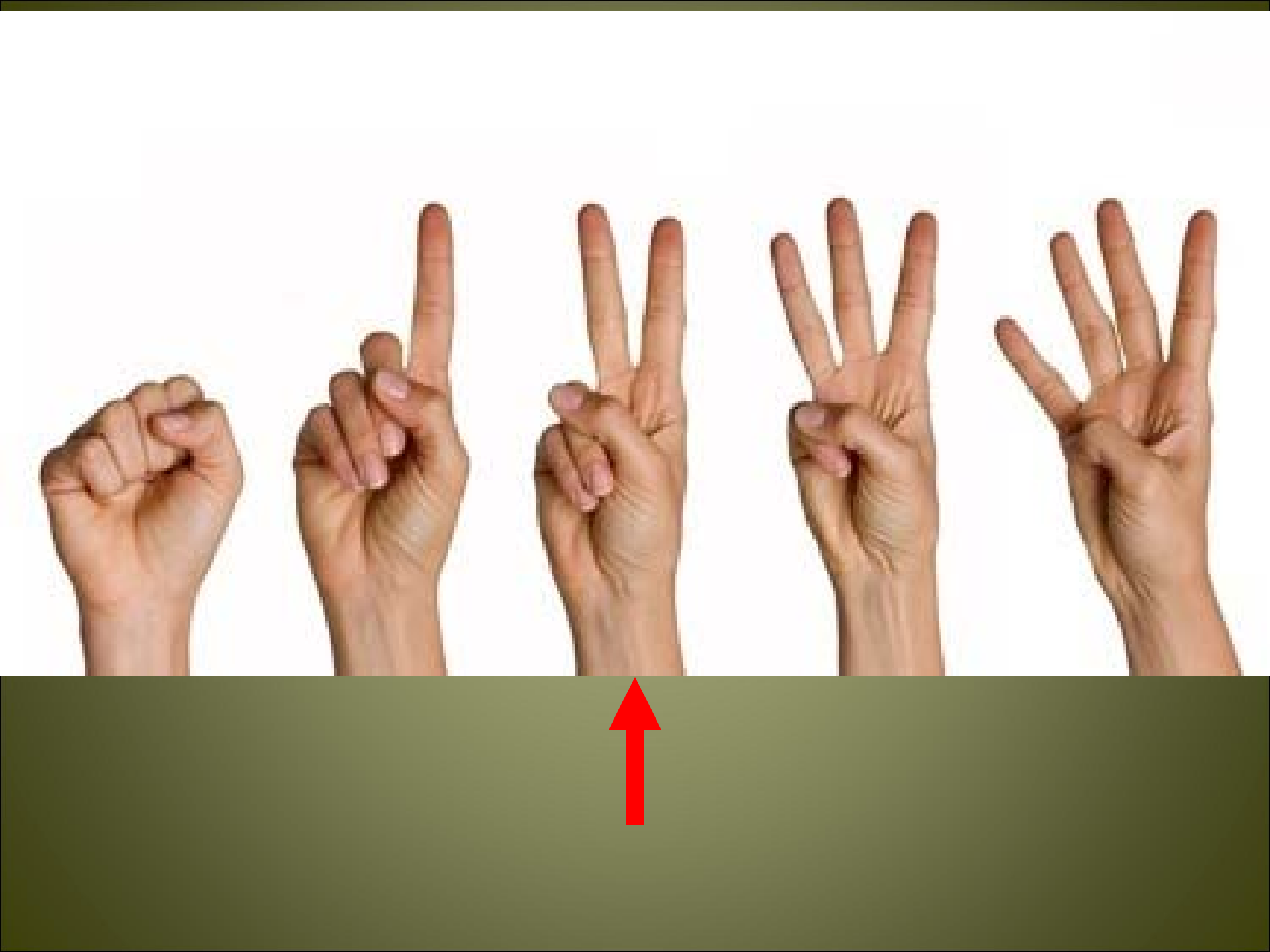
with a model checker?

