PHANtom:
A Modern Aspect Language for Pharo

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

Pleiad

Programming Languages and Environments for Intelligent, Adaptable and Distributed systems

AOP ??

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

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

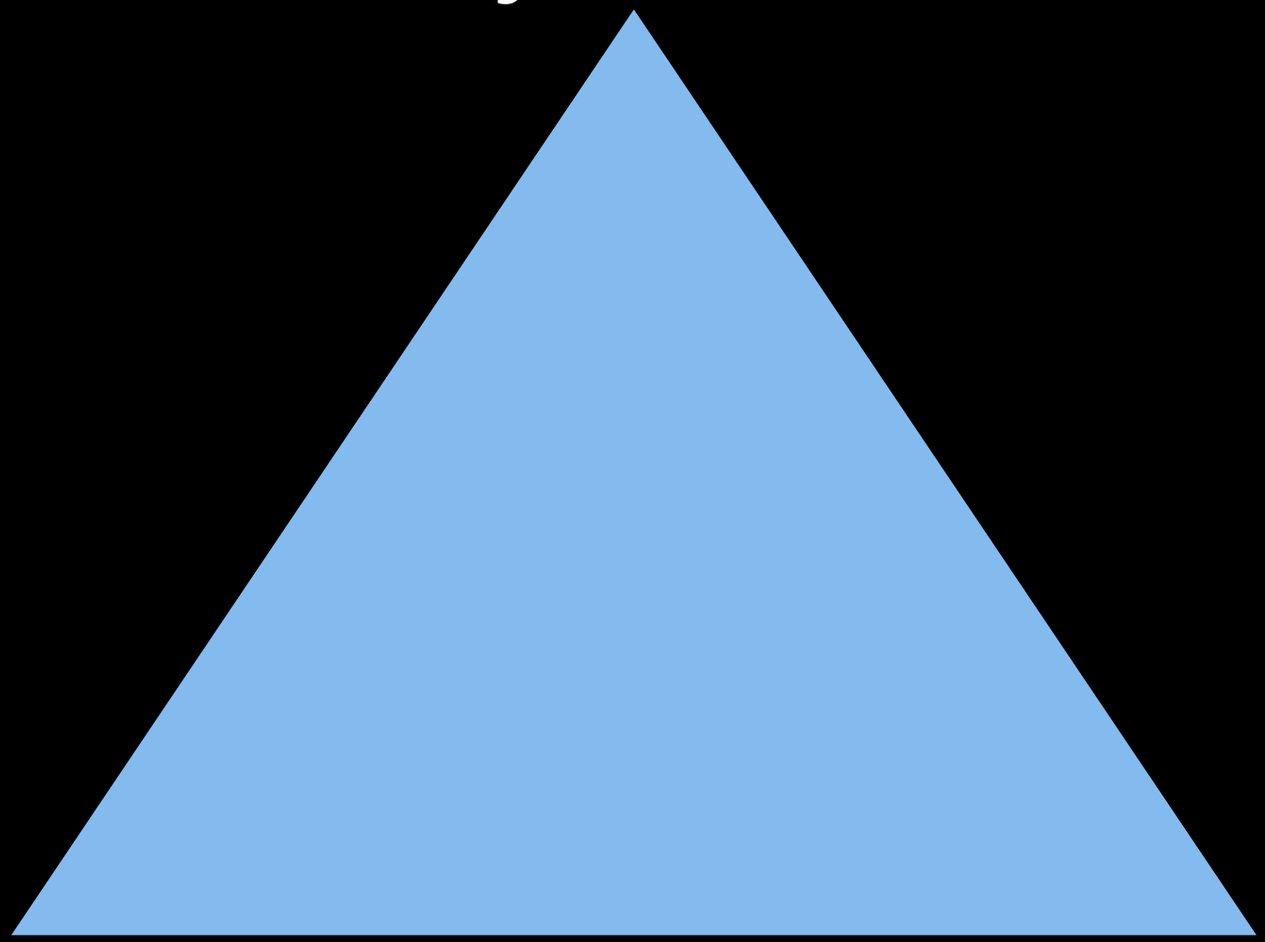
- Behavior = WHAT = advice
- Quantification = WHEN = pointcuts

