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

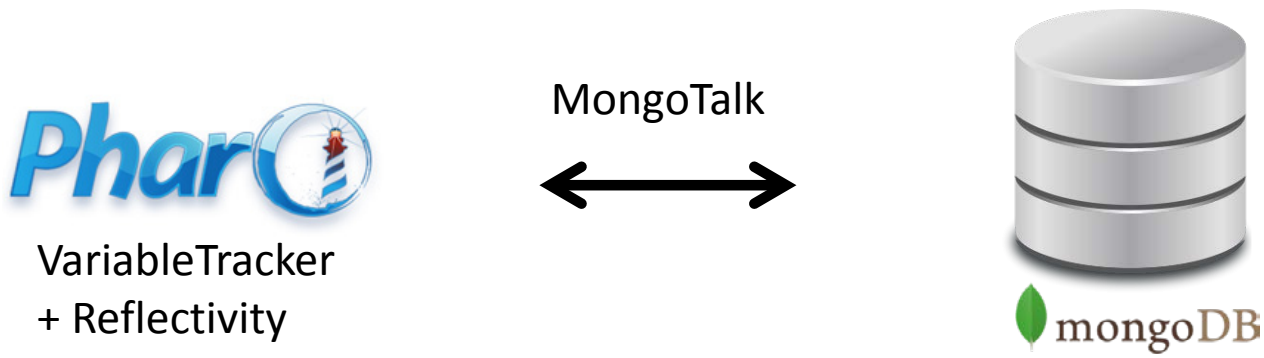
http://www.mongodb.org/

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



Reflectivity

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

Example:

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

Variable written

Add this code

Install wrapper

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

My code

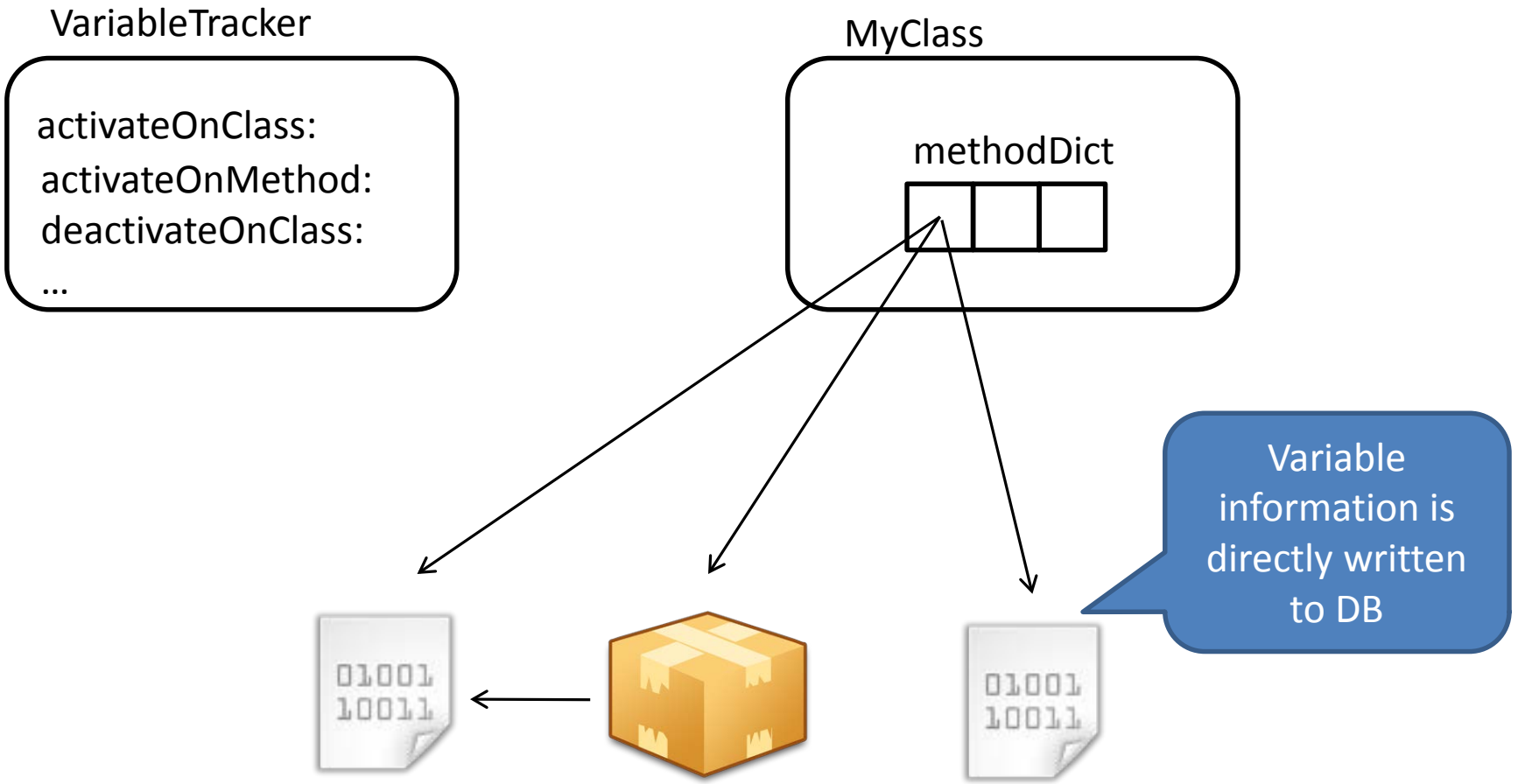
```
addWrapper: aMethod
   aMethod ast
      forAllNodes: [ :node | node isAssignment ]
      putAfter: [ :node | RFMetalink fromExpression:
         'VariableTracker addToCache:',
            '((Dictionary new)',
            'at: ''name'' put: ''', node variable name asString, ''';',
            'at: ''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.
```