How many

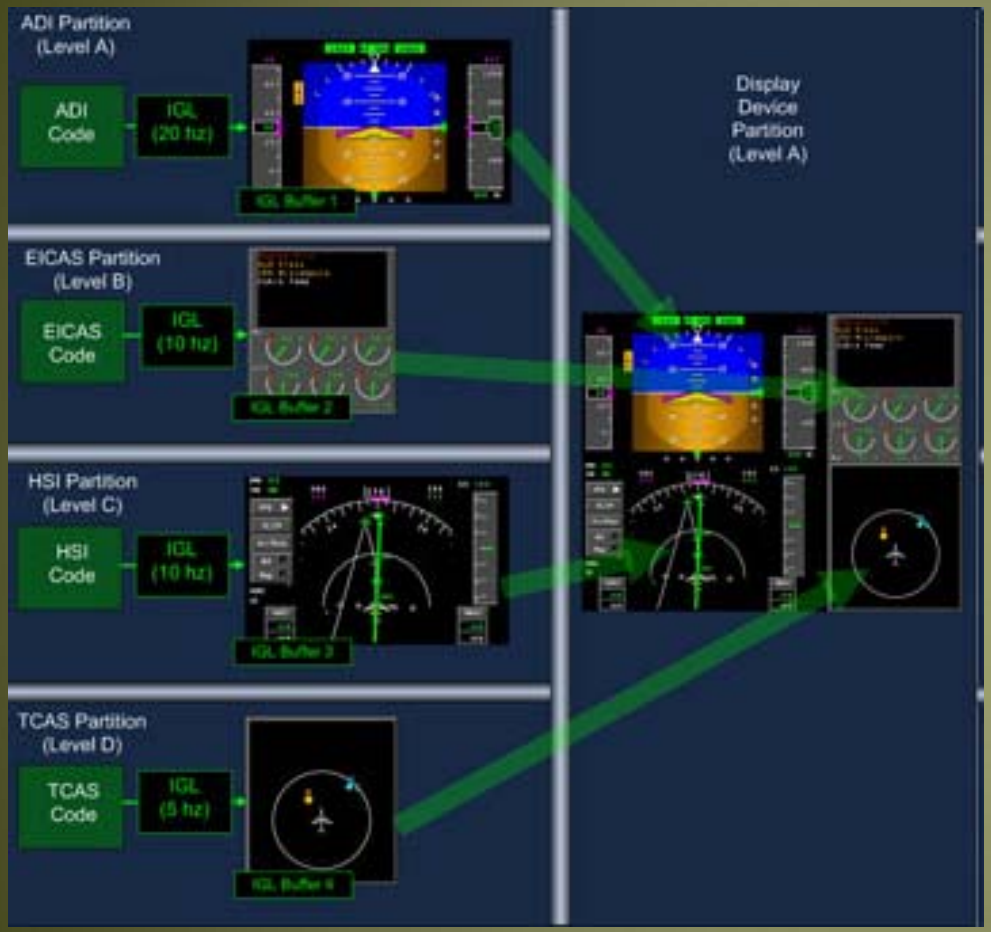
previously unknown bugs

have you found

with a model checker?



Integrated Modular Avionics



Honeywell to NASA:

"… we have a problem in the time partitioning, do you think a model checker can find it?"

DEOS OS on the Primus EPIC

Ok, it was not quite that easy

10,000 lines of code

C++ to PROMELA Translation took
2-3 weeks

Environment Generation took
2 months

Ah we have the bug…environment problem

Ah we have the bug…environment problem

Ah we have the bug…environment problem

Ah we have the bug…no environment problem

Honeywell engineers could not believe we found the bug with a model checker

Given enough resources
model checking can be
successfully applied to real programs

Creating the “correct” environment
is the hard problem
(This is also a problem for testing)

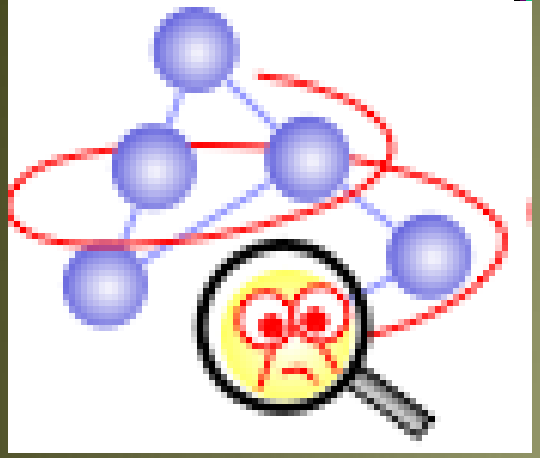
Hand translations from code to … does not scale

and

Automated translations don’t work
if the target language is not expressive enough

Java™ PathFinder

.. the (swiss army knife | vulcan mind probe | spiral-of-death | <what do you want it to be today?>) of Java program verification



