Pharo4
Plans and Dreams

Marcus Denker

http://rmod.lille.inria.fr

A bit early…

- We are hard working to get Pharo3 out
- Not much yet happened with planning Pharo4

1 Year

March 2014-December 2014

9 Months

+Time for Bug fixing

This is not a lot!

So not too many dreams…

It needs to be doable

Ideas

- Slots
- Reflectivity
- One-File Pharo
- Tools
- GIT
- Athens
- Bootstrap
- Sista

Bootstrap

- Create an image from a git repository
- Control what the image contains
- Easier to make changes
- Enforces Modularity

Boostrap

- Working for Pharo3 as a prototype
- Can we even use this for Pharo4 on the build server?

One File Pharo

- .sources, .changes. .image
- It is time to simplify that!

Epicea

- Replace .changes
- High level model:
  - aggregate changes (refactoring)
  - serialized to disk independent of source model

Epicea Log Browser

Prior+Trigger view

Expand all

Event

more

Undo 24/1/2014 10:34

ExampleClass >> #fortyTwo 24/1/2014 10:34

Undo 24/1/2014 11:06

ExampleClass >> #fortyTwo 24/1/2014 11:06

#fortyTwo ---> #forty2 24/1/2014 11:07

ExampleClass >> #forty2 24/1/2014 11:07 This is a comment

ExampleClass >> #fortyThree 24/1/2014 11:07

ExampleClass >> #fortyTwo 24/1/2014 11:07

'I didn''t like the name' 24/1/2014 11:08

'' 24/1/2014 11:09

'''comment '' trim' 24/1/2014 12:52

EpEntryItem >> #displayWidget 24/1/2014 12:53

MC save: Epicea-MartinDias.470 on: http://smalltalkhub.com/mc/MartinDias/Epicea/main/ 24/1/2014 12:55

Snapshot: /Users/tinchodias/Downloads/Epicea/Epicea-1.image 24/1/2014 12:55

'312312321' 24/1/2014 13:41

Content Filters

```
"protocol: #example"

fortyThree  ^ self fortyTwo + 1
```

```
"protocol: #example"

fortyThree  ^ self forty2 + 1
```

Step2: Sources

- It is 2014: Memory is cheap.
- Complexity is expensive
- Why not just put the sources in the image?
  - Just current version (compressed, of course)
  - Code history is in Monticello (or Git)

Slots

- First class Instance Variables
- Already in Pharo3, but compatible (ivar Slot)
- For Pharo4: Provide different Slot kinds

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 introduced to reduce the need for boilerplate code, but they

1. Introduction

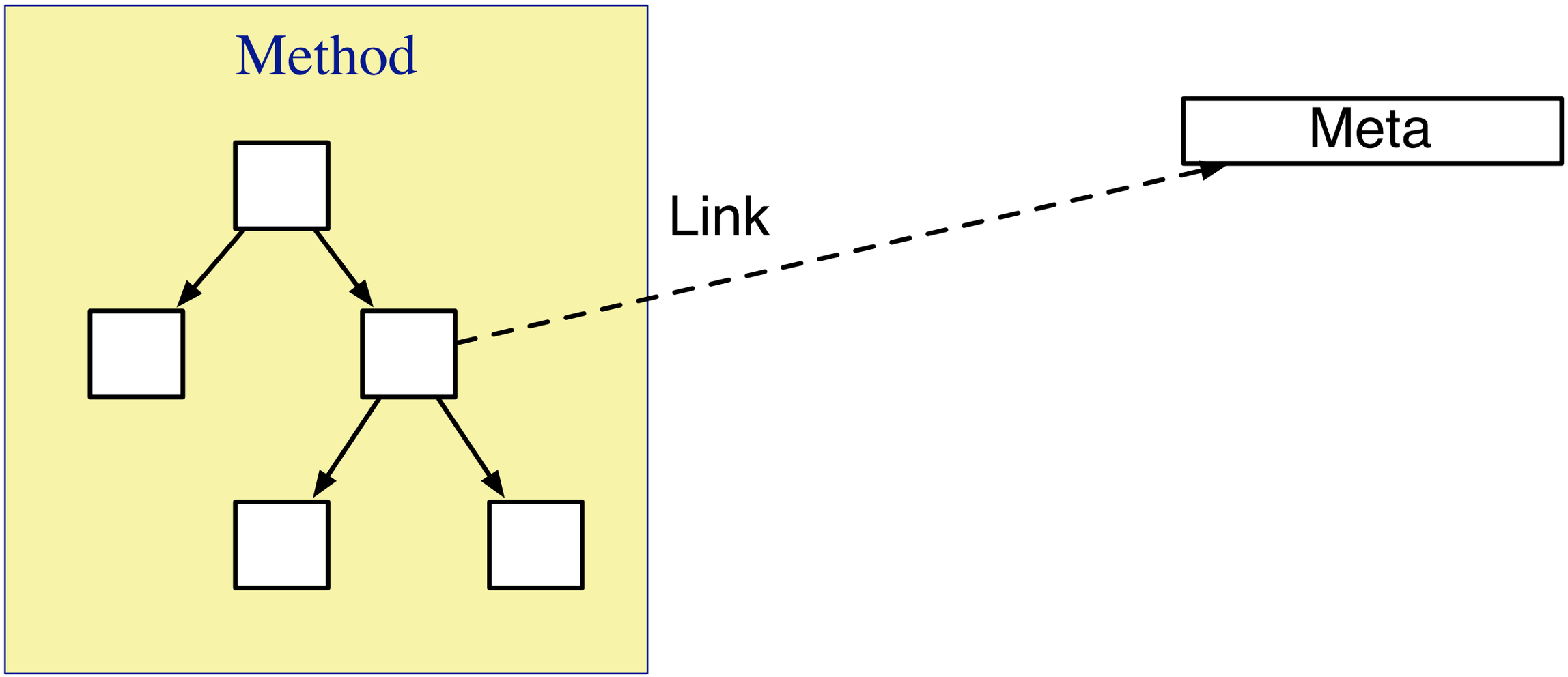
Object-oriented programming languages (OOPL) are hi
effective as modeling languages. Features including cla
and inheritance can be used to model concepts at a
level of abstraction, normally leading to compact and
cise code. Unfortunately there are many situations in w

