what

Towards Structural Decomposition of Reflection with Mirrors

who

Nick Papoulias
Noury Bouraqadi
Marcus Denker
Stephane Ducasse
Luc Fabresse

INRIA

Ecole d'Ingénieurs Centre de Recherche Mines de Douai LILLE EURORÉGION

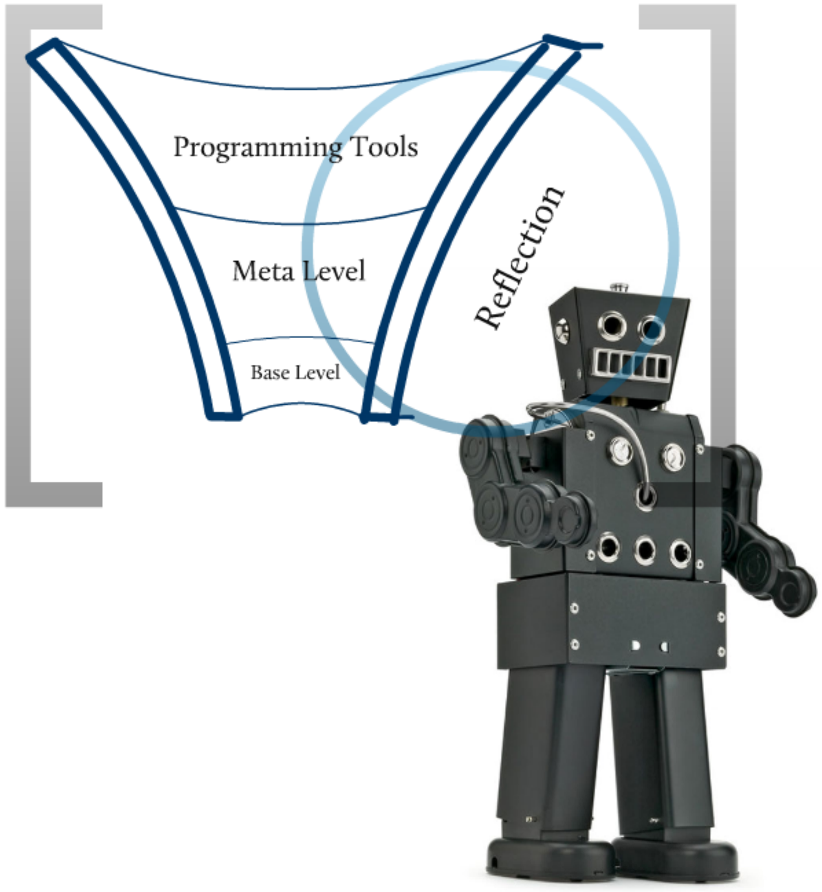
RMod

Université Lille 1 Sciences et Technologies

Prezi

why

- Reflection as a pluggable sub-system for:



- Remote Programming
- Resources
- Security
- Cleaner OO Design

Problems with Reflection

- Breaks encapsulation
- Mixed base and meta access
- Resource intentive
- Security / Integrity threat
- No OO design (supporting multiple implementations)

Smalltalk Example

Meta level functionality is indistinguishably mixed in program code.

```
myCar := Car new.
```

sign (supporting multiple
cations)

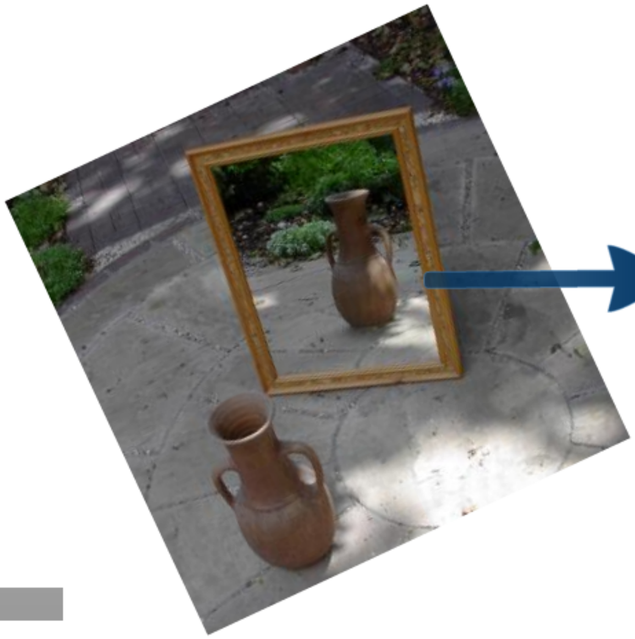
Smalltalk Example

Meta level functionality is indistinguishably mixed in program code.

```
myCar := Car new.
numberOfDoors := myCar numberOfDoors.
carClass := myCar class.
anotherCar := carClass new.
carSuperClass := carClass superclass.
```