Variable written

Add dictionary to cache

Install wrapper

First version



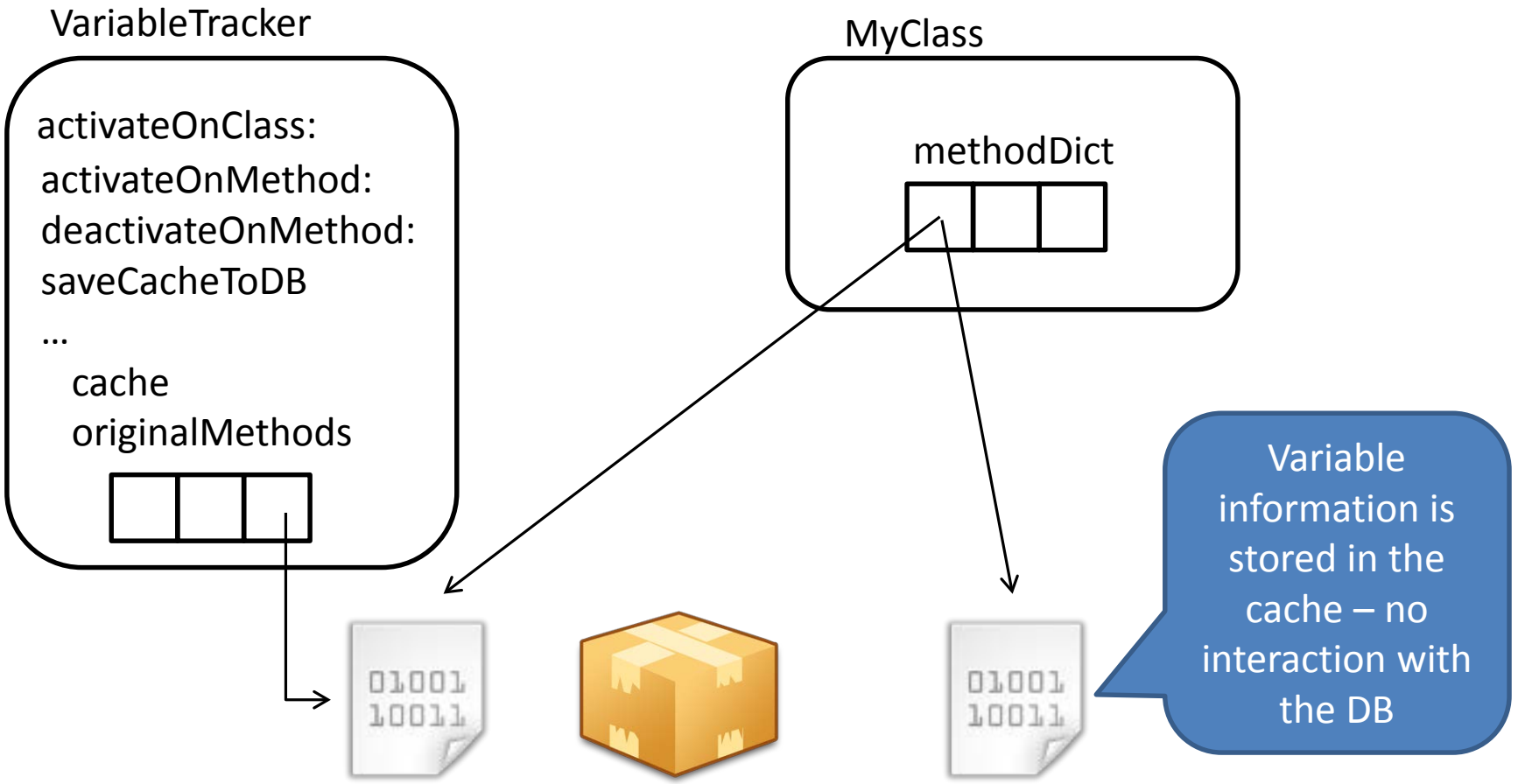
Speed comparison

	Duration
Without tracker	
Nautilus	~235ms
Roassal	~9.2s
First version of VariableTracker	
Nautilus	~15s
Roassal	~8h

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

	Variables written
Nautilus	~200
Roassal	~25 million

Current version



Speed comparison

	Duration
Without tracker	
Nautilus	~235ms
Roassal	~9.2s
First version of VariableTracker	
Nautilus	~15s
Roassal	~8h
Current version of VariableTracker	
Nautilus	~245ms
