Tracking Bad Apples: Reporting the Origin of Null & Undefined Value Errors

Michael D. Bond
UT Austin

Nicholas Nethercote
National ICT Australia

Stephen W. Kent
UT Austin

Samuel Z. Guyer
Tufts University

Kathryn S. McKinley
UT Austin

Example Code

User code:

```
float[][] data =
  {{1.0f, 2.0f}, {3.0f, 4.0f}};

ScatterPlot plot =
  new ScatterPlot(data, null);

...

plot.draw(...);
```

Example Code

User code:

```
float[][] data =
  {{1.0f, 2.0f}, {3.0f, 4.0f}};

ScatterPlot plot =
  new ScatterPlot(data, null);

...

plot.draw(...);
```

```
NullPointerException
  at ScatterPlot.draw():315
  at Test.doStuff():124
```

Example Code

User code:

```
float[][] data =
  {{1.0f, 2.0f}, {3.0f, 4.0f}};

ScatterPlot plot =
  new ScatterPlot(data, null);

...

plot.draw(...);
```

Library code:

```
ScatterPlot {
  draw(...) {
    ...
    xaxis.draw();
    ...
  }
}
```

```
NullPointerException
  at ScatterPlot.draw():315
  at Test.doStuff():124
```

Unusable Value

- Null is unusable value
  - Use causes error
- How/why did it become null?
  - Null's origin?

```
ScatterPlot {
  draw(...) {
    ...
    xaxis.draw();
    ...
  }
}
```

```
plot.draw(...);
```

```
NullPointerException
  at ScatterPlot.draw():315
  at Test.doStuff():124
```

Origin of Unusable Value

```
float[][] data =
  {{1.0f, 2.0f}, {3.0f, 4.0f}};

ScatterPlot plot =
  new ScatterPlot(data, null);
...

plot.draw(...);
```

```
Origin: Test.init():37
```

```
ScatterPlot {
  draw(...) {
    ...
    xaxis.draw();
    ...
  }
}
```

Origin of Unusable Value

```
float[][] data =
  {{1.0f, 2.0f}, {3.0f, 4.0f}};

ScatterPlot plot =
  new ScatterPlot(data, null);
```

Origin: Test.init():37

```
...

plot.draw(...);
```

```
ScatterPlot {
  ScatterPlot(data, xaxis) {
    this.data = data;
    this.xaxis = xaxis;
  }

  ...

  draw(...) {
    ...
    xaxis.draw();
    ...
  }
}
```

Origin: Test.init():37

Origin Tracking

```
float[][] data =
  {{1.0f, 2.0f}, {3.0f, 4.0f}};

ScatterPlot plot =
  new ScatterPlot(data, null);
```

Origin: Test.init():37

```
...
```

```
ScatterPlot {
  ScatterPlot(data, xaxis) {
    this.data = data;
    this.xaxis = xaxis;
  }

  ...

  draw(...) {
    ...
    xaxis.draw();
    ...
  }
}
```

Origin: Test.init():37

Track every unusable value's origin for 4% overhead

Key: store origin <u>in place</u> of unusable value

Outline

- Introduction
- Unusable values
- Instances of origin tracking
  - Null pointer exceptions (Java)
  - Undefined value errors (C/C++)
- Redefining program operations
- Evaluation
  - Performance
  - Usefulness
- Related Work

Unusable Values

- Using value causes error
- Examples:
  - Null values
  - Undefined values
- Tough bugs: no info
- **Why** is value unusable?
  - Where did unusable value originate?

Unusable Values

- Using value causes error
- Examples:
  - Null values
  - Undefined values
- Tough bugs: no info
- **Why** is value unusable?
  - Where did unusable value originate?

Opportunity:
Store info **in place** of value

Origin Tracking Implementations

- Null pointers (Java)
  - Jikes RVM
- Undefined values (native code)
  - Valgrind's MemCheck

Origin Tracking Implementations

- Null pointers (Java)
  - Jikes RVM Jikes RVM Research Archive
- Undefined values (native code)
  - Valgrind's MemCheck Valgrind Source Code Repository

Origin Tracking Implementations

- Null pointers (Java)
  - Jikes RVM Jikes RVM Research Archive
- Undefined values (native code)
  - Valgrind's MemCheck Valgrind Source Code Repository
  - Identifies origin for 32-bit values
    - 47 of 147 are 32-bit
    - For 32-bit: 34 of 47 identified
  - Adds negligible overhead (28X → 28X)