Mirrors (Bracha & Unghar 2004)

```
numberOfDoors := myCar numberOfDoors.
carClass := myCar class.
anotherCar := carClass new.
carSuperClass := carClass superclass.
```



Mirrors (Bracha & Unghar 2004)

Functional decomposition of Reflection

- Encapsulation
- Stratification
- Ontological Correspondance

Smalltalk Example - with Mirrors

Meta level functionality is indistinguishably mixed in program code.

```
myCar := Car new.
numberOfDoors := myCar numberOfDoors.
carMirror := Mirror on: myCar.
carClassMirror := carMirror class.
```

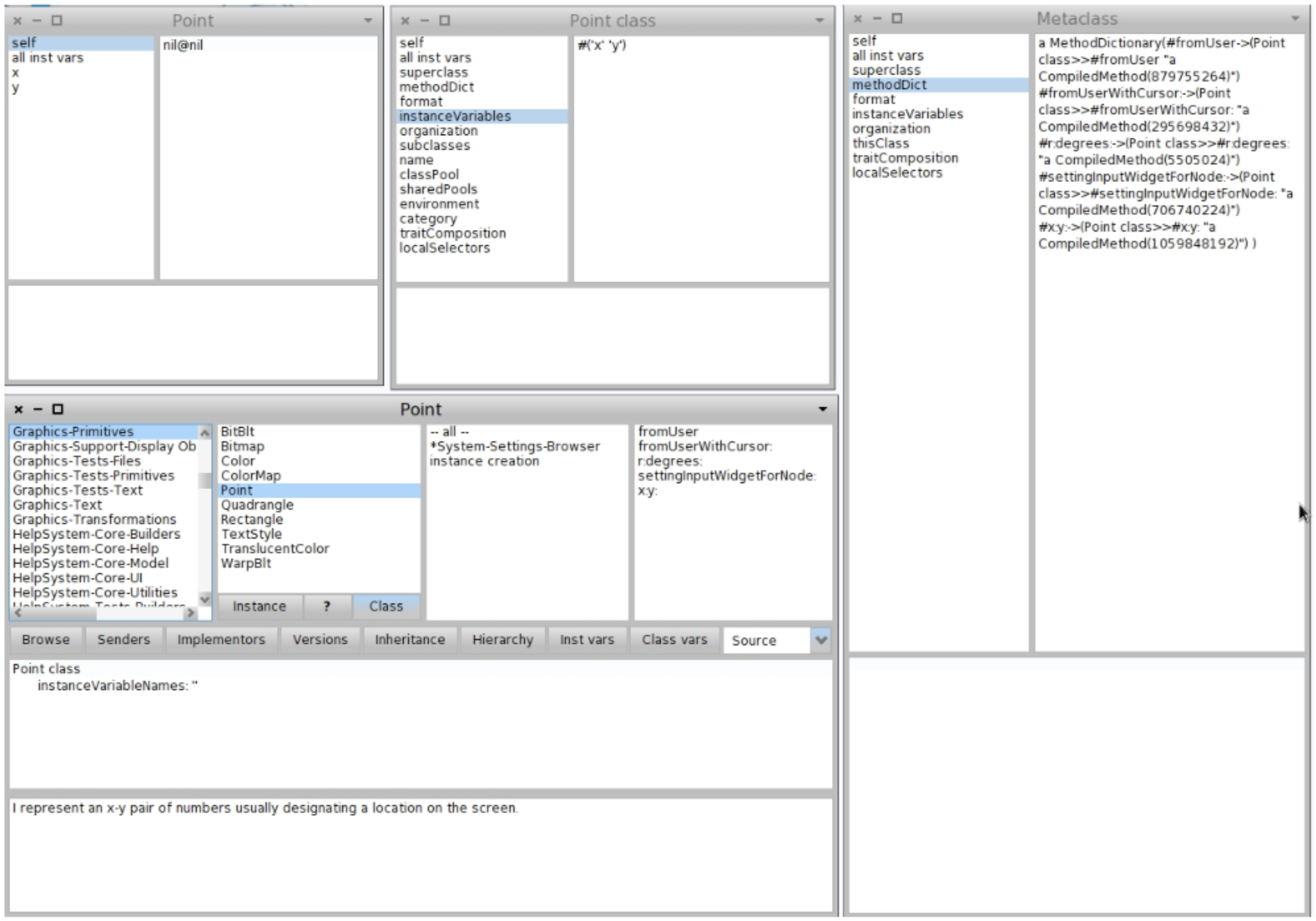
Ontological Correspondance

Smalltalk Example - with Mirrors

Meta level functionality is indistinguishably mixed in program code.

```
myCar := Car new.
numberOfDoors := myCar numberOfDoors.
carMirror := Mirror on: myCar.
carClassMirror := carMirror class.
carClassSuperMirror := carClassMirror superclass.
```

But what about meta-information ?



```
self
all inst vars
x
y
```

```
nil@nil
```

```
self
all inst vars
superclass
methodDict
format
instanceVariables
