# Microprints

## A Pixel-Based, Semantically Rich Visualization Of Methods

Romain Robbes, Stéphane Ducasse
And Michele Lanza

# Outline

- Visualization techniques
- Microprints
- Microprints at work
- Customizing Microprints
- Conclusion

# Visualization Techniques

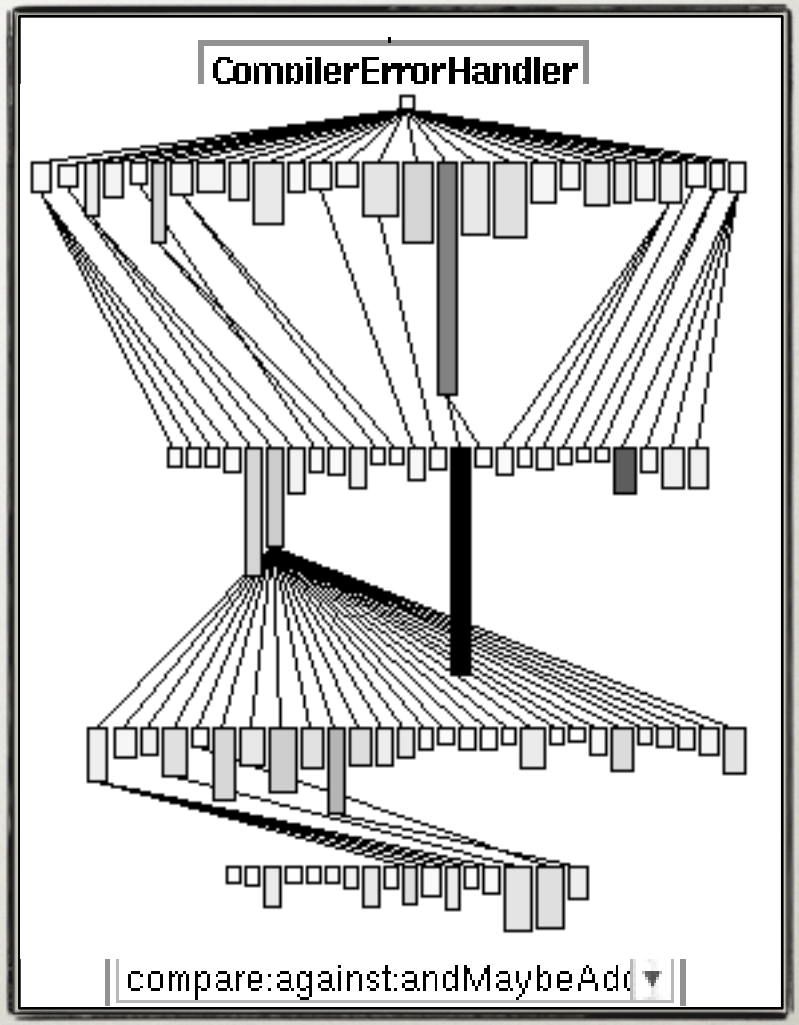
## And Some (Human) Limitations

# Why Visualizing Code?

- A way to:
  - Represent a lot of information in a small space
  - Choose starting points in an unknown code base
  - Facilitate code comprehension

# Visualisation Techniques

- Some examples:
  - UML
  - Syntax Highlighting
  - Polymetric Views



# Limitations

- Visualizations are interpreted by humans
- Unfortunately we are the limiting factor
  - Limited color choice
  - Limited information density

# Limited Color Choice

```
someMethod: anObject
	"Type a new method here and see how it formats as you type"

	| temp |
	#(a b c) do: [:tmp :temp | Transcript show: temp; cr; show: tmp].	"temp is redeclared"
	self isNil and: [temp isNil] ifTrue: [self foo].	"and:ifTrue: and #foo are not understood"
	temp := ('string' size + #symbol hash).
	(Array with: self with: super with: true with: thisContext) do: [:each | each halt].
	^undeclaredVariable value + 4 + $a asInteger
```

- We must avoid:
  - Colors too close to each other
  - Too many colors (7 max)

