Advanced Reflection in Pharo

Marcus Denker

http://rmod.lille.inria.fr

What you know…

- Smalltalk is reflective
- e.g. Classes and Methods are Objects
- Reflective API on all Objects

Instance Variables

2 instance variables

```
Object subclass: #Point
    instanceVariableNames: 'x y'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Kernel-BasicObjects'
```

- Ask the class:

```
Point instVarNames
```

- read:

```
3@4 instVarNamed: #x
```

- write:

```
3@4 instVarNamed: #x put: 5
```

Great!

But…

This is just a String!

```
Object subclass: #Point
    instanceVariableNames: 'x y'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Kernel-BasicObjects'
```

- returns an Array of Strings:

```
Point instVarNames
```

Why not Objects?

We can do better!

- All classes have a **Layout**

- Describes the **memory layout** defined by a class

- Layout and all the description are **Objects**

- `Point layout`
  - a Normal Object
- `Array layout`
  - an Array of Pointers
- `ByteArray layout`
  - an Array of Bytes

```
Point layout allSlots
```

```
==>  an OrderedCollection(x => Slot y => Slot)
```

- Slots know how to read values from Objects

```
mySlot := Point layout resolveSlot: #x.

mySlot read: 3@4.
```

Why?

Typed Slots

```
Slot subclass: #TypedSlot
    layout: PointerLayout
    slots: {#x => TypedSlot type: Integer}.


TypedSlot >> write: aValue to: anInstance
    (aValue isNil or: [aValue isKindOf: type])
        ifFalse: [ InvalidTypeError signal ].
    super write: aValue to: anInstance.
```

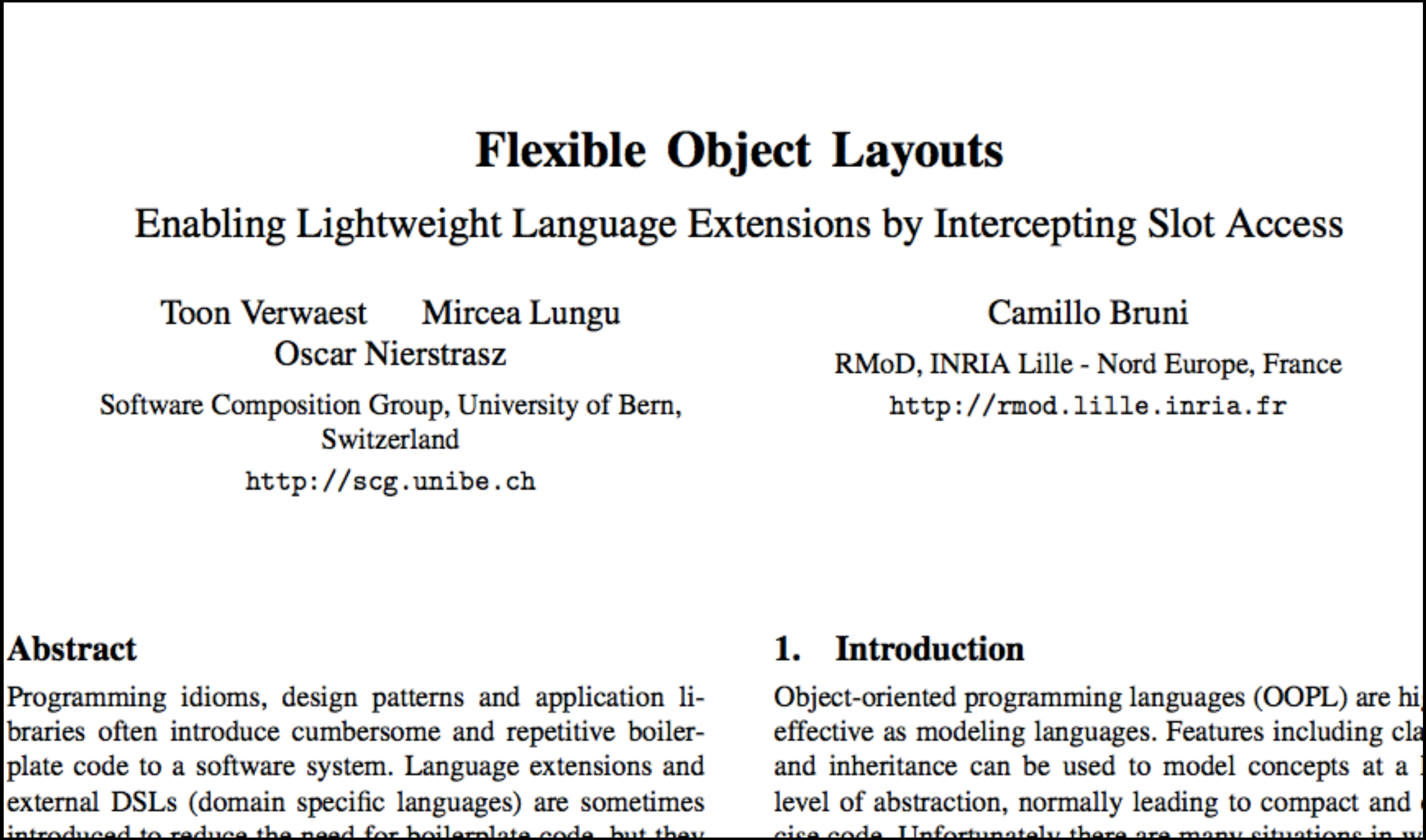
Property Slots

```
Object
    subclass: #PropertyObject
    layout: PointerLayout
    slots: {
        field     => Slot
        property1 => PropertySlot.
        property2 => PropertySlot.
        ...
        propertyN => PropertySlot.
    }
```

