PHANtom: A Modern Aspect Language for Pharo

Johan Fabry, Daniel Galdames. PLEIAD Lab, DCC - Universidad de Chile
AOP ??

Cross-cutting concerns

Aspect = module

- Behavior = WHAT = advice
- Quantification = WHEN = pointcuts

Running application = stream of join points

- ST computation = messages
- In PHANtom: method exec
AOP ??

Cross-cutting concerns

Aspect = module

‣ Behavior = WHAT = advice
‣ Quantification = WHEN = pointcuts

Running application = stream of join points

‣ ST computation = messages
‣ In PHANtom: **method exec**
AOP ??

Cross-cutting concerns

Aspect = module

- Behavior = WHAT = advice
- Quantification = WHEN = pointcuts

Running application = stream of join points

- ST computation = messages
- In PHANtom: method exec
Cross-cutting concerns

Aspect = module

- Behavior = WHAT = advice
- Quantification = WHEN = pointcuts

Running application = stream of join points

- ST computation = messages
- In PHANtom: method exec
Cross-cutting concerns

Aspect = module

‣ Behavior = WHAT = advice
‣ Quantification = WHEN = pointcuts

Running application = stream of join points

‣ ST computation = messages
‣ In PHANtom: method exec
In the spirit of Smalltalk

Dynamic

Simple

Powerful
In the spirit of Smalltalk

Dynamic

Dynamic AOP

Simple

Powerful
In the spirit of Smalltalk

Dynamic

Dynamic AOP

Develop normally

Simple

Powerful
In the spirit of Smalltalk

Dynamic

Dynamic AOP

Develop normally

Simple

Minimal set of Constructs

Powerful
In the spirit of Smalltalk

- Dynamic
- Simple: Minimal set of Constructs, Uniform / no restrictions
- Powerful: Dynamic AOP, Develop normally
In the spirit of Smalltalk

- Dynamic
  - Dynamic AOP
- Simple
  - Minimal set of Constructs
  - Uniform / no restrictions
- Powerful
  - First-class everything

Develop normally
In the spirit of Smalltalk

Dynamic

Dynamic AOP
Develop normally

Simple
Minimal set of Constructs
Uniform / no restrictions

Powerful
First-class everything
New language features
Pointcuts
Pointcuts

PhPointcut receivers: 'TestCase'
selectors: 'assert:'

Dyn
Sim
Pow
Pointcuts

PhPointcut receivers: 'TestCase'
selectors: 'assert:'

PhPointcut receivers: 'TestCase+'
selectors: #(assert: assert:equals:)

Dyn

Sim

Pow
PhPointcut receivers: 'TestCase'
selectors: 'assert:'

PhPointcut receivers: 'TestCase+'
selectors: #(assert: assert:equals:)
context: #(receiver sender arguments)
Pointcuts

PhPointcut receivers: 'TestCase'
selectors: 'assert:'

PhPointcut receivers: 'TestCase+'
selectors: #(assert: assert:equals:)
context: #(receiver sender arguments)

PhPointcut receivers: 'Test*'
selectors: #(assert: assert:_:
          assert:_:_:)
context: #(receiver)
restrict: #(Tests_Mine)
Advice
PhAdvice pointcut: pc
   send: #incCount: to: self
   type: #after
Advice

PhAdvice pointcut: pc
  send: #incCount: to: self
  type: #after

PhAdvice pointcut: pc
  advice: [:ctx |
    Transcript show:
    (ctx receiver asString);cr.
    ctx proceed.]]
  type: #around.
Inter-Type Declaration
PhClassModifier on: ACTestCase
    addIV: 'phacount'
Inter-Type Declaration

PhClassModifier on: ACTestCase
  addIV: 'phacount'

PhClassModifier on: ACTestCase
  addIM: 'phacount
  ^phacount ifNil: [phacount := 0]'
Aspect

PHAspect subclass: MyAspect ...

(Object subclass: #PhAspect ...)