Storing Origins in Null Values

- Requirements
  - Need bits in null values → multiple null values
  - Program operations support null values
  - Recover origin at exception

Storing Origins in Null Values

- Requirements
  - Need bits in null values → multiple null values
  - Program operations support null values
  - Recover origin at exception

Null ⇔ high 5 bits are zero (27 bits available)
Reserve & protect address range: **`0x00000000-0x07ffffff`**

00000_2	Method ID	Bytecode index
5 bits	14 bits	13 bits

Implementing Java Operations

	Java semantics	Standard VM	Origin tracking
Assignment of null constant	`obj = null;`	`obj = 0;`	`obj = this_location;`
Object allocation	`obj = new Object();`	`foreach ref slot i` `  obj[i] = 0;`	`foreach ref slot i` `  obj[i] = this_location;`
Null reference comparison	`if (obj == null) {`	`if (obj == 0) {`	`if ((obj & 0xf8000000) == 0) {`

General reference comparison	`if (obj1 == obj2) {`	`if (obj1 == obj2) {`	`if (((obj1 & 0xf8000000) == 0) ?`
			`  ((obj2 & 0xf8000000) == 0) :`
			`  (obj1 == obj2)) {`

Implementing Java Operations

	Java semantics	Standard VM	Origin tracking
Assignment of null constant	`obj = null;`	`obj = 0;`	`obj = this_location;`
Object allocation	`obj = new Object();`	`foreach ref slot i` `  obj[i] = 0;`	`foreach ref slot i` `  obj[i] = this_location;`
Null reference comparison	`if (obj == null) {`	`if (obj == 0) {`	`if ((obj & 0xf8000000) == 0) {`

General reference comparison	`if (obj1 == obj2) {`	`if (obj1 == obj2) {`	`if (((obj1 & 0xf8000000) == 0) ?`
			`  ((obj2 & 0xf8000000) == 0) :`
			`  (obj1 == obj2)) {`

Implementing Java Operations

	Java semantics	Standard VM	Origin tracking
Assignment of null constant	`obj = null;`	`obj = 0;`	`obj = this_location;`
Object allocation	`obj = new Object();`	`foreach ref slot i` `  obj[i] = 0;`	`foreach ref slot i` `  obj[i] = this_location;`
Null reference comparison	`if (obj == null) {`	`if (obj == 0) {`	`if ((obj & 0xf8000000) == 0) {`

General reference comparison	`if (obj1 == obj2) {`	`if (obj1 == obj2) {`	`if (((obj1 & 0xf8000000) == 0) ?`
			`  ((obj2 & 0xf8000000) == 0) :`
			`  (obj1 == obj2)) {`

Implementing Java Operations

	Java semantics	Standard VM	Origin tracking
Assignment of null constant	`obj = null;`	`obj = 0;`	`obj = this_location;`
Object allocation	`obj = new Object();`	`foreach ref slot i` `  obj[i] = 0;`	`foreach ref slot i` `  obj[i] = this_location;`
Null reference comparison	`if (obj == null) {`	`if (obj == 0) {`	`if ((obj & 0xf8000000) == 0) {`

General reference comparison	`if (obj1 == obj2) {`	`if (obj1 == obj2) {`	`if (((obj1 & 0xf8000000) == 0) ?`
			`  ((obj2 & 0xf8000000) == 0) :`
			`  (obj1 == obj2)) {`

Implementing Java Operations

	Java semantics	Standard VM	Origin tracking
Assignment of null constant	`obj = null;`	`obj = 0;`	`obj = this_location;`
Object allocation	`obj = new Object();`	`foreach ref slot i` `  obj[i] = 0;`	`foreach ref slot i` `  obj[i] = this_location;`
Null reference comparison	`if (obj == null) {`	`if (obj == 0) {`	`if ((obj & 0xf8000000) == 0) {`

General reference comparison	`if (obj1 == obj2) {`	`if (obj1 == obj2) {`	`if (((obj1 & 0xf8000000) == 0) ?`
			`  ((obj2 & 0xf8000000) == 0) :`
			`  (obj1 == obj2)) {`