organization
subclasses
name
classPool
sharedPools
environment
category
traitComposition
localSelectors
```

```
#('x' 'y')
```

```
self
all inst vars
superclass
methodDict
format
instanceVariables
organization
thisClass
traitComposition
localSelectors
```

```
a MethodDictionary(#fromUser->(Point
class>>#fromUser "a
CompiledMethod(879755264)")
#fromUserWithCursor:->(Point
class>>#fromUserWithCursor: "a
CompiledMethod(295698432)")
#r:degrees:->(Point class>>#r:degrees:
"a CompiledMethod(5505024)")
#settingInputWidgetForNode:->(Point
class>>#settingInputWidgetForNode: "a
CompiledMethod(706740224)")
#x:y:->(Point class>>#x:y: "a
CompiledMethod(1059848192)") )
```

```
Graphics-Primitives
Graphics-Support-Display Ob
Graphics-Tests-Files
Graphics-Tests-Primitives
Graphics-Tests-Text
Graphics-Text
Graphics-Transformations
HelpSystem-Core-Builders
HelpSystem-Core-Help
HelpSystem-Core-Model
HelpSystem-Core-UI
HelpSystem-Core-Utilities
```

```
BitBlt
Bitmap
Color
ColorMap
Point
Quadrangle
Rectangle
TextStyle
TranslucentColor
WarpBlt
```