Roassal	~11min

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

	Variables written
Nautilus	~200
Roassal	~25 million

Current version: ~**50x** faster

Overhead: ~0.3ms per variable assignment

Overview

- Introduction
- VariableTracker
- **Case Studies**
- Conclusion

Case studies

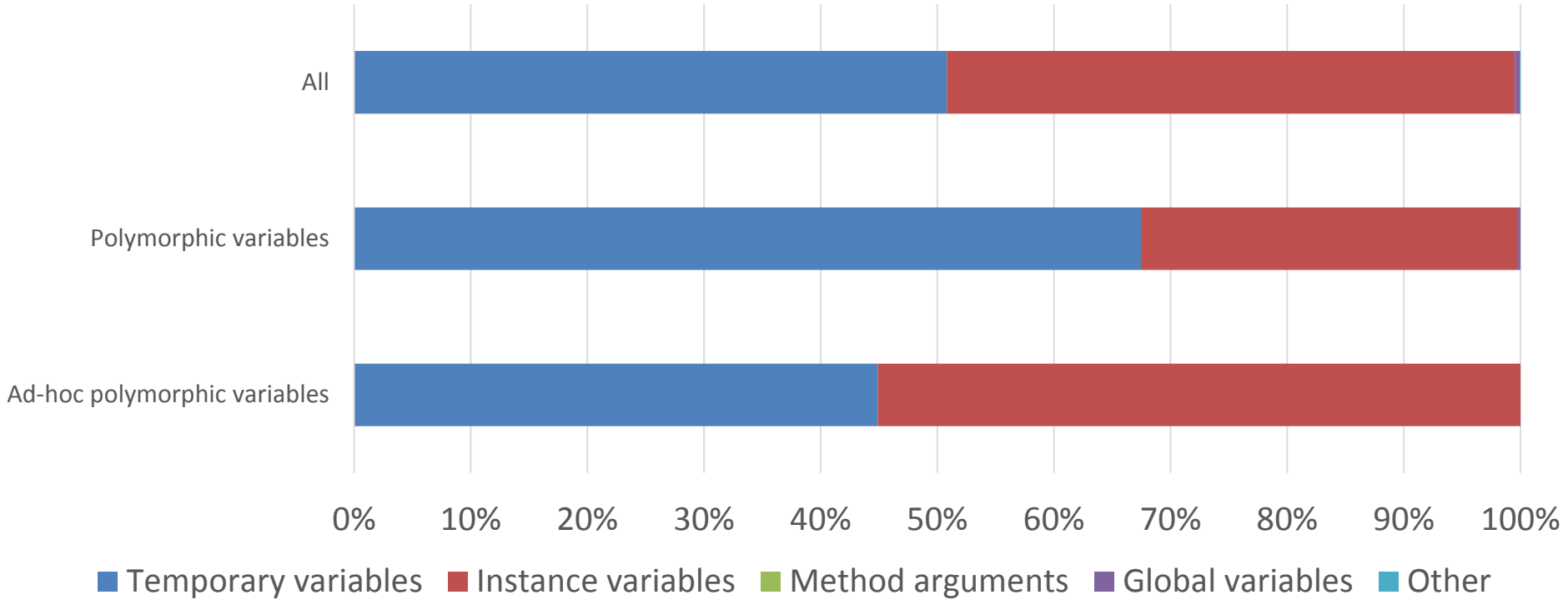
- Nautilus
  - default system browser since Pharo 2.0
- Roassal
  - a visualization engine
- Glamour
  - a framework for browser creation
- Phratch
  - a platform for kids to learn programming
- Pangea
  - an analysis tool on OO software corpora