Running application = stream of join points

- ST computation = messages
- In PHANtom: method exec

In the spirit of Smalltalk

Dynamic



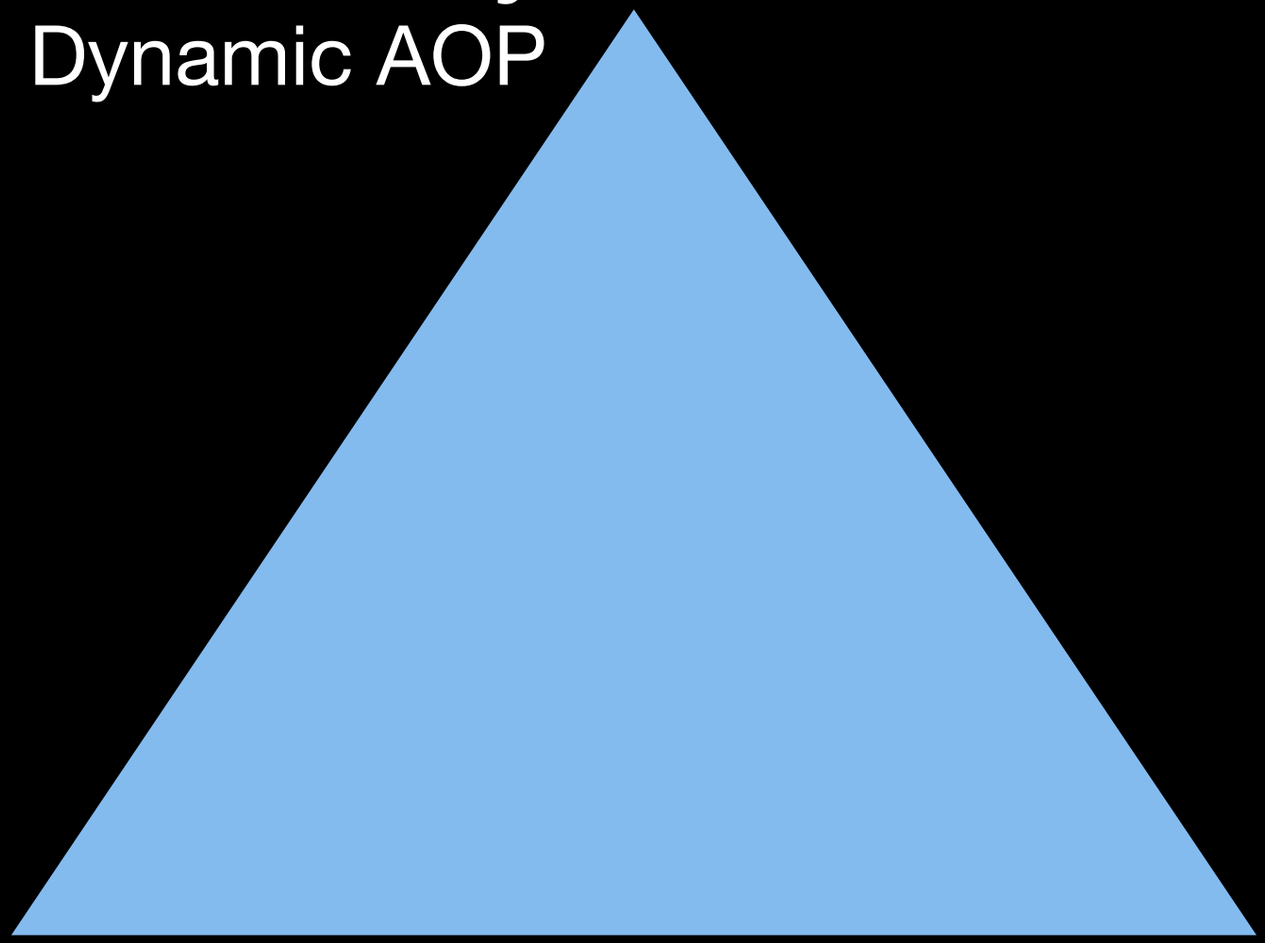
Simple

Powerful

In the spirit of Smalltalk

Dynamic

Dynamic AOP



Simple

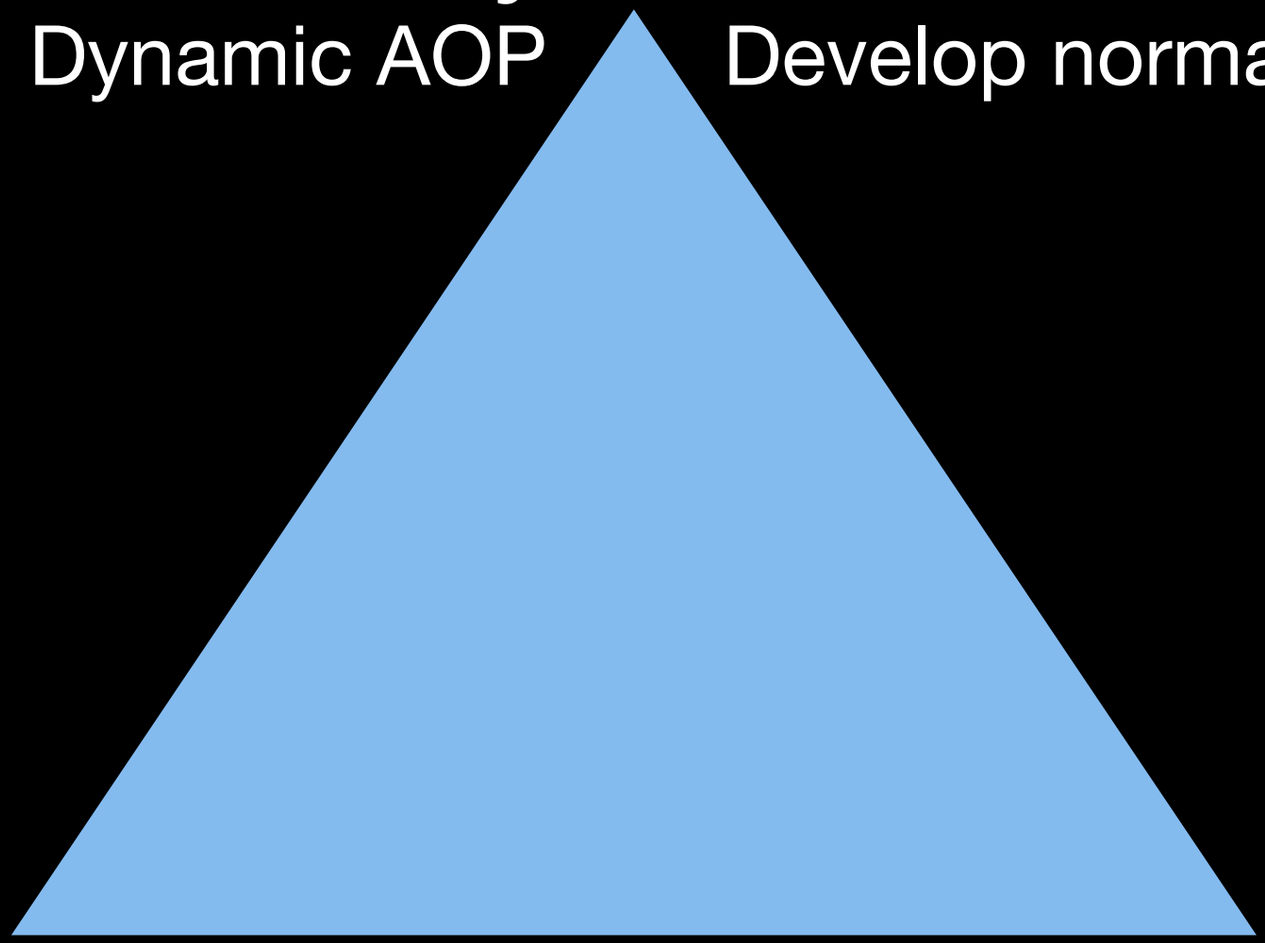
Powerful

In the spirit of Smalltalk

Dynamic

Dynamic AOP

Develop normally



Simple

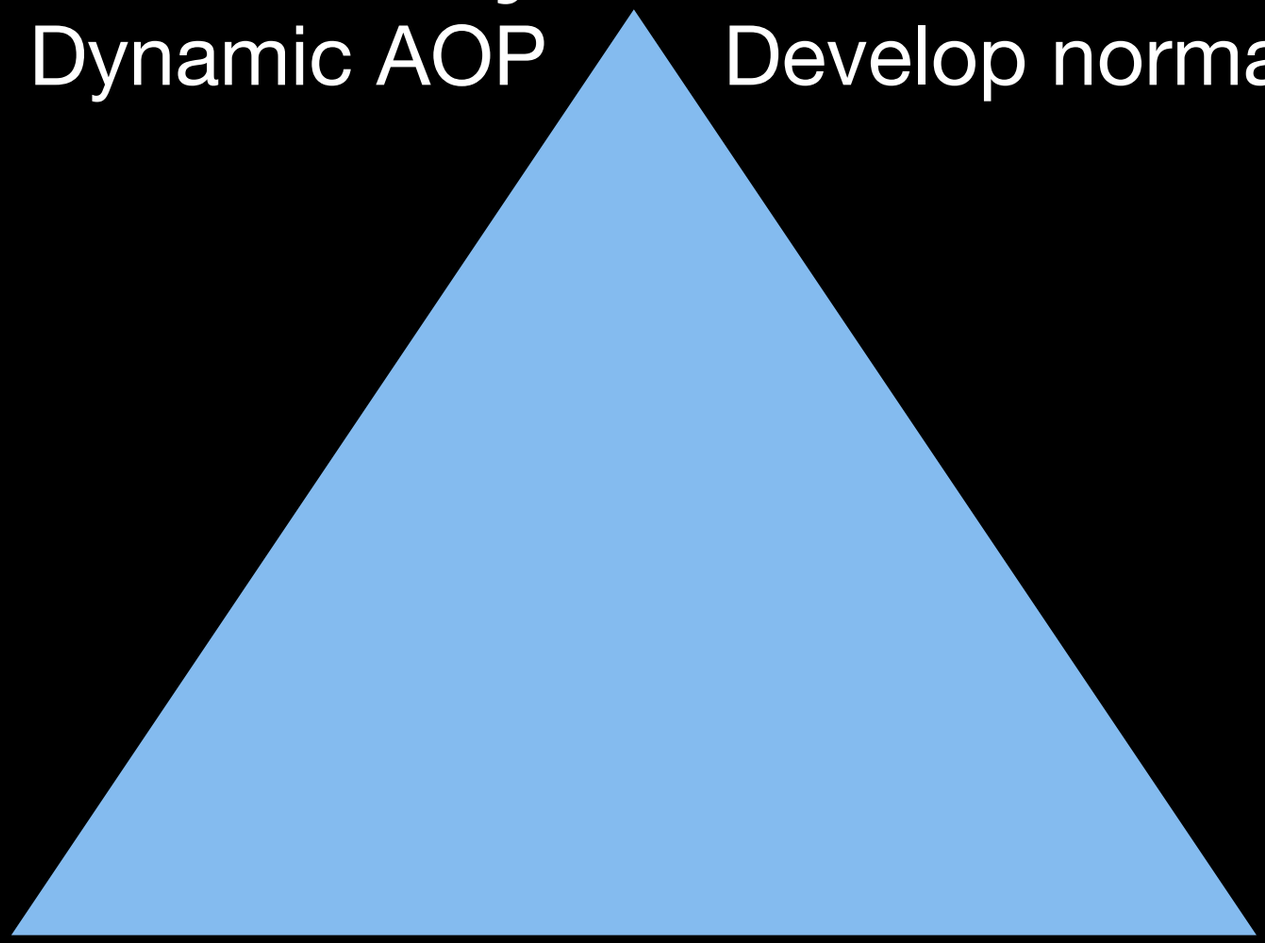
Powerful

In the spirit of Smalltalk

Dynamic

Dynamic AOP

Develop normally