Implementing Java Operations

	Java semantics	Standard VM	Origin tracking
Assignment of null constant	`obj = null;`	`obj = 0;`	`obj = this_location;`
Object allocation	`obj = new Object();`	`foreach ref slot i` `  obj[i] = 0;`	`foreach ref slot i` `  obj[i] = this_location;`
Null reference comparison	`if (obj == null) {`	`if (obj == 0) {`	`if ((obj & 0xf8000000) == 0) {`

General reference comparison	`if (obj1 == obj2) {`	`if (obj1 == obj2) {`	`if (((obj1 & 0xf8000000) == 0) ?`
			`  ((obj2 & 0xf8000000) == 0) :`
			`  (obj1 == obj2)) {`

General assignment	`obj1 = obj2;`	`obj1 = obj2;`	`obj1 = obj2;`

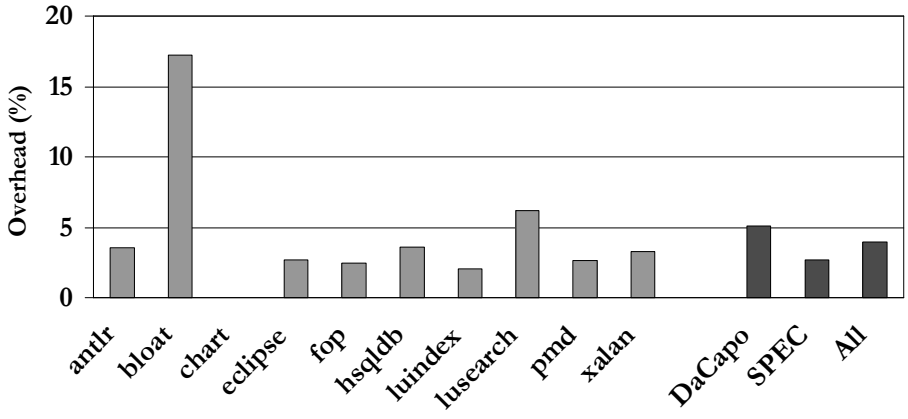
Outline

- Introduction
- Unusable values
- Instances of origin tracking
  - Null pointer exceptions (Java)
  - Undefined value errors (C/C++)
- Redefining program operations
- Evaluation
  - Performance
  - Usefulness
- Related Work

Methodology

- Adaptive methodology
  - Mix of application & compilation time
  - Single iteration; 25 trials
- DaCapo, SPEC JBB2000, SPEC JVM98
- 3.6 GHz Pentium 4 w/Linux

Performance of Java Implementation



Finding and Fixing Bugs

- 12 real NPEs from SourceForge

Origin: identified by origin tracking?

Triviality: origin obvious by inspection?

Usefulness: origin useful for fixing bug?

Null Pointer Exceptions

Program	Lines	Exception description	Origin?	Trivial?	Useful?
Mckoi SQL DB	94,681	Access closed connection	Yes	Nontrivial	Definitely useful
FreeMarker	64,442	JUnit test crashes unexpectedly	Yes	Nontrivial	Definitely useful
JFreeChart	223,869	Plot without x-axis	Yes	Nontrivial	Definitely useful
JRefactory	231,338	Invalid class name	Yes	Nontrivial	Definitely useful
Eclipse	2,425,709	Malformed XML document	Yes	Nontrivial	Most likely useful
Checkstyle	47,871	Empty default case	Yes	Nontrivial	Most likely useful
JODE	44,937	Exception decompiling class	Yes	Nontrivial	Most likely useful
Jython	144,739	Use built-in class as variable	Yes	Nontrivial	Potentially useful
JFreeChart	223,869	Stacked XY plot with lines	Yes	Somewhat nontrivial	Marginally useful
Jython	144,739	Problem accessing __doc__	Yes	Somewhat nontrivial	Marginally useful
JRefactory	231,338	Package and import on same line	Yes	Trivial	Not useful
Eclipse	2,425,709	Close Eclipse while deleting project	Yes	Trivial	Not useful