Case study	Variable assignments
Nautilus	68’102
Roassal	28’290’898
Glamour	1’288’100
Phratch	9’197’807
Pangea	346’430

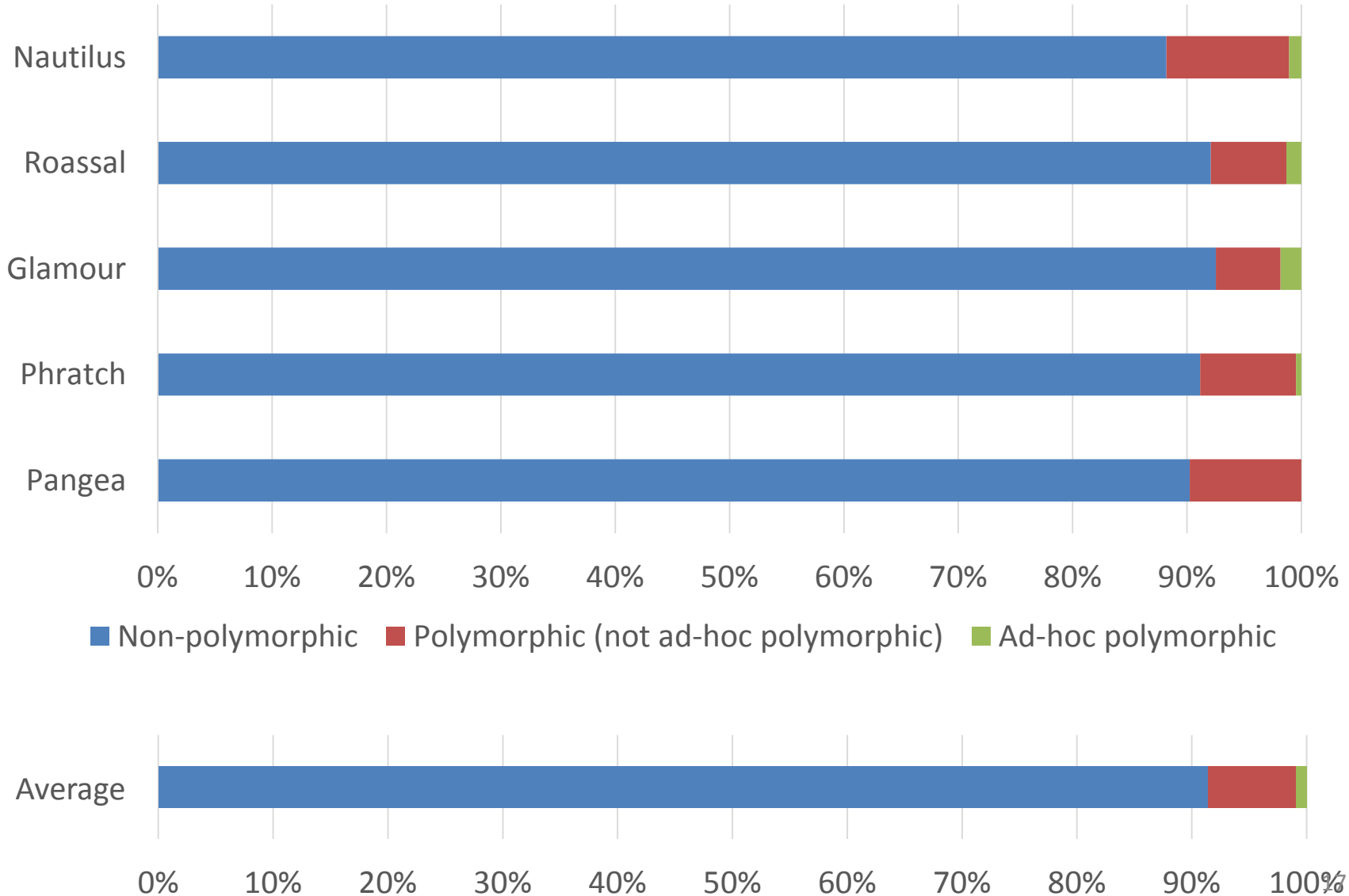
Overview

- Introduction
- VariableTracker
- Case Studies
- **Conclusion**
  - Results
  - Patterns