Simple

Minimal set of Constructs

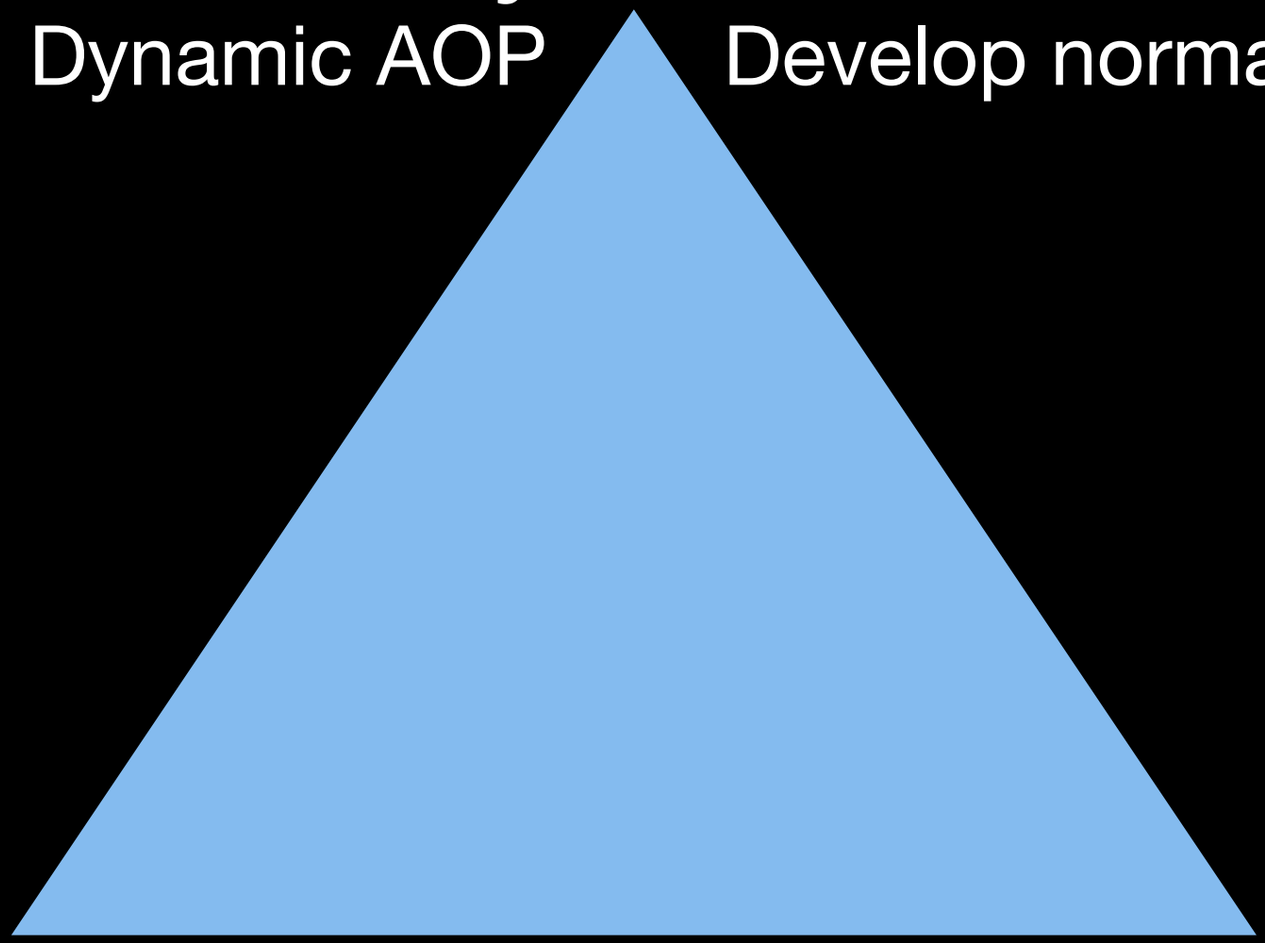
Powerful

In the spirit of Smalltalk

Dynamic

Dynamic AOP

Develop normally



Simple

Minimal set of Constructs

Uniform / no restrictions

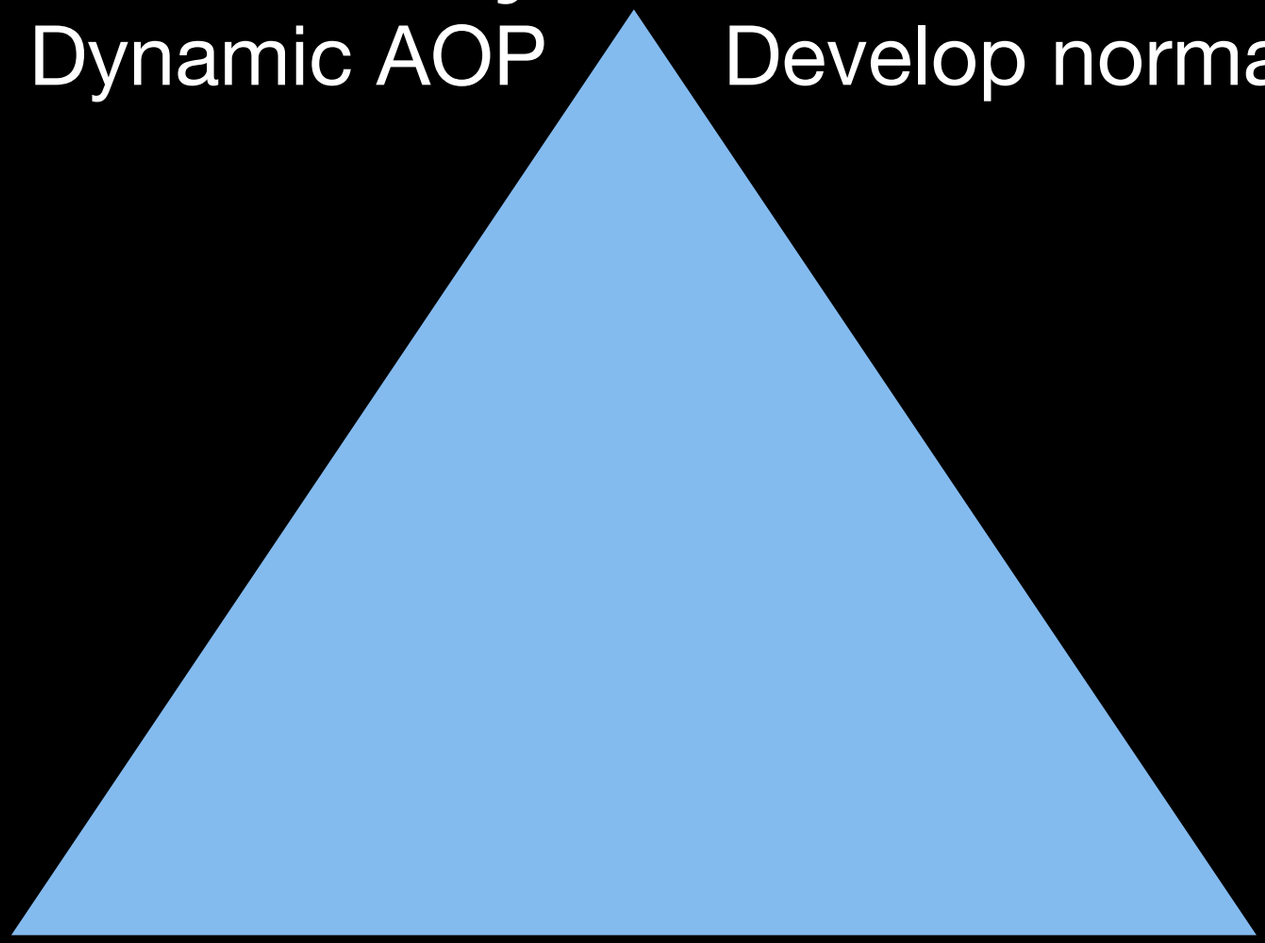
Powerful

In the spirit of Smalltalk

Dynamic

Dynamic AOP

Develop normally



Simple

Minimal set of Constructs

Uniform / no restrictions

Powerful

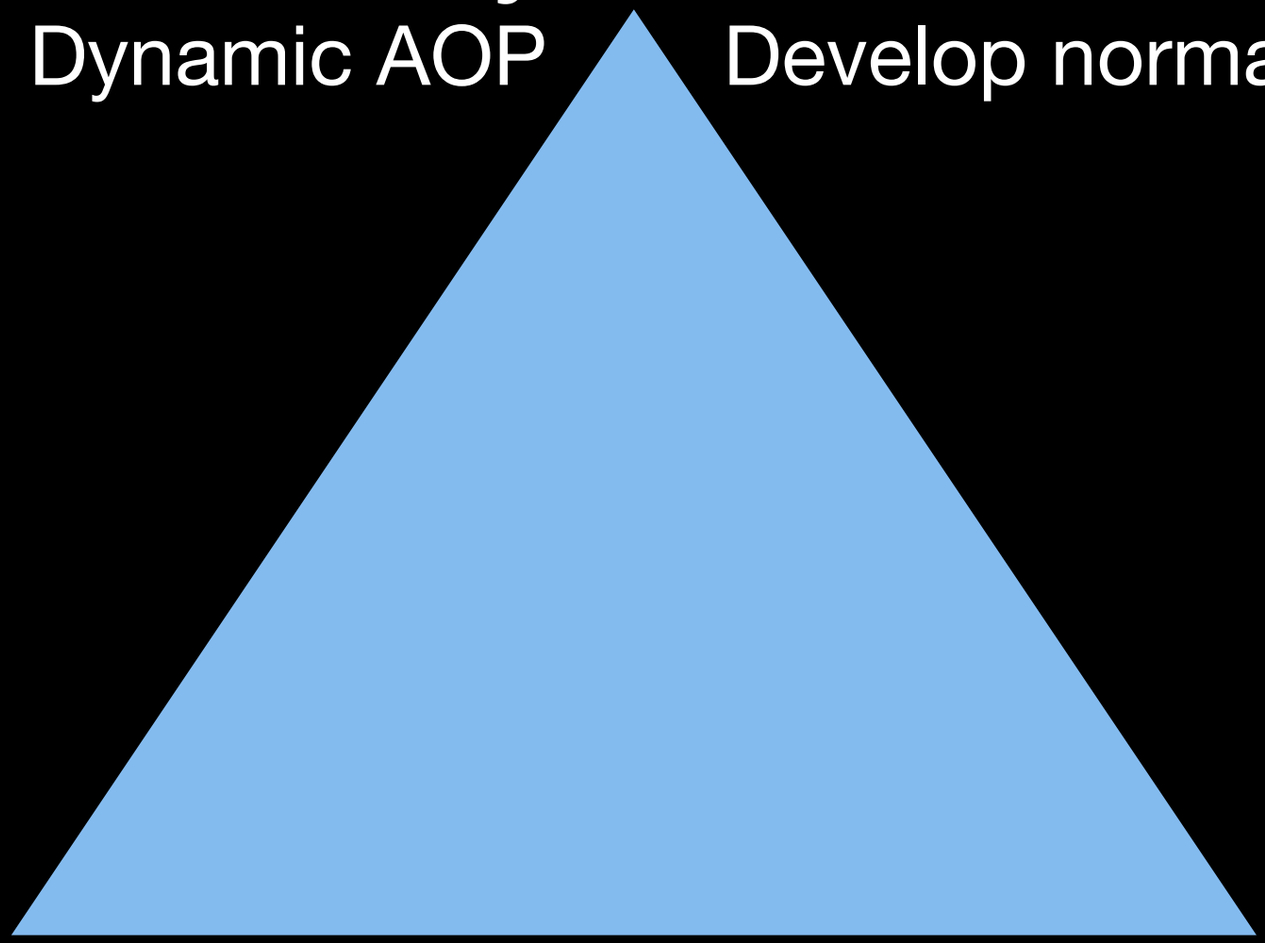
First-class everything

In the spirit of Smalltalk