Add PHAdvice and PHClassModifier instances

Send install to activate

Send uninstall to deactivate
What happens here?

|asp|
asp := PhAspect new add:
  (PhAdvice
    pointcut: (PhPointcut
      receivers: 'Transcript class'
      selectors: 'show:('
    advice: [ Transcript show: 'Have ']
    type: #before).
asp install.
Transcript show: 'reentrancy control'; cr
What happens here?

\begin{verbatim}
|asp|
asp := PhAspect new add:
(PhAdvice
 pointcut: (PhPointcut
     receivers: 'Transcript class'
     selectors: 'show:')</n   advice: [ Transcript show: 'Have ']
 type: #before).
asp install.
Transcript show: 'reentrancy control'; cr
\end{verbatim}
What happens here?

```plaintext
|asp|
asp := PhAspect new add:
   (PhAdvice
      pointcut: (PhPointcut
         receivers: 'Transcript class'
         selectors: 'show:'),
      advice: [ Transcript show: 'Have ']
      type: #before).
asp install.
Transcript show: 'reentrancy control'; cr
```
What happens here?

\[
\text{asp} := \text{PhAspect new add: (PhAdvice}
\]

pointcut: (PhPointcut

receivers: 'Transcript class'
selectors: 'show: ')

advice: [ Transcript show: 'Have ']
type: #before).

asp install.

Transcript show: 'reentrancy control'; cr
What happens here?

|asp|

asp := PhAspect new add:
  (PhAdvice
    pointcut: (PhPointcut
      receivers: 'Transcript class'
      selectors: 'show:')
    advice: [ Transcript show: 'Have '
    type: #before).
  asp install.
Transcript show: 'reentrancy control'; cr
What happens here?

\texttt{|asp|}

\texttt{asp := PhAspect new add:}
\texttt{(PhAdvice}
\texttt{   pointcut: (PhPointcut}
\texttt{     receivers: 'Transcript class'
\texttt{       selectors: 'show:}')
\texttt{   advice: ['Transcript show: 'Have ']}
\texttt{   type: #before).
\texttt{asp install.}
\texttt{Transcript show: 'reentrancy control'; cr}
What happens here?

\[
\begin{align*}
| & asp | \\
asp & := \text{PhAspect} \text{ new add:} \\
(\text{PhAdvice} \\
\quad \text{pointcut: (PhPointcut} \\
\qquad \text{receivers: 'Transcript class'} \\
\qquad \text{selectors: 'show:'}) \\
\quad \text{advice: [Transcript show: 'Have ']} \\
\quad \text{type: #before}). \\
\text{asp install.} \\
\text{Transcript show: 'reentrancy control'; cr}
\end{align*}
\]
What happens here?

|asp|
asp := PhAspect new add:
    (PhAdvice
        pointcut: (PhPointcut
            receivers: 'Transcript class'
            selectors: 'show:')
        advice: [ Transcript show: 'Have ' ]
        type: #before).
asp install.
Transcript show: 'reentrancy control'; cr

BOOM
Reentrancy control

PHANtom: no problem!
  › Aspect execution by default NOT MATCHED

More power if you want it: Membranes [Tanter & Tabareau]

Use membranes to control
  › join point emission and capture scope
  › aspect observing join points
Advice ordering

Multiple advice 1 join point

“Classic”: advice precedence
  ‣ Global and static

PHANtom deployment precedence
  ‣ Global and ‘static’
  ‣ Complement or override on pointcut

PHANtom dynamic precedence in the advice
  ‣ `OrderedCollection` of `PHAdvice` instances
  ‣ get from context
  ‣ set on context
Not in the paper ...

TDD

- 155 tests
- 90+% coverage
TDD

- 155 tests
- 90+% coverage

Figura 15: Test coverage con Hapao
Future work

Optimize the implementation

Compile PHANtom code

Infrastructure for Domain-Specific Aspect Languages
http://pleiad.cl/PHANptom