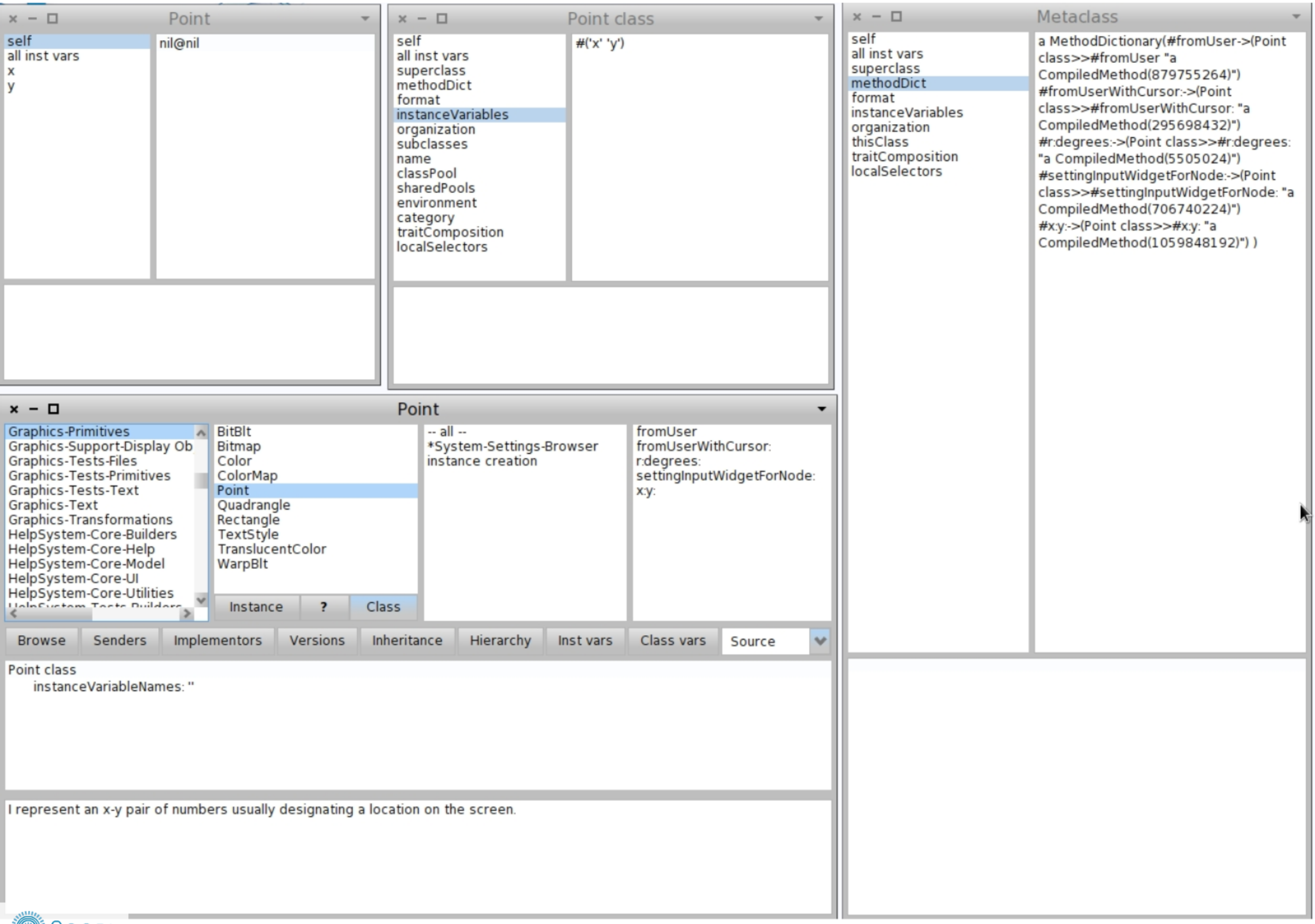
```
-- all --
*System-Settings-Browser
instance creation
```

```
fromUser
fromUserWithCursor:
r:degrees:
settingInputWidgetForNode:
x:y:
```

```
Point class
    instanceVariableNames: ''
```

I represent an x-y pair of numbers usually designating a location on the screen.

Prezi



```
a MethodDictionary(#fromUser->(Point class>>#fromUser "a CompiledMethod(879755264)") #fromUserWithCursor:->(Point class>>#fromUserWithCursor: "a CompiledMethod(295698432)") #r:degrees:->(Point class>>#r:degrees: "a CompiledMethod(5505024)") #settingInputWidgetForNode:->(Point class>>#settingInputWidgetForNode: "a CompiledMethod(706740224)") #x:y:->(Point class>>#x:y: "a CompiledMethod(1059848192)") )
```

```
Point class
    instanceVariableNames: ''
```

I represent an x-y pair of numbers usually designating a location on the screen.

Prezi

impact on:

- Encapsulation/Stratification
- Resources
- Security

Our goal

What is Base and what is Meta ?

There is a powerfull inherent ambiguity !

"With great power comes great responsibility.." --Uncle Ben--

or Voltaire !