Dynamic

Dynamic AOP

Develop normally



Simple

Minimal set of Constructs

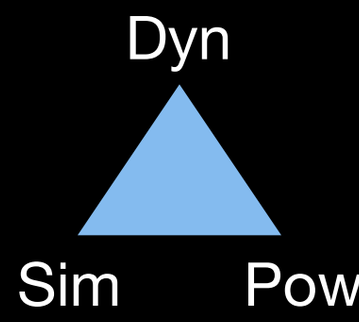
Uniform / no restrictions

Powerful

First-class everything

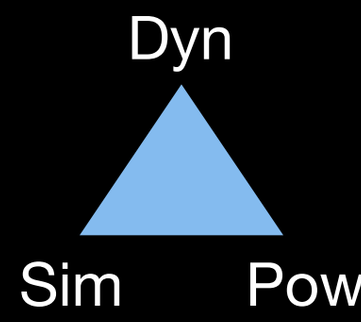
New language features

Pointcuts



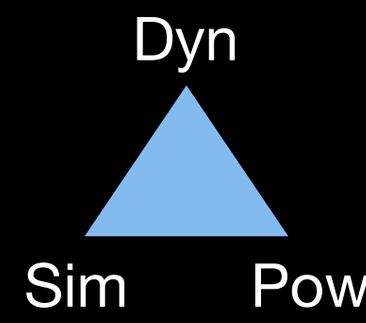
Pointcuts

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



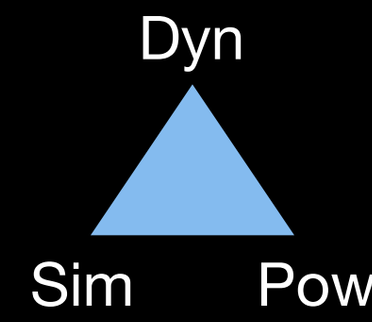
Pointcuts

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

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