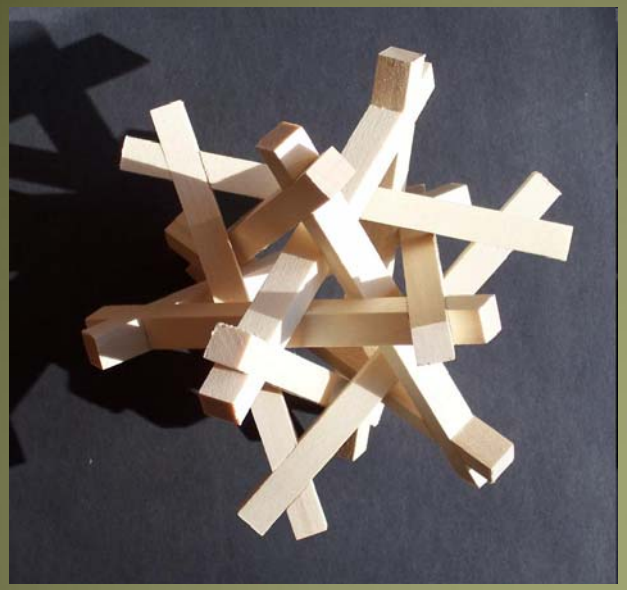
What PROMELA is to SPIN, JAVA is to JPF

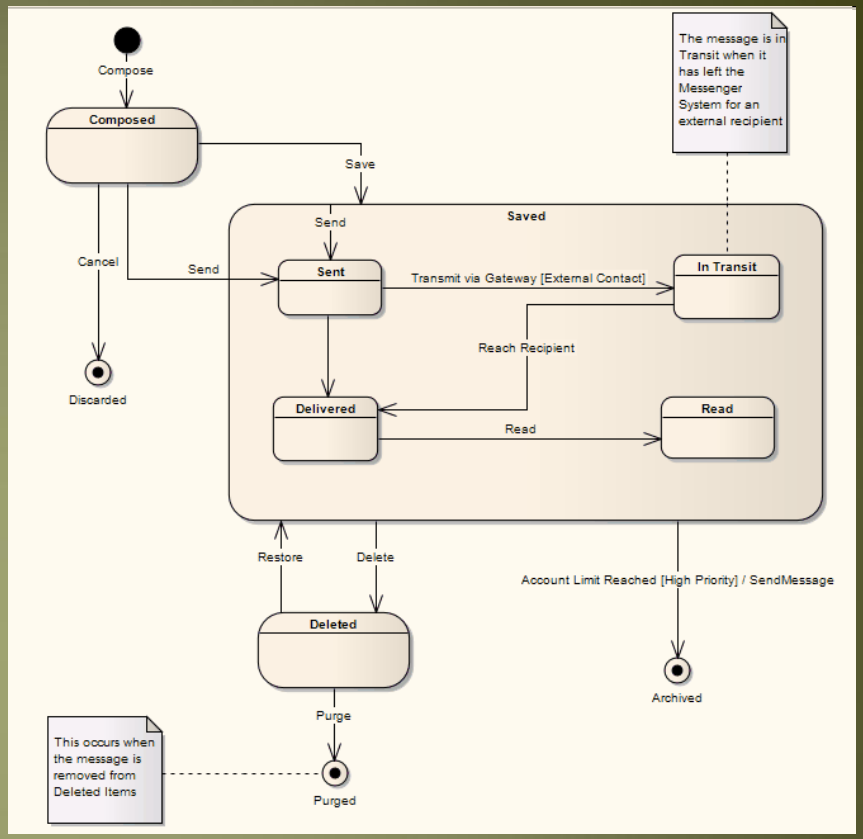
…java MyClass

…jpf MyClass

- Custom Java Virtual Machine executes Java class files
- Open Source – first NASA project to be developed exclusively in open source
- Supports all the regular bells and whistles: po reduction, symmetry reduction, heuristics search, user-defined non-determinism etc.
- **Highly extensible!**