Others

- BitSlot
- BooleanSlot
- Alias
- Relationships (e.g. one-one, one-many)
- …. Your Domain level Slot!

More in Paper from OOPSLA



Flexible Object Layouts

Enabling Lightweight Language Extensions by Intercepting Slot Access

Toon Verwaest Mircea Lungu
Oscar Nierstrasz
Software Composition Group, University of Bern, Switzerland
http://scg.unibe.ch

Camillo Bruni
RMoD, INRIA Lille - Nord Europe, France
http://rmod.lille.inria.fr

Abstract

Programming idioms, design patterns and application libraries often introduce cumbersome and repetitive boilerplate code to a software system. Language extensions and external DSLs (domain specific languages) are sometimes

1. Introduction

Object-oriented programming languages (OOPL) are hi
effective as modeling languages. Features including cla
and inheritance can be used to model concepts at a
level of abstraction, normally leading to compact and

Status

- Slots are in Pharo3, but hidden
- In Pharo4: un-hide and introduce some Special Slots (e.g. Boolean, Property)
    - e.g. for Morphic (user interface objects)

Help Wanted!!!

So this worked well…

Lets do it again!

Turn another String into Objects

:-)

Methods

Lets have a look

- Method are Objects, but…
- No high-level model for sub-method elements
  - Message sends
  - Assignments
  - Variable access
- Structural reflection stops at the granularity of methods!