# Information Density

```
someMethod: anObject
    "Type a new method here and see how it formats as you type"

    | temp |
    #(a b c) do: [:tmp :temp | Transcript show: temp; cr; show: tmp].    "temp is redeclared"
    self isNil and: [temp isNil] ifTrue: [self foo].    "and:ifTrue: and #foo are not understood"
    temp := ('string' size + #symbol hash).
    (Array with: self with: super with: true with: thisContext) do: [:each | each halt].
    ^undeclaredVariable value + 4 + $a asInteger
```

- Too much information renders it useless
- Harder to locate a precise information...
  - Like finding a needle in a haystack

# Microprints

## Assumption, Principle and Advantages

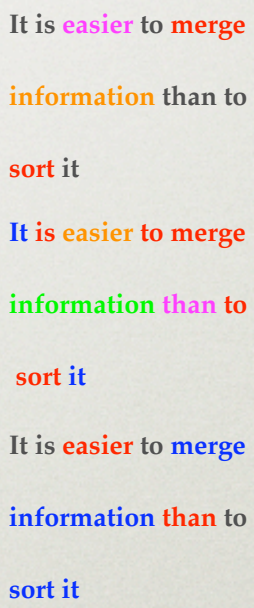
# Microprint Assumption

- It is easier to merge information than to sort it
- It is easier to merge information than to sort it
- It is easier to merge information than to sort it
- It is easier to merge information than to sort it

# But We Do Not Have That Much Space!

- So we reduce and stack each view:

It is easier to merge information than to sort it



# Microprint Principle

- Several views of a method via a variant of code highlighting:
  - Variable access
  - Control flow
  - Object interactions
  - Code and frequency coverage