Extensions

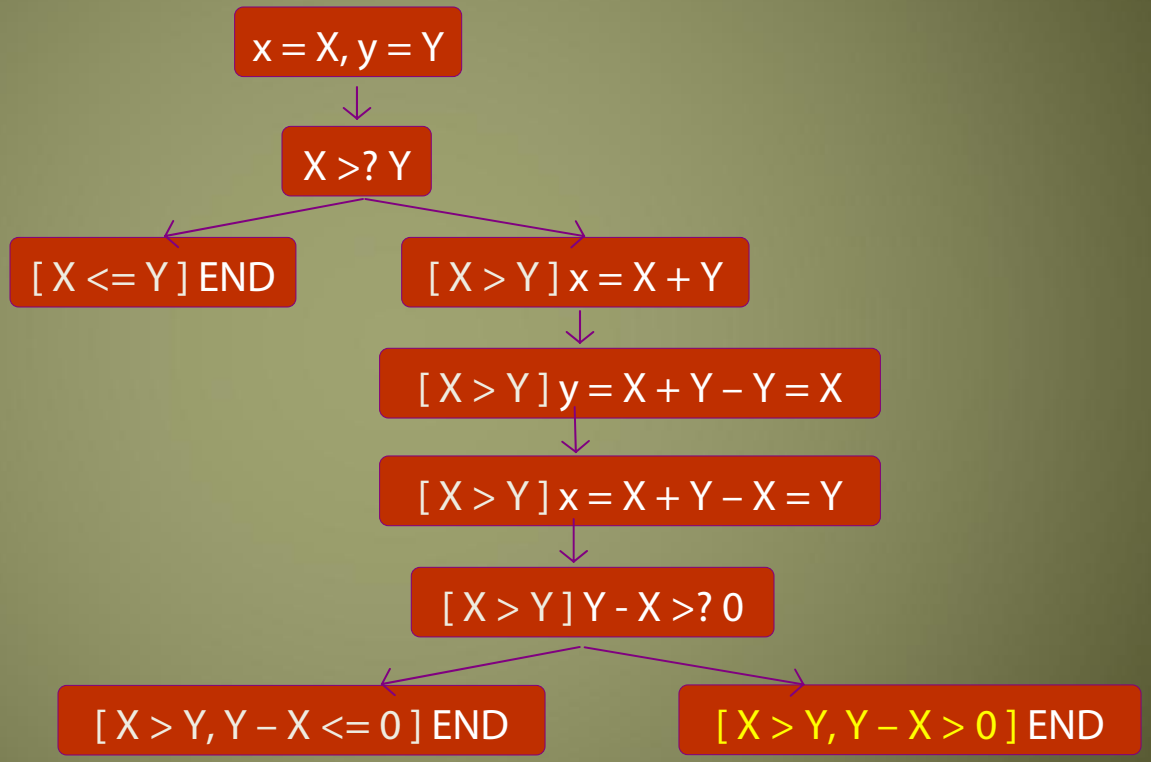




Compositional Verification, Statecharts, Symbolic Execution, Concolic Testing, Advanced Coverage, UI analysis,…

Symbolic Execution Tree (example)

```
int x, y;

if (x > y) {

   x = x + y;

   y = x – y;

   x = x – y;

   if (x – y > 0)

      //Reachable?

}
```



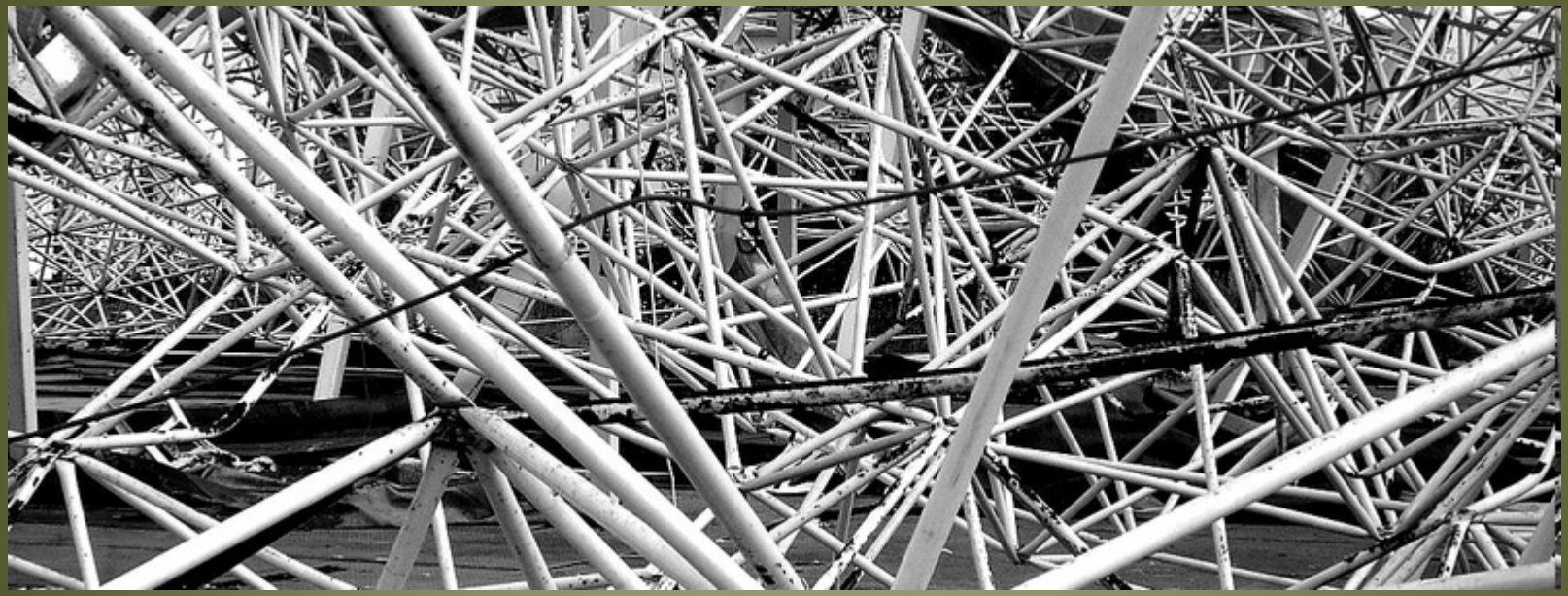
What makes model checking specia

What makes model checking specia

Finding the unintended consequences due to complex interactions between components