Reflectivity

- Partial Behavioral Reflection
- Associate MetaObject with structural object
  - e.g. Slots
  - AST nodes

Can we modify the behaviour of code?

> Annotate the AST with meta-links

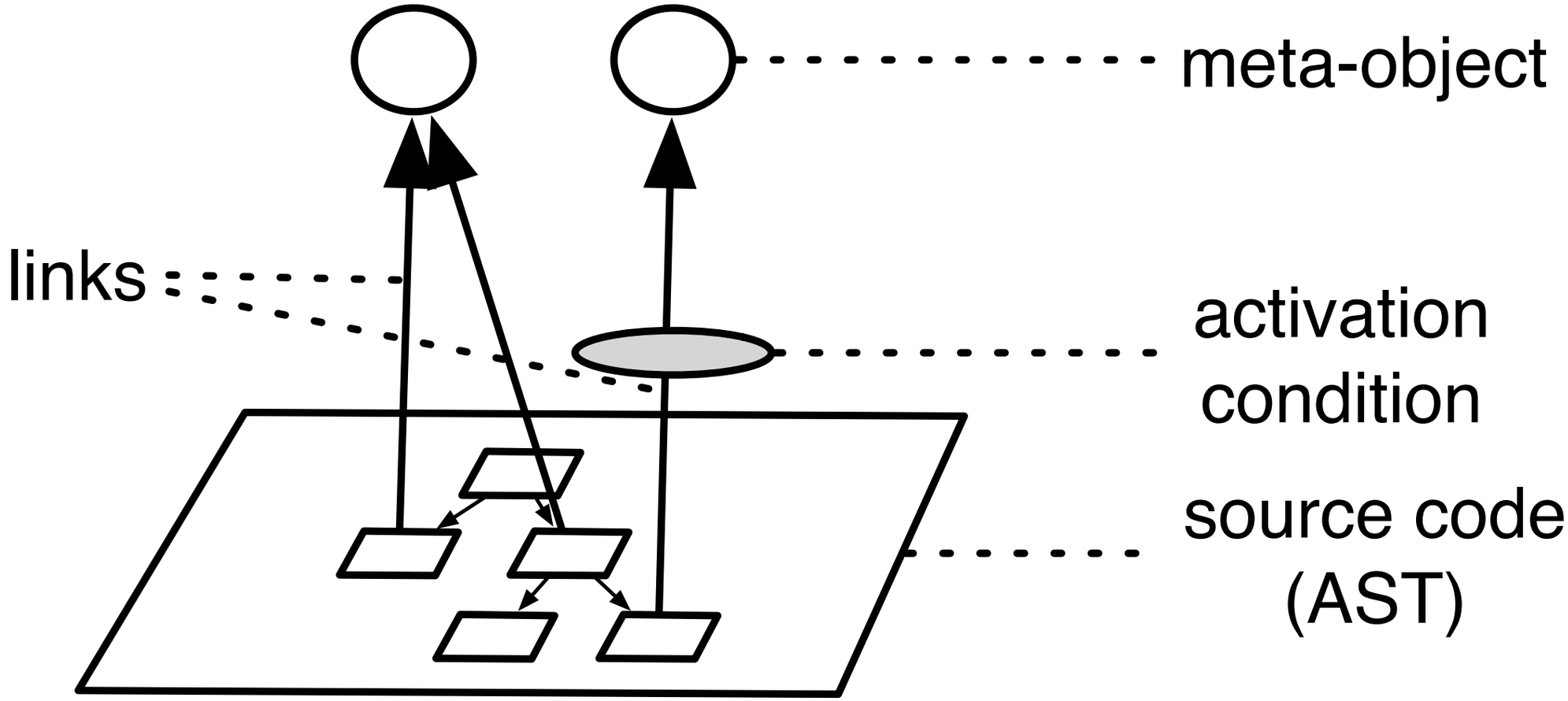


Why?

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

But without changing the program code!

Behavioral Reflection



Uses...

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

… And Beyond

- Every year one Release
- Research happens in Parallel
  - Lots of Interesting Stuff
  - Sadly another talk

Questions ?