Prezi

Our proposition

Our goal

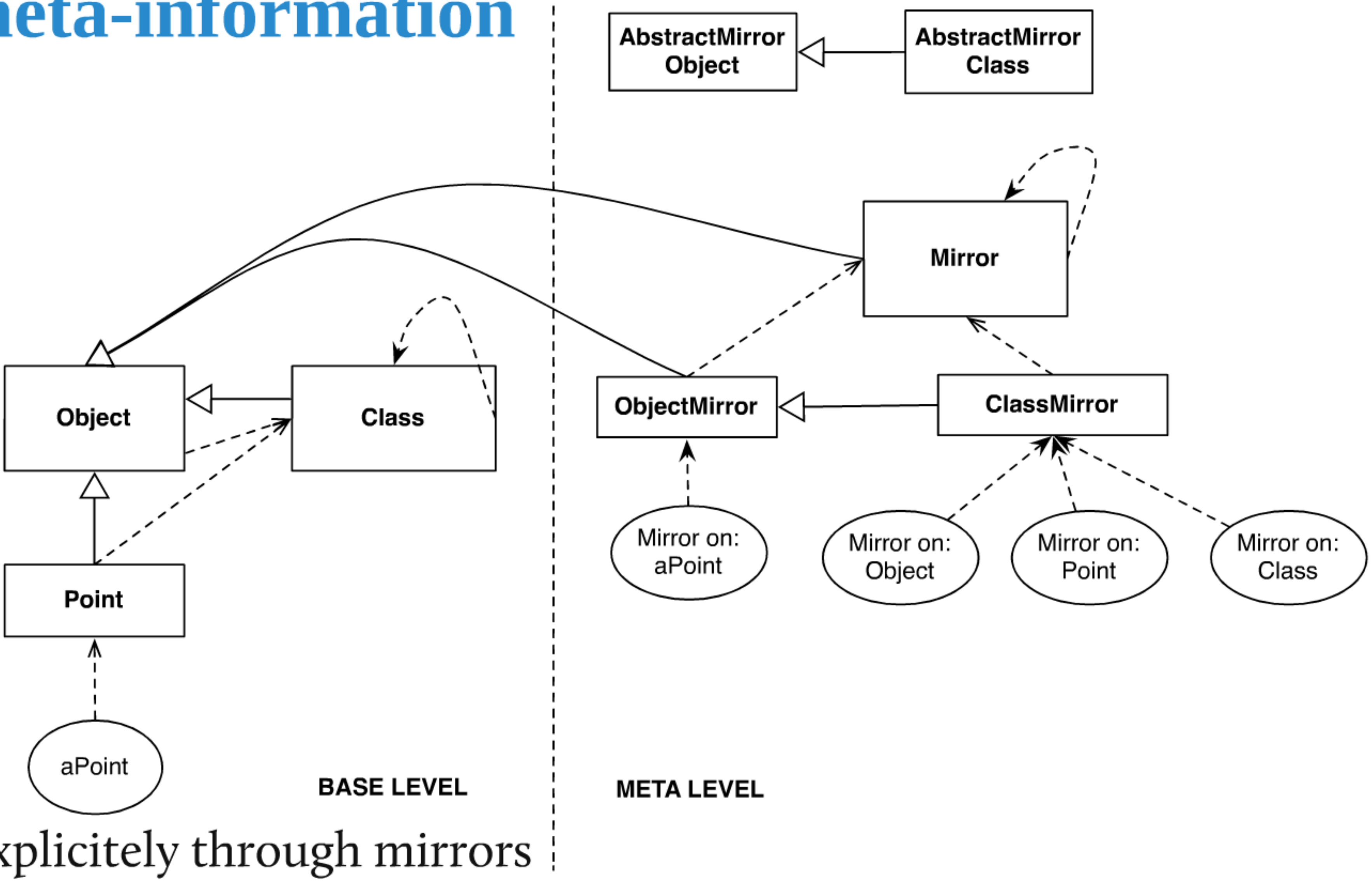
	Smalltalk	Classic mirror-based systems	Our goal
Reflective Functionality	FIX	DISCARDABLE	DISCARDABLE
Meta-Data	FIX	FIX	DISCARDABLE
Base-Level	FIX	FIX	FIX