What makes symbolic execution spec

What makes symbolic execution spec



Allows one to see the complexity of the
in even a sequential program

One of the basic blocks in the Binomial Heap implementation required a minimum sequence of 13 API calls to be covered

```
private void merge(BinomialHeapNode binHeap) {
  BinomialHeapNode temp1 = Nodes, temp2 = binHeap;
  while ((temp1 != null) && (temp2 != null)) {
    if (temp1.degree == temp2.degree) {
      BinomialHeapNode tmp = temp2;
      temp2 = temp2.sibling;
      tmp.sibling = temp1.sibling;
      temp1.sibling = tmp;
      temp1 = tmp.sibling;
    } else {
     if (temp1.degree < temp2.degree) {
       if ((temp1.sibling == null) ||
           (temp1.sibling.degree > temp2.degree)) {
         // HERE!
         …
```

```
X4(1) >= X8(1) && X10(2) > X8(1) &&
X10(2) <= X11(2) && X11(2) > 0 &&
X10(2) > 0 && X8(1) <= X9(1) &&
X9(1) > 0 && X8(1) > 0 && X4(1) <= X2(1) &&
X6(2) > X4(1) && X6(2) <= X7(2) &&
X7(2) > 0 && X6(2) > 0 && X4(1) <= X5(1) &&
X5(1) > 0 && X4(1) > 0 && X2(1) <= X0(1) &&
X2(1) <= X3(1) && X3(1) > 0 && X2(1) > 0 &&
X0(1) <= X1(1) && X1(1) > 0 && X0(1) > 0

insert(X0);insert(X1);insert(X2);insert(X3);insert(X4);
insert(X5);insert(X6);insert(X7);insert(X8);insert(X9);
insert(X10);insert(X11);extractMin();
```

Orion Onboard Abort Executive

- 27 flight rules for when one of 7 aborts should occur
- JPF within 1 min finds 23 tests that cover all the flight rules and all 7 of the aborts
- BUG: abort should happen, but none apply, under the following constraint…

