Evaluating the dynamic behavior of Smalltalk applications

Bachelor thesis
Roger Stebler

Supervisor: Boris Spasojević
SCG Uni Berne
Overview

• Introduction
• VariableTracker
• Case Studies
• Conclusion
Introduction

• Give a man a dynamically typed language, will he write dynamically typed code?

|a|
a := 42.
a := ‘hello world’.
a := Dictionary new.

• Do developers need/use this behavior?

|a b c|
a := 42.
b := ‘hello world’.
c := Dictionary new.
Goals

• Program to monitor how types change during execution
• Data is stored to MongoDB
• Program to analyze the results from the database
• Draw conclusions
Overview

• Introduction

• **VariableTracker**
  – Ad-hoc polymorphism
  – Reflectivity
  – First and current version
  – Speed comparison

• Case Studies

• Conclusion
VariableTracker

• Collect and analyze information about variables being written during run time
• Collected data can be written to a MongoDB
• Analysis can find ad-hoc polymorphic variables

VariableTracker + Reflectivity  

MongoTalk

mongoDB
Reflectivity

«Reflectivity is a tool to annotate AST nodes with metalinks. A metalink corresponds to a message sent to an arbitrary object.»

Example:

```smalltalk
MyClass methods do: [ :method |
    method ast
    forAllNodes: [ :node | node isAssignment ]
    putAfter: [ RFMetalink fromExpression: 'Transcript crShow: ''variable written'' ' ];
    installWrapper ]
```

http://smalltalkhub.com/#!/~RMoD/Reflectivity
addWrapper: aMethod

aMethod ast

forAllNodes: [ :node | node isAssignment ]

putAfter: [ :node | RFMetalink fromExpression:
    'VariableTracker addToCache:','
    '((Dictionary new)',
    'at: ''name'' put: ''', node variable name asString, '''';',
    'at: ''class'' put: self class name asString;',
    'at: ''method'' put: thisContext method selector asString;',
    'at: ''isClassSide'' put: self class isClassSide;',
    'at: ''isInstanceVariable'' put: ', node variable isInstance asString, '''';',
    'at: ''isTemp'' put: ', node variable isTemp asString, '''';',
    'at: ''isArgument'' put: ', node variable isArgument asString, '''';',
    'at: ''isGlobal'' put: ', node variable isGlobal asString, '''';',
    'at: ''type'' put: '', node variable name asString, '' class asString;',
    'at: ''count'' put: 1;',
    'yourself).' ];

installWrapper.
VariableTracker
activateOnClass:
activateOnMethod:
deactivateOnClass:
...

MyClass
methodDict

Variable information is directly written to DB
## Speed comparison

<table>
<thead>
<tr>
<th></th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Without tracker</strong></td>
<td></td>
</tr>
<tr>
<td>Nautilus</td>
<td>~235ms</td>
</tr>
<tr>
<td>Roassal</td>
<td>~9.2s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>First version of VariableTracker</strong></th>
<th>Variables written</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nautilus</td>
<td>~200</td>
</tr>
<tr>
<td>Roassal</td>
<td>~25 million</td>
</tr>
</tbody>
</table>

**Actions:**
- Starting Nautilus browser
- Run Roassal example called «ForceLayoutWithGreatCharge»
Current version

VariableTracker

activateOnClass:
activateOnMethod:
deactivateOnMethod:
saveCacheToDB
...
cache
originalMethods

MyClass

methodDict

Variable information is stored in the cache – no interaction with the DB
# Speed comparison

<table>
<thead>
<tr>
<th></th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Without tracker</strong></td>
<td></td>
</tr>
<tr>
<td>Nautilus</td>
<td>~235ms</td>
</tr>
<tr>
<td>Roassal</td>
<td>~9.2s</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>First version of VariableTracker</strong></td>
<td></td>
</tr>
<tr>
<td>Nautilus</td>
<td>~15s</td>
</tr>
<tr>
<td>Roassal</td>
<td>~8h</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Current version of VariableTracker</strong></td>
<td></td>
</tr>
<tr>
<td>Nautilus</td>
<td>~245ms</td>
</tr>
<tr>
<td>Roassal</td>
<td>~11min</td>
</tr>
</tbody>
</table>

Actions:
- Starting Nautilus browser
- Run Roassal example called «ForceLayoutWithGreatCharge»

<table>
<thead>
<tr>
<th></th>
<th>Variables written</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nautilus</td>
<td>~200</td>
</tr>
<tr>
<td>Roassal</td>
<td>~25 million</td>
</tr>
</tbody>
</table>

Current version: ~50x faster

**Overhead**: ~0.3ms per variable assignment
Overview

• Introduction
• VariableTracker
• Case Studies
• Conclusion
Case studies

- Nautilus
  - default system browser since Pharo 2.0
- Roassal
  - a visualization engine
- Glamour
  - a framework for browser creation
- Phratch
  - a platform for kids to learn programming
- Pangea
  - an analysis tool on OO software corpora

<table>
<thead>
<tr>
<th>Case study</th>
<th>Variable assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nautilus</td>
<td>68’102</td>
</tr>
<tr>
<td>Roassal</td>
<td>28’290’898</td>
</tr>
<tr>
<td>Glamour</td>
<td>1’288’100</td>
</tr>
<tr>
<td>Phratch</td>
<td>9’197’807</td>
</tr>
<tr>
<td>Pangea</td>
<td>346’430</td>
</tr>
</tbody>
</table>
Overview

• Introduction
• VariableTracker
• Case Studies
• Conclusion
  – Results
  – Patterns
Kind of variable

- **Temporary variables**
- **Instance variables**
- **Method arguments**
- **Global variables**
- **Other**

### All

- Total: 100%
- **Temporary variables**: 50%
- **Instance variables**: 50%
- **Method arguments**: 0%
- **Global variables**: 0%
- **Other**: 0%

### Polymorphic variables

- Total: 100%
- **Temporary variables**: 60%
- **Instance variables**: 40%
- **Method arguments**: 0%
- **Global variables**: 0%
- **Other**: 0%

### Ad-hoc polymorphic variables

- Total: 100%
- **Temporary variables**: 45%
- **Instance variables**: 55%
- **Method arguments**: 0%
- **Global variables**: 0%
- **Other**: 0%
Patterns

• semantics are same, different APIs

```smalltalk
minValue := transformation rtValue: (metricBlock rtValue: elements anyOne model).
```

• object or BlockClosure

```smalltalk
coll := OrderedCollection new.
...
coll collect: [ :each | each asFloat ].
coll collect: #asFloat
```

• any object as data model

```smalltalk
model := anObject
```
Patterns

• collection or single element

```plaintext
roots := ROGraphTransformation new
fromEdgesToNesting: nodeCollection edges: edges.

(roots isKindOf: Collection) ifFalse: [
   roots := OrderedCollection with: roots.
].
```

• real ad-hoc usage

```plaintext
transformation := anObject
```

• potential wrong usage

```plaintext
button := action actionIcon.
```
Patterns

- real ad-hoc usage
- semantics are same, different APIs
- collection or single element
- object or BlockClosure
- any object as data model
- potential wrong usage
Summary

• VariableTracker can collect and analyze information about variables being written during run time
• Collected data can be written to a MongoDB
• Analysis can find ad-hoc polymorphic variables

Results of 5 case studies:
• Temporary- and instance variables each with ~50%
• Polymorphic (non ad-hoc polymorphic): ~8%
• Ad-hoc polymorphic: ~1%
• Real ad-hoc usage: ~40%