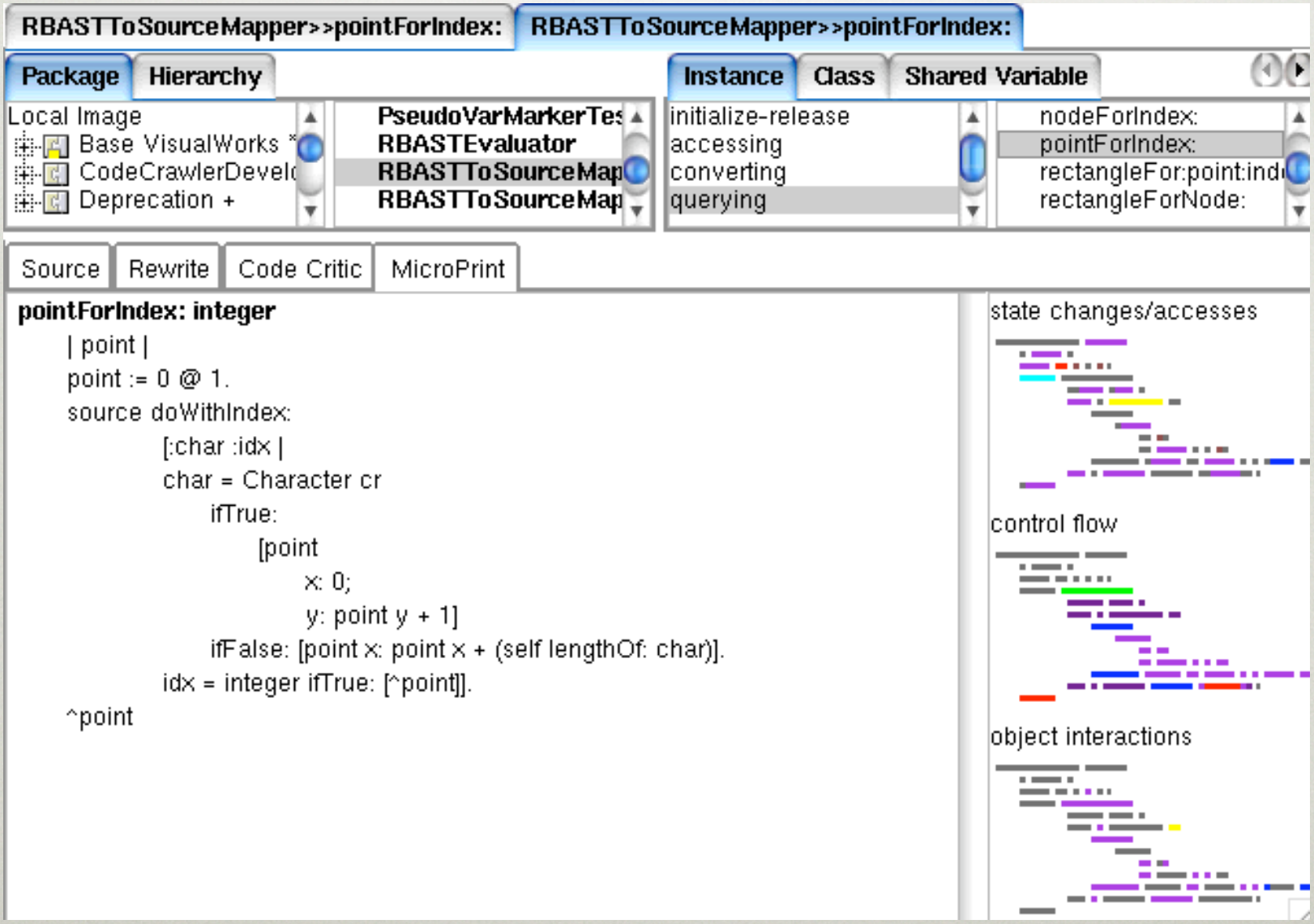
# Advantages of the Approach

- Several focused views
- Few information in each view
- Few colors as well
- Colors can be reused in several views

# Microprints at Work

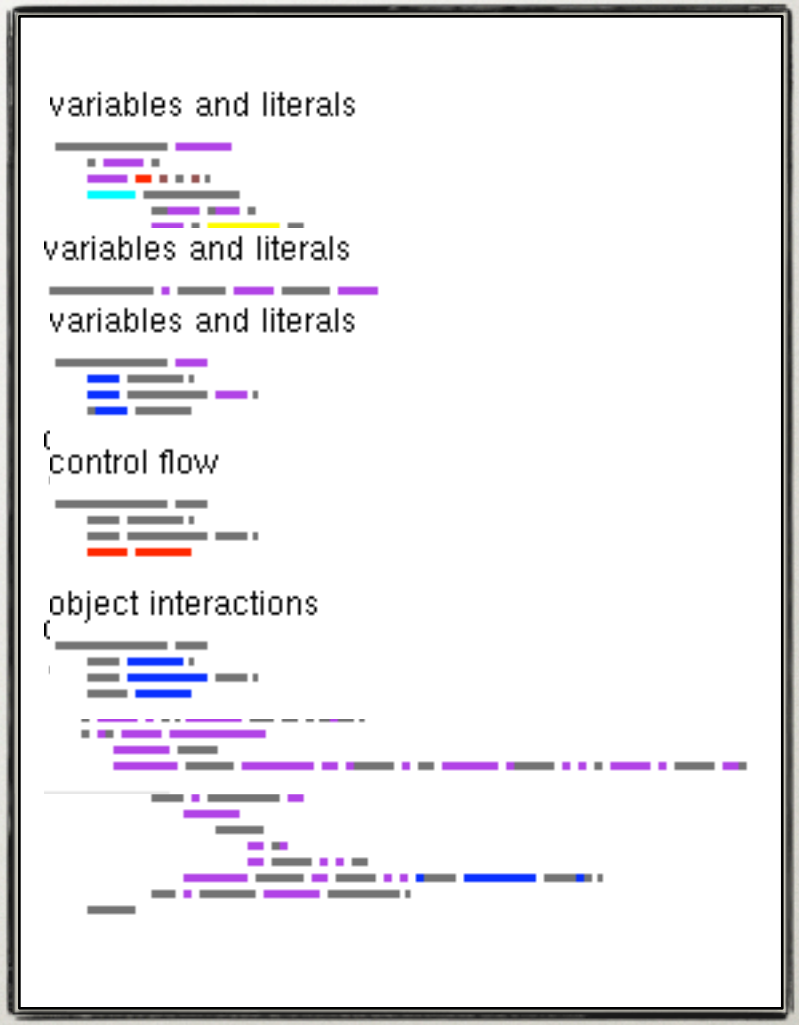
## Sample Patterns, Alternative Uses

# Microprints in Pictures



# Sample Visual Patterns

- Lazy initialisation
- Complex / trivial logic
- Template / framework method



# Alternative Uses of Microprints

- Visualizing:
  - Method protocols
  - Classes
  - Hierarchies
  - Or more?

