Design and Implementation of a Backward-In-Time Debugger

Christoph Hofer
Marcus Denker
Stephane Ducasse
Roadmap

- Problem
- Unstuck: A new Debugger for Squeak
- Implementation
- Lessons Learned
- Future work
Problem

> Debugger: Snapshot of state at time of error

> Cause for errors is in the past
  — Who assigned *that* value?

> very incomplete history available
  — guess were to set breakpoint, rerun
Example

```
Foo>>initialize
    var1 := 0.
    var2 := ''. 

Foo>>start
    self beforeBar.
    self bar.
    self moreBar.

Foo>>beforeBar
    var1 = 0 ifTrue: [ 
        var2 := nil.
    ]

Foo>>bar
    .......

Foo>>moreBar
    var2 size > 0 ifTrue: [
        ^var2 at: 1].
    ^''
```
Stack Trace

> Squeak Debugger
> Shows stack trace
  - methods not returned
  - old state lost
Stack Trace

> Squeak Debugger
> Shows stack trace
  — methods not returned
  — old state lost

Methods in italic means their execution is finished
Solution

> We want:
  - Record the history of the program
  - View the state at any point in the past

> Unstuck
  - A new Debugger for Squeak
  - Provides full trace information
Unstuck UI

1. Trace
2. Object
3. Code
4. History
5. Query
## Searching

<table>
<thead>
<tr>
<th>Variable</th>
<th>Search Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>All events</td>
</tr>
<tr>
<td>send</td>
<td>all message sends</td>
</tr>
<tr>
<td>return</td>
<td>all method returns</td>
</tr>
<tr>
<td>varAccess</td>
<td>all variable accesses</td>
</tr>
<tr>
<td>instVarAccess</td>
<td>instance variable access</td>
</tr>
<tr>
<td>tempVarAccess</td>
<td>temporary variables</td>
</tr>
</tbody>
</table>
## Searching: Example

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>send selector = #foo</code></td>
<td>All the executed methods named ”foo”</td>
</tr>
<tr>
<td><code>return returnValue &gt; 4</code></td>
<td>All returns with a return value greater than 4</td>
</tr>
<tr>
<td><code>events isSend &amp; (even arguments size = 1)</code></td>
<td>Message sends with exactly one argument</td>
</tr>
</tbody>
</table>
Coloring

> We can assign a color to any object

> Easy tracking of objects

> Color is shown in all views of the UI
Implementation

Debugger

Trace Library

uses

ByteSurgeon

uses
ByteSurgeon

> Framework for editing bytecode for Squeak
  — Like Javasist in Java, but:

> Uses structural reflection to transform at runtime
  — Simple model: Inline code before / after a bytecode
  — Inlined code is normal smalltalk code
  — Not much knowledge about bytecode needed
Trace Library

- Called from annotated code
- Builds up the trace

- Provides
  - Trace model
  - Event pre-processing (ordering)
  - State reconstruction
State reconstruction

> State not recorded for completely annotated classes
  — past state can be reconstructed from trace

> System never completely annotated
  — Tracer saves state of non-annotated objects
Debugging system classes

> Annotate classes used by Bytesurgeon or Tracer
  - System classes (e.g. Collection or String)
  - Compiler (e.g. AST)

> Problems:
  - Classes used for annotation --> crash
  - Tracer records events generated by the tracer
Solution

- Retain both methods (original + annotated)
- Generate preamble
  - test for global
  - call original methods when inactive

- Common problem when using reflection!
  - General solution?
  - Future work!
## Benchmarks

<table>
<thead>
<tr>
<th></th>
<th>Events</th>
<th>Slowdown</th>
<th>Memory (Kb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>74</td>
<td>6</td>
<td>16</td>
</tr>
<tr>
<td>AST Bug</td>
<td>2725</td>
<td>3.8</td>
<td>800</td>
</tr>
<tr>
<td>Pier Trace</td>
<td>389689</td>
<td>248</td>
<td>88800</td>
</tr>
</tbody>
</table>
Future Work

> further analyze + improve
  — Memory Consumption (GC effects)
  — Performance

> Use behavioral reflection
  — fine grained selection
  — Scoping
  — Annotation of system classes
Conclusion

> Problem of current debugging tools

> Overview of Unstuck
   - UI
   - Implementation

> Having the history available helps
   - Possible for small programs
   - Work needed for bigger systems + continuous use
Conclusion

> Problem of current debugging tools

> Overview of Unstuck
  – UI
  – Implementation

> Having the history available helps
  – Possible for small programs
  – Work needed for bigger systems + continuous use

Questions?