inputs_fr.navIN_fr.geod_alt(300000) >= 300000 && inputs_fr.navIN_fr.geod_alt(300000) >= 120000 && inputs_fr.navIN_fr.geod_alt(300000) >= 38000 && inputs_fr.navIN_fr.geod_alt(300000) >= 10000 && inputs_fr.sysIN_fr.cev_cm_cabin_pres_rate(-1) >= -1 && inputs_fr.sysIN_fr.cev_cm_cabin_pres_rate(-1) >= -2 && inputs_fr.navIN_fr.las_jettison_cmd(-1000000) != 1 && inputs_fr.navIN_fr.roll_rate(40) <= 50 && inputs_fr.navIN_fr.yaw_rate(31) <= 41 && inputs_fr.navIN_fr.pitch_rate(70) <= 100 && inputs_fr.navIN_fr.las_jettison_cmd(-1000000) != 0 && inputs_fr.sysIN_fr.stage2_apu_volt(22) <= 33 && inputs_fr.sysIN_fr.stage2_apu_volt(22) >= 22 && inputs_fr.navIN_fr.vmissmag(5) <= 10 && inputs_fr.sysIN_fr.stage2_thrust(221922) <= 332883 && inputs_fr.sysIN_fr.stage2_thrust(221922) >= 221922 && inputs_fr.sysIN_fr.stage2_helium_tnk_pres(560) <= 840 && inputs_fr.sysIN_fr.stage2_helium_tnk_pres(560) >= 560 && inputs_fr.sysIN_fr.stage2_lh2_tnk_pres(26) <= 40 && inputs_fr.sysIN_fr.stage2_lh2_tnk_pres(26) >= 26 && inputs_fr.sysIN_fr.stage2_lox_tnk_pres(17) <= 25 && inputs_fr.sysIN_fr.stage2_lox_tnk_pres(17) >= 17 && inputs_fr.sysIN_fr.stage2_hpft_speed(28288) <= 42432 && inputs_fr.sysIN_fr.stage2_hpft_speed(28288) >= 28288 && inputs_fr.sysIN_fr.stage2_lpft_speed(12948) <= 19422 && inputs_fr.sysIN_fr.stage2_lpft_speed(12948) >= 12948 && inputs_fr.sysIN_fr.stage2_hpot_speed(22496) <= 33744 && inputs_fr.sysIN_fr.stage2_hpot_speed(22496) >= 22496 && inputs_fr.sysIN_fr.stage2_lpot_speed(4120) <= 6180 && inputs_fr.sysIN_fr.stage2_lpot_speed(4120) >= 4120 && inputs_fr.sysIN_fr.stage2_efi_pres(4800) <= 7200 && inputs_fr.sysIN_fr.stage2_efi_pres(4800) >= 4800 && ((inputs_fr.sysIN_fr.stage1_tvc_actual(0) MINUS inputs_fr.sysIN_fr.stage1_tvc_commanded(0)) MULT 10) <= (inputs_fr.sysIN_fr.stage1_tvc_commanded(0) MULT 3) && inputs_fr.sysIN_fr.stage1_chmbr_pres(640) <= 960 && inputs_fr.sysIN_fr.stage1_chmbr_pres(640) >= 640 && theQueue.q[7].spred.outputs_p.vgo_Mag(115) <= 515 && theQueue.q[7].spred.outputs_p.vgo_Mag(115) >= 115 && theQueue.q[0].spred.outputs_p.vmissmag_eo_now(410) <= 610 && theQueue.q[0].spred.outputs_p.vmissmag_eo_now(410) >= 410 && inputs_fr.navIN_fr.yaw_rate(31) <= 51 && inputs_fr.navIN_fr.yaw_rate(31) >= 31 && inputs_fr.navIN_fr.yaw(100) <= 140 && inputs_fr.navIN_fr.yaw(100) >= 100 && inputs_fr.navIN_fr.roll_rate(40) <= 60 && inputs_fr.navIN_fr.roll_rate(40) >= 40 && inputs_fr.navIN_fr.roll(100) <= 140 && inputs_fr.navIN_fr.roll(100) >= 100 && inputs_fr.navIN_fr.pitch_rate(70) <= 130 && inputs_fr.navIN_fr.pitch_rate(70) >= 70 && inputs_fr.navIN_fr.pitch(45) <= 65 && inputs_fr.navIN_fr.pitch(45) >= 45 && inputs_fr.navIN_fr.vmissmag(5) <= 15 && inputs_fr.navIN_fr.vmissmag(5) >= 5 && inputs_fr.navIN_fr.inert_vel_mag(22000) <= 26000 && inputs_fr.navIN_fr.inert_vel_mag(22000) >= 22000 && inputs_fr.navIN_fr.geod_alt(300000) <= 310000 && inputs_fr.navIN_fr.geod_alt(300000) >= 0 && inputs_fr.sysIN_fr.stage2_thrust(221922) <= 342883 && inputs_fr.sysIN_fr.stage2_thrust(221922) >= 211922 && inputs_fr.sysIN_fr.stage2_lpot_speed(4120) <= 6680 && inputs_fr.sysIN_fr.stage2_lpot_speed(4120) >= 3620 && inputs_fr.sysIN_fr.stage2_lpft_speed(12948) <= 20422 && inputs_fr.sysIN_fr.stage2_lpft_speed(12948) >= 11948 && inputs_fr.sysIN_fr.stage2_lox_tnk_pres(17) <= 30 && inputs_fr.sysIN_fr.stage2_lox_tnk_pres(17) >= 12 && inputs_fr.sysIN_fr.stage2_lh2_tnk_pres(26) <= 45 && inputs_fr.sysIN_fr.stage2_lh2_tnk_pres(26) >= 21 && inputs_fr.sysIN_fr.stage2_hpot_speed(22496) <= 35744 && inputs_fr.sysIN_fr.stage2_hpot_speed(22496) >= 20496 && inputs_fr.sysIN_fr.stage2_hpft_speed(28288) <= 30288 && inputs_fr.sysIN_fr.stage2_hpft_speed(28288) >= 26288 && inputs_fr.sysIN_fr.stage2_helium_tnk_pres(560) <= 1040 && inputs_fr.sysIN_fr.stage2_helium_tnk_pres(560) >= 360 && inputs_fr.sysIN_fr.stage2_efi_pres(4800) <= 7700 && inputs_fr.sysIN_fr.stage2_efi_pres(4800) >= 4300 && inputs_fr.sysIN_fr.stage2_apu_volt(22) <= 48 && inputs_fr.sysIN_fr.stage2_apu_volt(22) >= 7 && inputs_fr.sysIN_fr.stage1_tvc_commanded(0) <= 10 && inputs_fr.sysIN_fr.stage1_tvc_commanded(0) >= 0 && inputs_fr.sysIN_fr.stage1_tvc_actual(0) <= 10 && inputs_fr.sysIN_fr.stage1_tvc_actual(0) >= 0 && inputs_fr.sysIN_fr.stage1_chmbr_pres(640) <= 1160 && inputs_fr.sysIN_fr.stage1_chmbr_pres(640) >= 440 && inputs_fr.sysIN_fr.cev_cm_cabin_pres_rate(-1) <= 0 && inputs_fr.sysIN_fr.cev_cm_cabin_pres_rate(-1) >= -3 &&