# Customizing Microprints

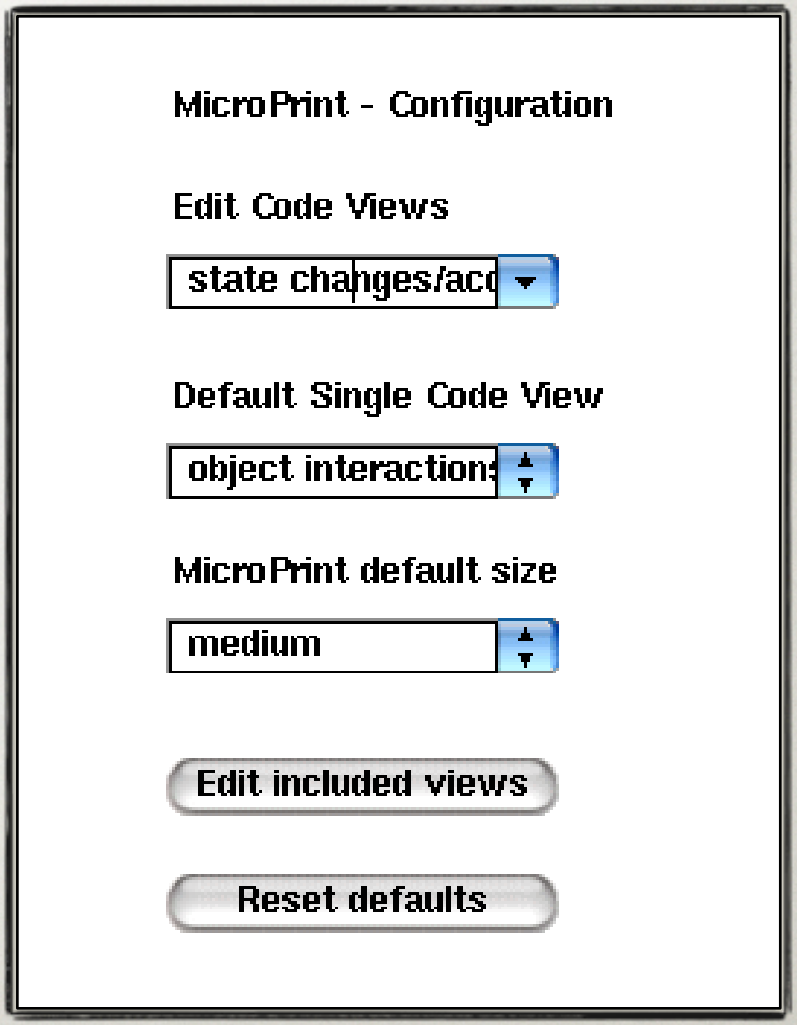
## Roll Your Own Microprints in Two Easy Lessons

# First: Some Implementation Details

- Microprints use the Refactoring Browser:
  - RB parser, AST and visitor
- Microprint = color to marker mapping
  - Markers = visitors of RB ASTs
  - Looking for “interesting” nodes

# Compose Your Own Microprint

- Use the GUI to:
  - Add markers
  - Map them to colors
  - Set their priorities



# Create Your Own Marker

- Define a new visitor
- When an AST node is interesting:
  - self mark: aNode (boolean)
  - self mark: aNode value: v (integer)
    - If you want to store a metric

# Conclusion

# Conclusion

- Merging information...
  - Is easier than sorting it
- Microprints:
  - Several semantic views of methods
  - Integrated in VW and CodeCrawler
  - Customizable and extensible

# Any Questions?

- Here's one:
- Which microprints would you need?