Reflection in Pharo5
ESUG 2015
Marcus Denker
http://www.pharo.org
Everything is an Object
Everything?
Classes, yes.
Methods, yes
But Code?
Code is a String!
AST: Abstract Syntax Tree
AST in Pharo5

- AST of the Refactoring browser
  - Transformation
  - Visitors
  - Annotations (properties)
- Deeper integrated:
  - Pretty Printing, Syntax Highlight, Suggestions
  - Compiler uses RB AST
• Easy access

• #ast

• Demo: method and block
(OrderedCollection>>#do:) ast.

[ 1 + 2 ] sourceNode == thisContext method ast blockNodes first

- ASTCache: as twice, get the same answer (flushed on image save for now)
AST + Tools

// Override the superclass for performance reasons.

do: aBlock
    "Override the superclass for performance reasons."

firstIndex to: lastIndex do: [ :index |
aBlock value: (array at: index) ]
Opal Compiler

- Uses RB AST
- Based on Visitors

Text → AST → AST + vars → IR → CM

Parser → Semantic Analysis → AST Translator+IRBuilder → BytecodeBuilder+Encoder
Opal: API

- All staged are Pluggable
  - e.g, Semantic Analyzer or Code Generator can be changed.
- compiler options
Opal: Bytecode editing

- IR can be used to manipulate methods on a bytecode level
Too complicated
Too low level
Can we do better?
AST Meta Annotation

- We have an AST with properties
- We have Opal with Pluggable API
Can’t we use that?
Basis: the Evil Twin

CompiledMethod

Know each other

ReflectiveMethod

Bytecode

AST
Basis: the Evil Twin

run: aSelector with: anArray in: aReceiver
self installCompiledMethod.
self recompileAST.
self installCompiledMethod.
^compiledMethod
  valueWithReceiver: aReceiver
  arguments: anArray
Demo: Morph

- Morph methods do: #createTwin
- Morph methods do: #invalidate
- inspect “Morph methods”
Putting it together

• Annotate the AST
• Create Twin if needed
• Invalidate method
• Next call: generate code changed by annotation
recompileAST
  ast compilationContext
    semanticAnalyzerClass: RFSemanticAnalyzer;
    astTranslatorClass: RFASTTranslator.
  ast doSemanticAnalysis. "force semantic analysis"
  compiledMethod := ast generate: compiledMethod trailer.
  compiledMethod reflectiveMethod: self.
Annotations?
MetaLink
DEMO: Simple Link

node := (ReflectivityExamples>>#exampleMethod) ast.
link := MetaLink
    new metaObject: (Object new);
selector: #halt.

node link: link.

ReflectivityExamples new exampleMethod
Meta Link

• When setting link:
  • create twin if needed
  • install reflective method

• On execution
  • generate code and execute, install CM
Twin Switch

CompiledMethod

Know each other

ReflectiveMethod

Bytecode

AST
Link: metaobject

The object to send a message to

link := MetaLink new
    metaObject: [self halt]
Link: selector

The selector to send

```ruby
link := MetaLink new

.....

selector: #value
```
Link: control

before, after, instead

link := MetaLink new

.....

control: #after
Link: control

after: #ensure: wrap

link := MetaLink new

    control: #after
Link: control

instead: last link wins
(for now no AOP around)

link := MetaLink new

.....

control: #instead
Link: condition

boolean or block

link := MetaLink new

.....

condition: [self someCheck]
Link: arguments

what to pass to the meta?
Reifications

- Every operation has data that it works on
- Send: #arguments, #receiver, #selector
- Assignment: #newValue, #name
- All: #node, #object, #context
Link: arguments

what to pass to the meta?

link := MetaLink new

.....

arguments: #(name newValue)
Reifications: condition

link := MetaLink new
condition: [: object | object == 1];
Virtual meta

- Reifications can be the meta object

```plaintext
link := MetaLink new
  metaObject: #receiver;
  selector: #perform:withArguments:;
  arguments: #(selector arguments).
```
Statement Coverage

link := MetaLink new
    metaObject: #node;
    selector: #tagExecuted.

“set this link on all the AST nodes"
(ReflectivityExamples>>#exampleMethod) ast
    nodesDo: [:node | node link: link].
Variables

• Helper methods

  Point assignmentNodes

• But: can’t we annotate variables directly?
Friday: Slots+Globals
RoadMap

- Pharo4: Opal is default
- Pharo5
  - Remove old Compiler/AST
  - Reflectivity: First finished version
- Pharo6: Object specific links
Users

- Tools of ObjectProfile are being ported
- BreakPoints Pharo5
- Coverage Kernel by Pavel
- ....
Thanks!

• Work of many people…

• Too many to list here. (And I would forget for sure someone)
Questions ?