Null Pointer Exceptions

Program	Lines	Exception description	Origin?	Trivial?	Useful?
Mckoi SQL DB	94,681	Access closed connection	Yes	Nontrivial	Definitely useful
FreeMarker	64,442	JUnit test crashes unexpectedly	Yes	Nontrivial	Definitely useful
JFreeChart	223,869	Plot without x-axis	Yes	Nontrivial	Definitely useful
JRefactory	231,338	Invalid class name	Yes	Nontrivial	Definitely useful
Eclipse	2,425,709	Malformed XML document	Yes	Nontrivial	Most likely useful
Checkstyle	47,871	Empty default case	Yes	Nontrivial	Most likely useful
JODE	44,937	Exception decompiling class	Yes	Nontrivial	Most likely useful
Jython	144,739	Use built-in class as variable	Yes	Nontrivial	Potentially useful
JFreeChart	223,869	Stacked XY plot with lines	Yes	Somewhat nontrivial	Marginally useful
Jython	144,739	Problem accessing __doc__	Yes	Somewhat nontrivial	Marginally useful
JRefactory	231,338	Package and import on same line	Yes	Trivial	Not useful
Eclipse	2,425,709	Close Eclipse while deleting project	Yes	Trivial	Not useful

Null Pointer Exceptions

Program	Lines	Exception description	Origin?	Trivial?	Useful?
Mckoi SQL DB	94,681	Access closed connection	Yes	Nontrivial	Definitely useful
FreeMarker	64,442	JUnit test crashes unexpectedly	Yes	Nontrivial	Definitely useful
JFreeChart	223,869	Plot without x-axis	Yes	Nontrivial	Definitely useful
JRefactory	231,338	Invalid class name	Yes	Nontrivial	Definitely useful
Eclipse	2,425,709	Malformed XML document	Yes	Nontrivial	Most likely useful
Checkstyle	47,871	Empty default case	Yes	Nontrivial	Most likely useful
JODE	44,937	Exception decompiling class	Yes	Nontrivial	Most likely useful
Jython	144,739	Use built-in class as variable	Yes	Nontrivial	Potentially useful
JFreeChart	223,869	Stacked XY plot with lines	Yes	Somewhat nontrivial	Marginally useful
Jython	144,739	Problem accessing __doc__	Yes	Somewhat nontrivial	Marginally useful
JRefactory	231,338	Package and import on same line	Yes	Trivial	Not useful
Eclipse	2,425,709	Close Eclipse while deleting project	Yes	Trivial	Not useful

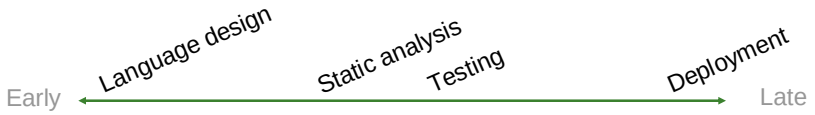
Null Pointer Exceptions

Program	Lines	Exception description	Origin?	Trivial?	Useful?
Mckoi SQL DB	94,681	Access closed connection	Yes	Nontrivial	Definitely useful
FreeMarker	64,442	JUnit test crashes unexpectedly	Yes	Nontrivial	Definitely useful
JFreeChart	223,869	Plot without x-axis	Yes	Nontrivial	Definitely useful
JRefactory	231,338	Invalid class name	Yes	Nontrivial	Definitely useful
Eclipse	2,425,709	Malformed XML document	Yes	Nontrivial	Most likely useful
Checkstyle	47,871	Empty default case	Yes	Nontrivial	Most likely useful
JODE	44,937	Exception decompiling class	Yes	Nontrivial	Most likely useful
Jython	144,739	Use built-in class as variable	Yes	Nontrivial	Potentially useful
JFreeChart	223,869	Stacked XY plot with lines	Yes	Somewhat nontrivial	Marginally useful
Jython	144,739	Problem accessing __doc__	Yes	Somewhat nontrivial	Marginally useful
JRefactory	231,338	Package and import on same line	Yes	Trivial	Not useful
Eclipse	2,425,709	Close Eclipse while deleting project	Yes	Trivial	Not useful