Rockets for the few

Email for everyone

NASA vs SEVEN

- Run once
- Huge cost to develop
- Any error can cause loss of mission

- No Java code
- Little concurrency

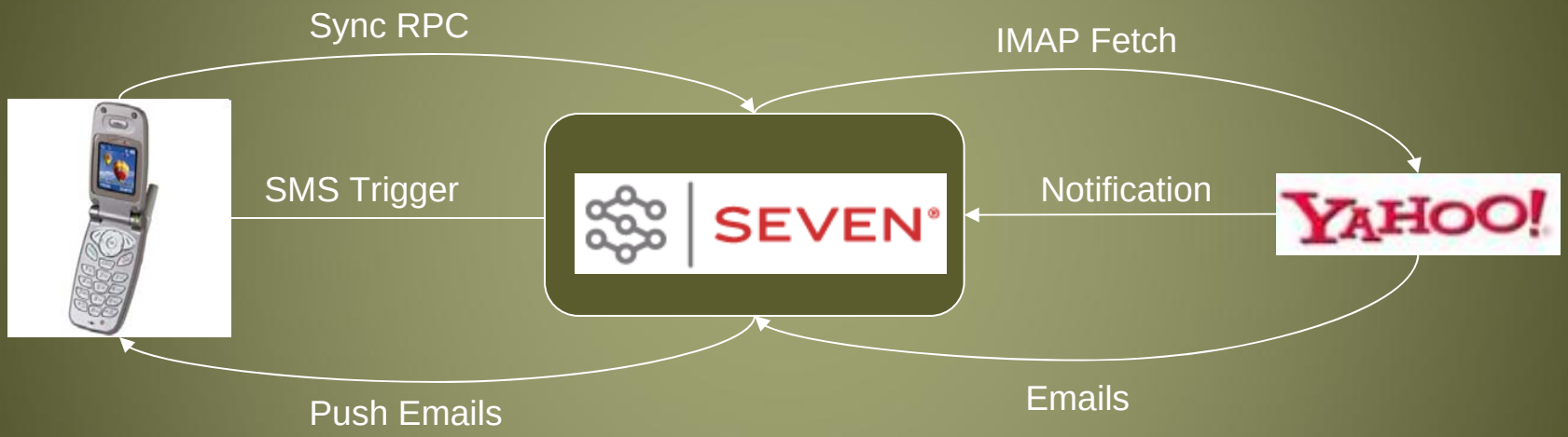
- Can afford
  - SE research tool development
  - Evaluations

- Run 1M times an hour
- Cost is in Maintenance
- Errors degrade user experience and revenue

- Only Java code
- Lots of concurrency

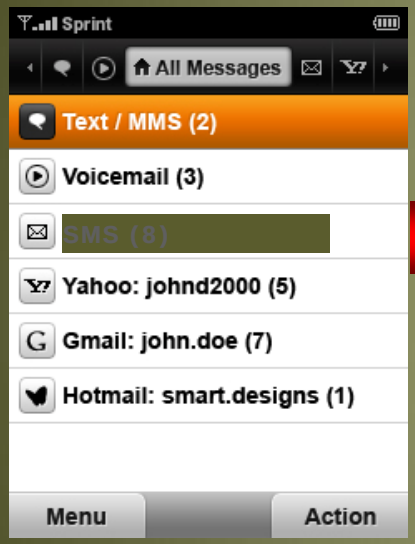
- Needs to use off-the-shelf tools

System 7



Server: huge thread pools where all threads do the same thing

SEVEN Clients



JAVA · Microsoft · Adobe · symbian · brew · palm · Linux

System SEVEN Server Platform

SEVEN

Desktop Connector

SEVEN Connectors

- Exchange/Notes
- YAHOO!
- Gmail talk BETA
- msn Hotmail
- AOL
- facebook
- myspace.com a place for friends..
- flickr
- Voicemail Platforms
- Network Address Books

Try it out at community.seven.com

Model Based Testing is the best thing since sliced bread!!

Model Based Testing can be too Expensive

The glue to code to make the generated tests run on the actual system, requires too much time and expertise

Concurrency bugs seldom happen,
but when they do they cause big problems

Concurrency bugs can happen all the time;
Deadlocks are no problem, Races are though

The very same concurrency infrastructure
gets executed thousands of times per second
which means even the unlikely errors become very likely

The tale of two concurrency bugs

Deadlock happened once

Race, happened all the time

Very easy to find from thread dump

Found it by hand

Model Checker found it *easily*

Model Checker could *never* find this

Which are the important bugs?

and how do you find them?

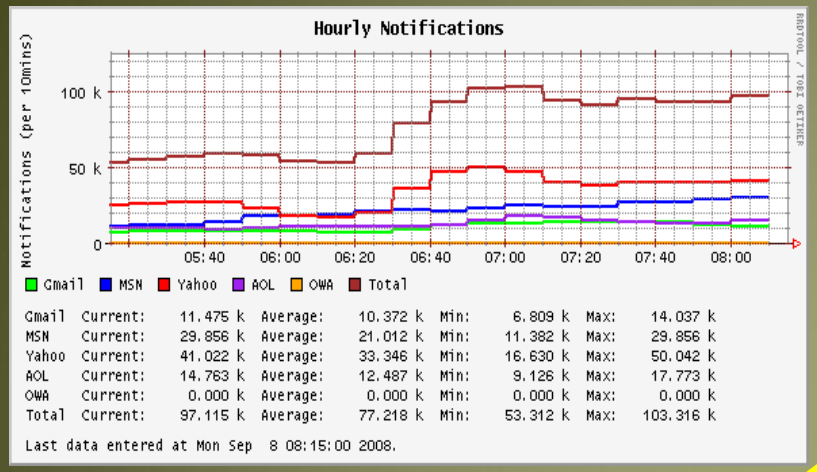
They are not obvious, otherwise they would have been found during testing