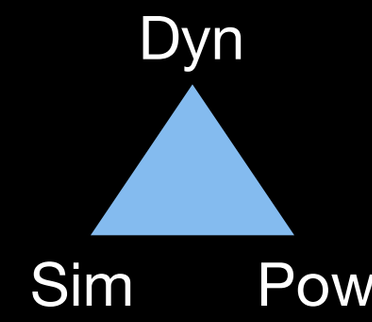
Pointcuts

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

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

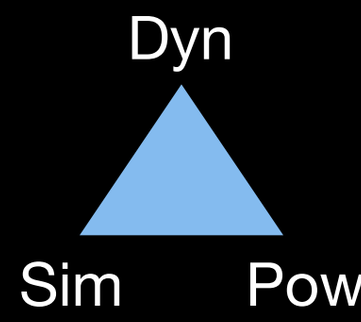


Pointcuts

```
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)
```

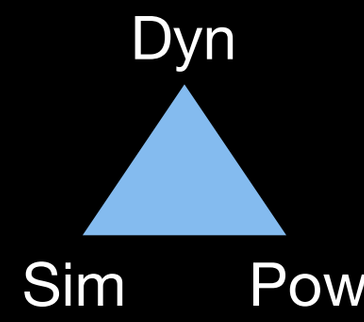


Advice



Advice

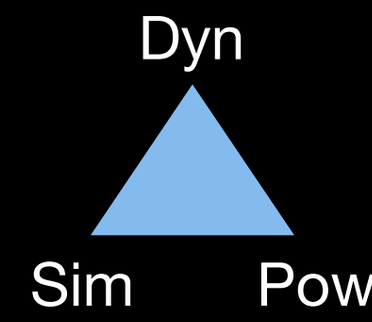
```
PhAdvice pointcut: pc
     send: #incCount: to: self
     type: #after
```



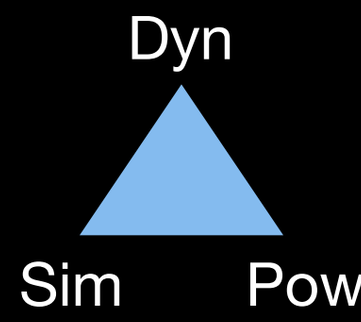