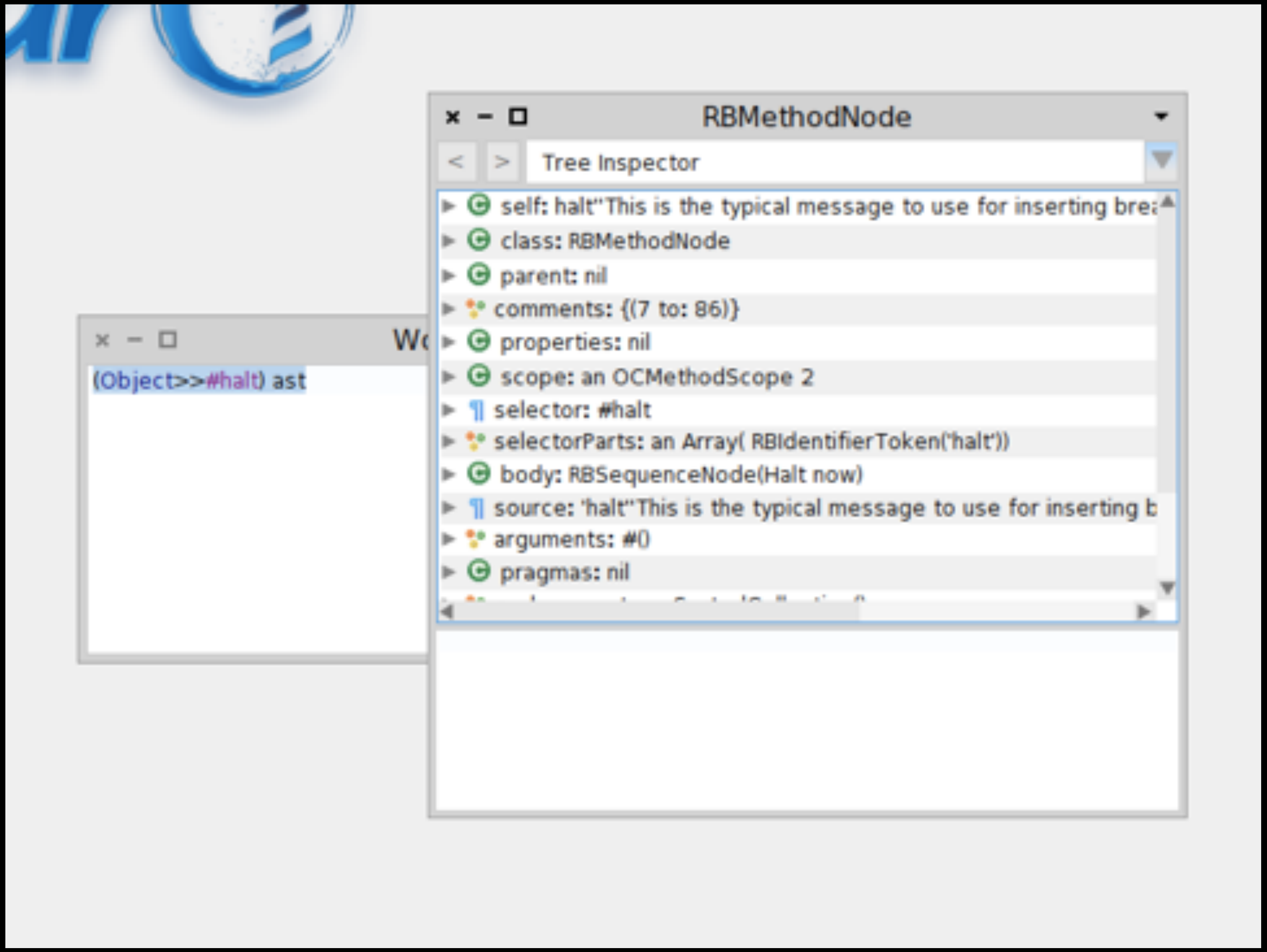
Can we do better?

Compilers have ASTs

Abstract **S**yntax **T**rees

- Lets have a look at an example

```
(Object>>#halt) ast
```



```
(Object>>#halt) ast
```

RBMethodNode

Tree Inspector

```
self: halt"This is the typical message to use for inserting brea
class: RBMethodNode
parent: nil
comments: {(7 to: 86)}
properties: nil
scope: an OCMethodScope 2
selector: #halt
selectorParts: an Array( RBIdentifierToken('halt'))
body: RBSequenceNode(Halt now)
source: 'halt"This is the typical message to use for inserting b
arguments: #()
pragmas: nil
```

- Encodes the method as a tree of node-objects
- Visitor Pattern
- Transformations
    - Refactoring tool uses this!

```
RBProgramNode
  RBDoItNode
  RBMethodNode
  RBReturnNode
  RBSequenceNode
  RBValueNode
       RBArrayNode
       RBAssignmentNode
       RBBlockNode
       RBCascadeNode
       RBLiteralNode
       RBMessageNode
       RBVariableNode
```