Kind of variable



Polymorphism



Patterns

- semantics are same, different APIs

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

- object or `BlockClosure`

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

- any object as data model

```
model := anObject
```

Patterns

- collection or single element

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

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

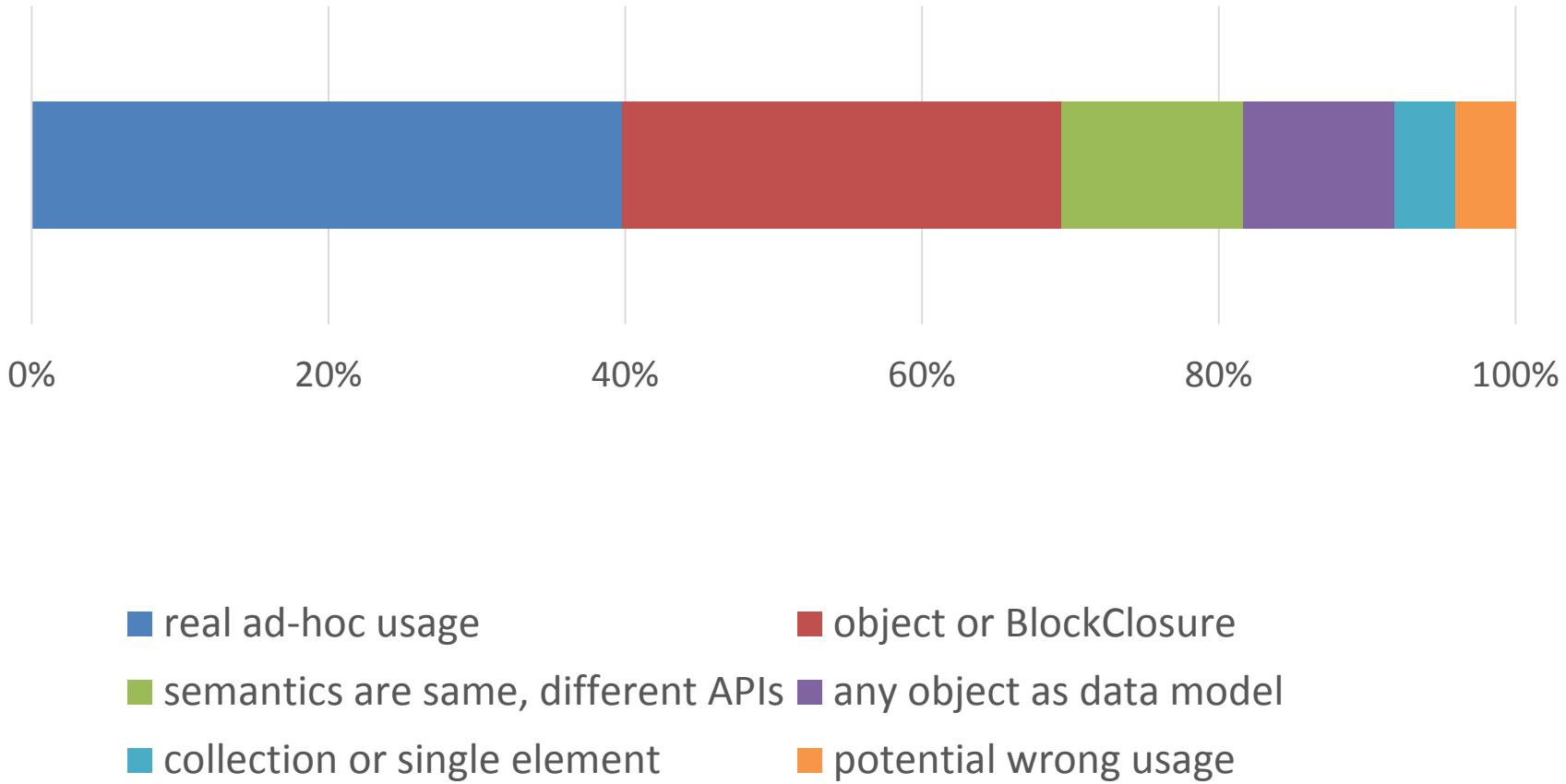
- real ad-hoc usage

```
transformation := anObject
```

- potential wrong usage

```
button := action actionIcon.
```

Patterns


0%
20%
40%
60%
80%
100%
real ad-hoc usage
object or BlockClosure
semantics are same, different APIs
any object as data model
collection or single element
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%