The really tricky bugs you
find with sophisticated tools

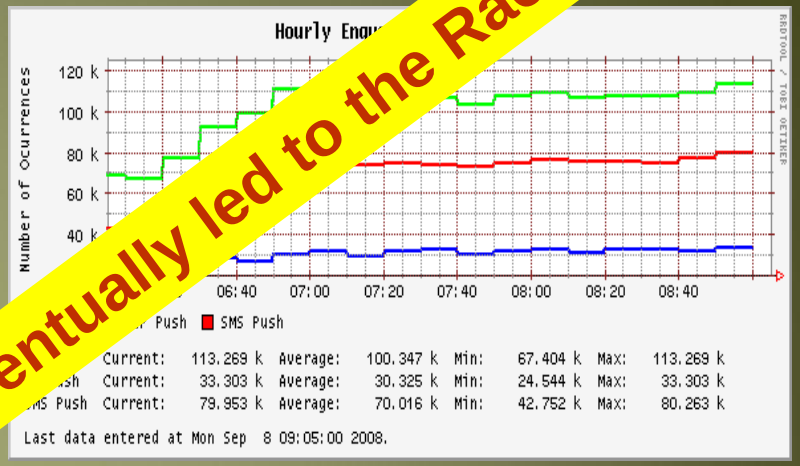
We use log scraping (tail, awk, grep,orca)
and SQL queries to a reporting DB
visualized with EXCEL pivot tables

You only see the important bugs in large volumes of data

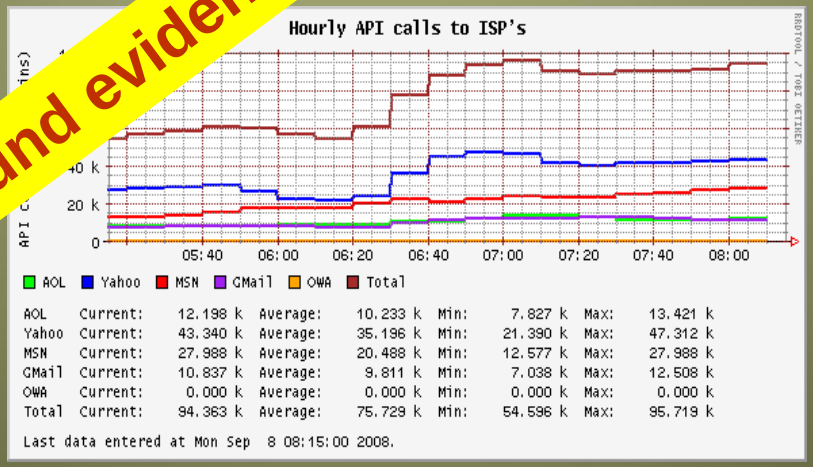
Monitoring Graphs



Notifications from ISPs



Triggers sent to Clients



Clients Syncing to retrieve Email

This is how we found evidence that eventually led to the Race bug

But where is the research?

Log files

Runtime Monitoring

How do you build behavioral models from the log file

Sequence of outputs

Interleavings

Very large

They need to be probabilistic

Explosion

Performar

What kind of properties do you check?

lity

Where do we get the data from?

Live System

Load Testing

Performance Testing

Load testing under peak load

- Expensive
  - If you are going to test under production load you need production level hardware
- Difficult
  - One needs to “fake” the environment so well that the system thinks it is dealing with the real thing
  - Not only must the system under test scale, but also the test environment

In the SEVEN setting

Performance testing finds more bugs than any other form of testing, automated or otherwise

But where is the research?

Can we calculate the
performance characteristics of the code?

Inefficient code…”hot” locks…???

Can we generate tests
to expose these issues?

Performance Bug Examples

java.text.SimpleDateFormat.parse

Some fun facts

Not thread safe

Inefficient

Too few → Hot locks

Too many → High CPU load

public String preserveTags(String body) {…}

Infinite loops is the worst kind of error,
since it is input driven and therefore can reappear
frequently, in fact infinitely often!

Who Really Cares if the Program Cra

Who really wants to find the bug in the
deepest darkest corner of the state-space

They care about bugs
that happen all the time

My Future Work

Discovering
non-termination

Performance
Analysis & Testing

Conclusions

Conclusions

Look beyond the “once in a lifetime” bugs

Current static analysis tools finds mostly unimportant bugs

What is model checking’s role?

Conclusions

Conclusions