In Pharo3

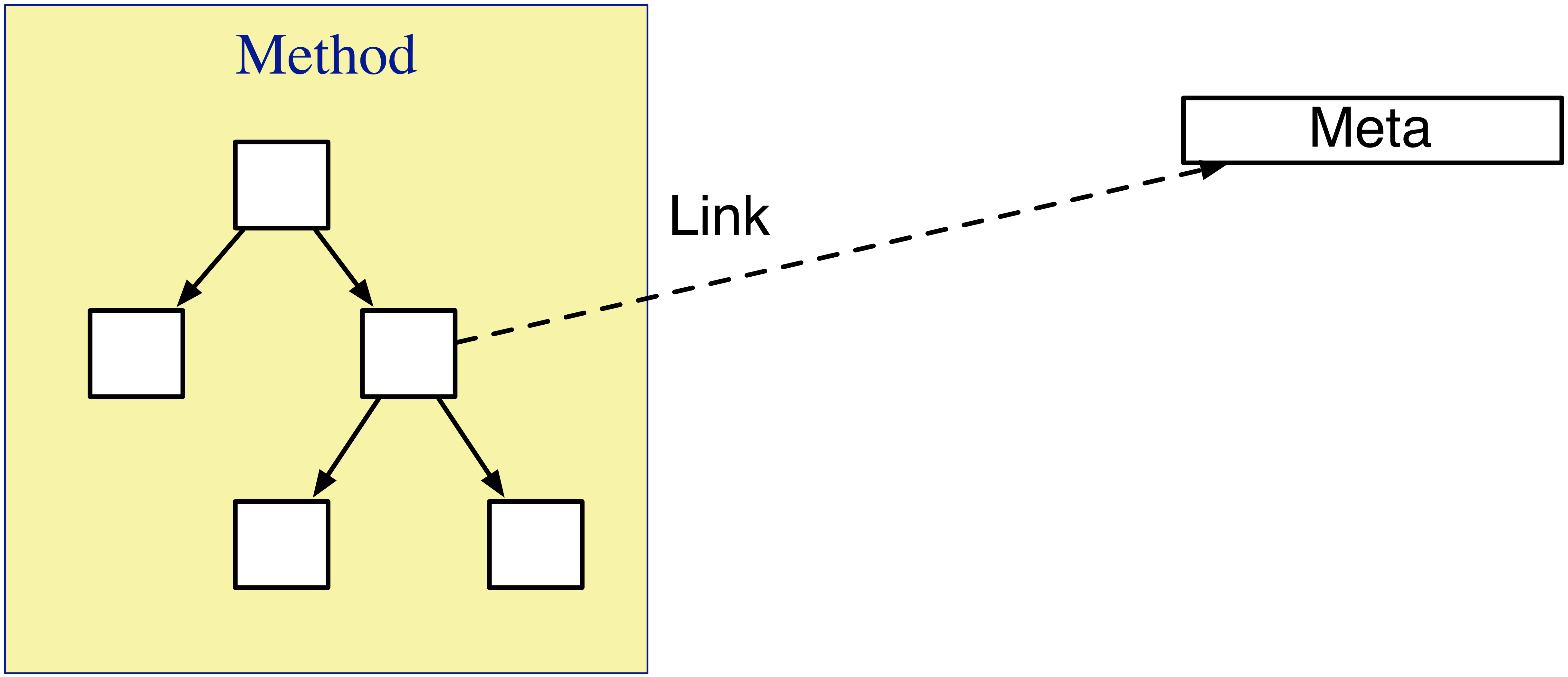
- AST based Navigation in the Editor
- “Suggestions”
- Debugger uses AST for pc->code mapping
- AST Interpreter for experiments

Future

- AST everywhere!
- Do we need to store strings?
- Can we have an AST based editor?
- Sub-Method Reflection: The MetaLink

Can we modify the behaviour of code?

- Annotate the AST with meta-links



Why?