all the power, without the responsibility

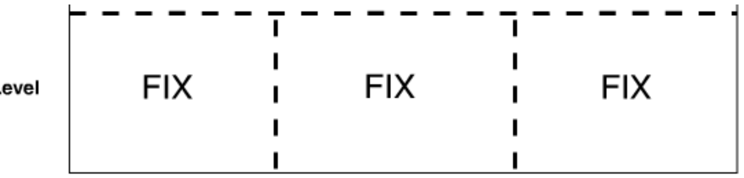
• V
whe

Our proposition

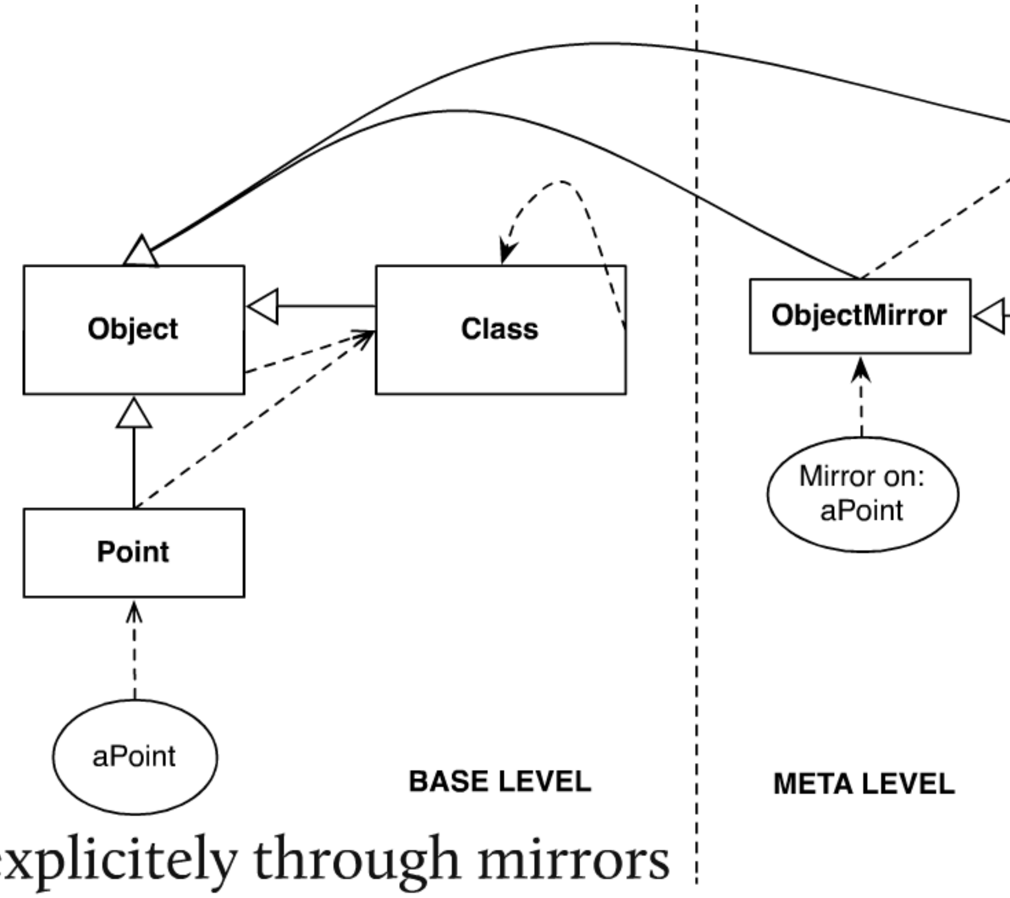
Mirrors as the storage entities of meta-information



ation
ection explicitely through mirrors
ur can only be done through mirrors



l the power, without
the responsibility



- every object has a mirror
- mirrors hold all meta-information
- all other entities provide reflection explicitely through mirrors
- dynamic addition of behaviour can only be done through mirrors

Validation & Prototype

MetaTalk

Whole system: VM, Compiler, Object Model implemented on top of Pharo.

Sound execution of base level functionality both:

- In the presence and
- In the absence of mirrors
- Validated complete stratification of meta-information

when they are not needed.

Related Work

- Bracha and Unghar 2004
- Lorenz and Vlissides 2003
- Declarative model for Smalltalk

eded.

Related Work

- Bracha and Unghar 2004
- Lorenz and Vlissides 2003
- Declarative model for Smalltalk
- Resilient
- Mirages and AmbientTalk

Future Work

... small

Ontological Correspondance....

- Metrics
- Behavioral Reflection
- Remote Programming (case studies)

Ontological Correspondance....

- Metrics
- Behavioral Reflection
- Remote Programming (case studies)

http://squeaksource/MetaTalk

Thank you !