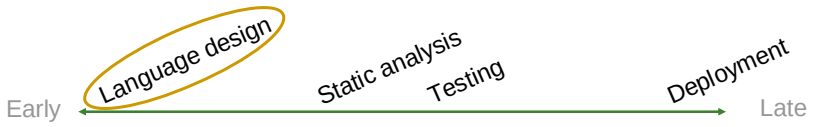
Null Pointer Exceptions

Program	Lines	Exception description	Origin?	Trivial?	Useful?
Mckoi SQL DB	94,681	Access closed connection	Yes	Nontrivial	Definitely useful
FreeMarker	64,442	JUnit test crashes unexpectedly	Yes	Nontrivial	Definitely useful
JFreeChart	223,869	Plot without x-axis	Yes	Nontrivial	Definitely useful
JRefactory	231,338	Invalid class name	Yes	Nontrivial	Definitely useful
Eclipse	2,425,709	Malformed XML document	Yes	Nontrivial	Most likely useful
Checkstyle	47,871	Empty default case	Yes	Nontrivial	Most likely useful
JODE	44,937	Exception decompiling class	Yes	Nontrivial	Most likely useful
Jython	144,739	Use built-in class as variable	Yes	Nontrivial	Potentially useful
JFreeChart	223,869	Stacked XY plot with lines	Yes	Somewhat nontrivial	Marginally useful
Jython	144,739	Problem accessing __doc__	Yes	Somewhat nontrivial	Marginally useful
JRefactory	231,338	Package and import on same line	Yes	Trivial	Not useful
Eclipse	2,425,709	Close Eclipse while deleting project	Yes	Trivial	Not useful

Debugging Timeline



Debugging Timeline



Prevent bugs

- Memory bugs [Java & C#]
- Null pointer exceptions [Chalin & James '06]
  - Add "never null" types to Java
  - Programmer effort
  - Exceptions still possible
- Functional languages

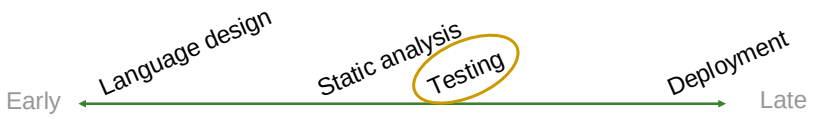
Debugging Timeline

Detect errors in **any** execution

[FindBugs] [PMD] [ESC/Java] [JLint] [Metal]

- Dataflow analysis & pattern matching
- Program complexity → conservative (false positives)
- Some intentionally unsound (false negatives)

Debugging Timeline

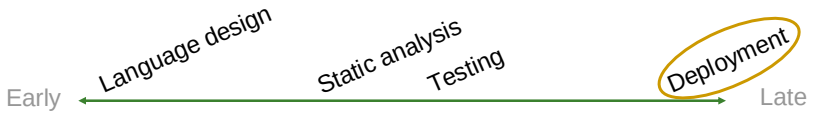


Catch errors in real executions

- Assertions
- Checking tools [Valgrind] [Purify]
- Dynamic slicing [Agrawal & Horgan '90] [Zhang et al. '07]

Powerful but high overhead

Debugging Timeline



Ideal environment for debugging?

- Stack/dump reporting
- Invariant-based bug detection [Liblit et al. '05]
  - Many executions
- Limited dynamic slicing
  [TaintCheck] [Chilimbi & Hauswirth '04] [Origin tracking]
  - Single execution
  - Narrow focus

Summary

- Unusable values: tough bugs
  - Error stack trace not enough
  - Opportunity: store origin in place of unusable value
  - Managed and native implementations
- Java null origins: fast, useful, silent
- Add it to your VM today!

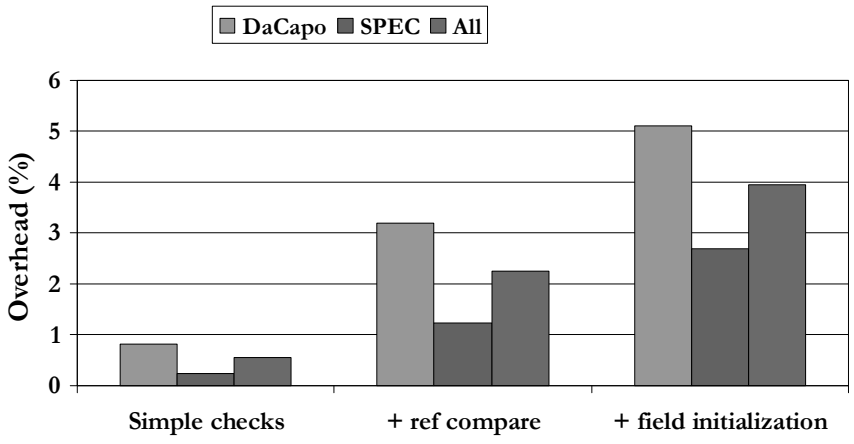
Summary

- Unusable values: tough bugs
  - Error stack trace not enough
  - Opportunity: store origin in place of unusable value
  - Managed and native implementations
- Java null origins: fast, useful, silent
- Add it to your VM today!

Thank you!

Extra slides

Runtime Overhead



(origin tracking)