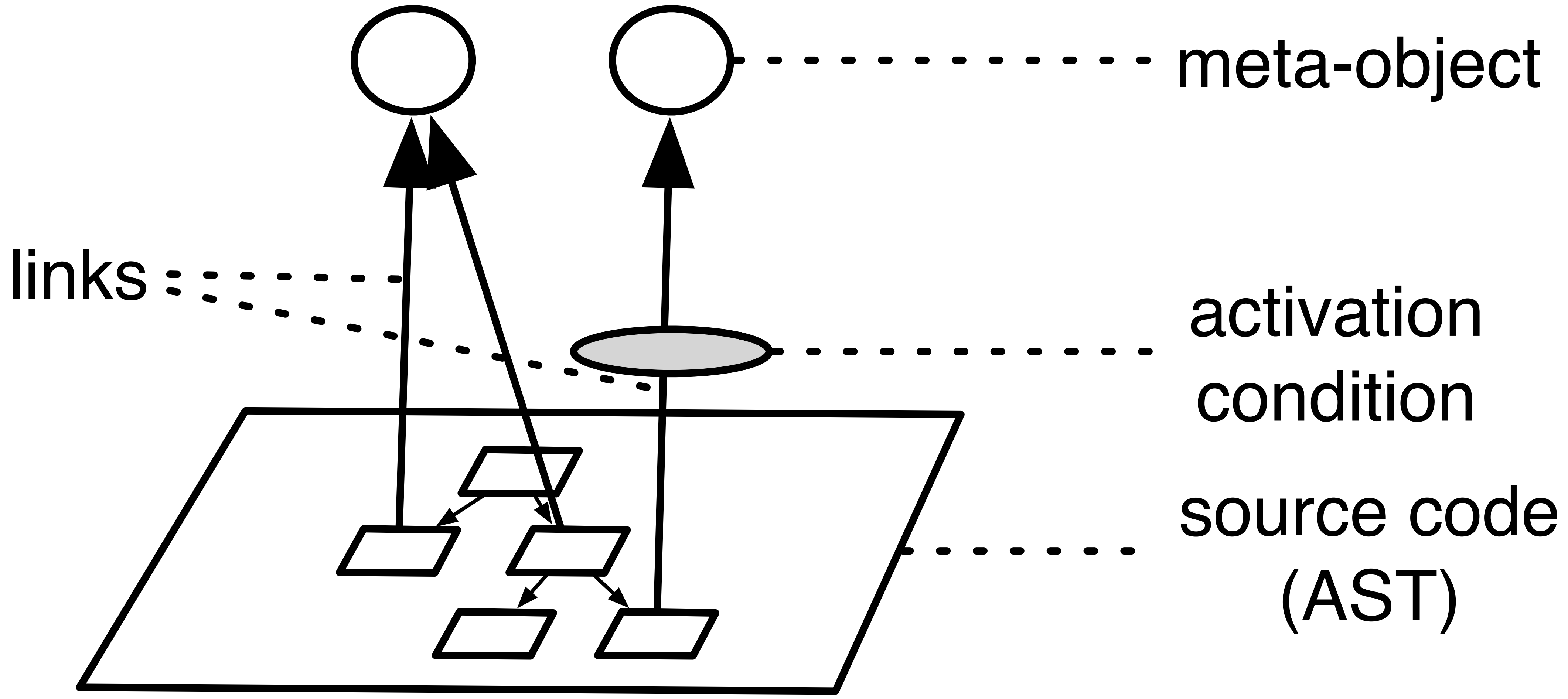
- Change behaviour for selected AST Nodes
- “All assignments”
- “this message send”

But without changing the program code!

Breakpoints

DEMO: Atoms

Behavioral Reflection



Uses…

- Debugger
    - BreakPoints, WatchPoints
- Profilers
- Coverage Analysis
- AOP

Will be in Pharo4

Will be in Pharo4

!!help wanted!!

What did we see?

- **Slots**
  - Instance variables are just described with strings
  - We can do better! Layout, Slots
- **ASTs Everywhere**
  - Methods are objects, but internal structure not modelled
  - We can do better! AST, Sub-Method Reflection, Meta-Links

Questions???