Advice

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

```
PhAdvice pointcut: pc
     advice: [:ctx |
          Transcript show:
               (ctx receiver asString);cr.
          ctx proceed.]]
     type: #around.
```

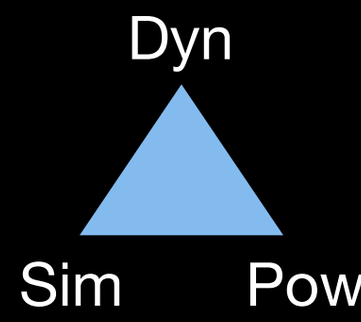


Inter-Type Declaration



Inter-Type Declaration

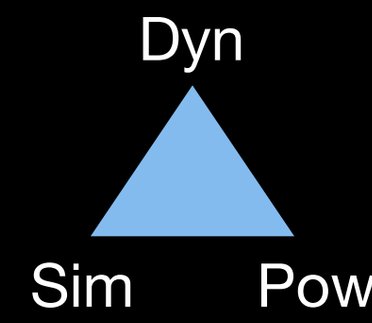
```
PhClassModifier on: ACTestCase
    addIV: 'phacount'
```



Inter-Type Declaration

```
PhClassModifier on: ACTestCase
    addIV: 'phacount'
```

```
PhClassModifier on: ACTestCase
    addIM: 'phacount
        ↑phacount ifNil: [phacount := 0]'
```



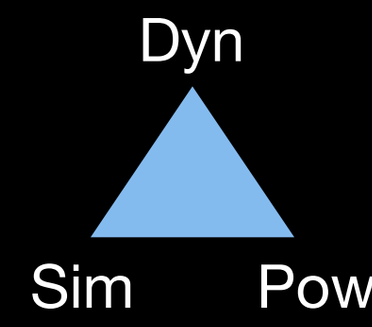
Aspect

```
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?

```
|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?

```
|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?

```
|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?

```
|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?

```
|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?

```
|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?

```
|asp|
asp := PhAspect new ad
  (PhAdvice
     pointcut: (P  oint  t
        receivers 'Tra  cript class'
        se   ct  :    ow  )
     advice    [ Tr  cript show: 'Have ']
     ty   : #  fore)
asp i  t  ll.
Tran      show: 'reentrancy control'; cr
```

BOOM

Reentrancy control

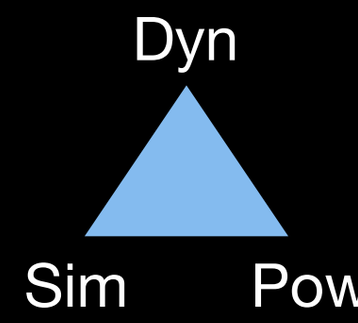
PHANtom: no problem!

- 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

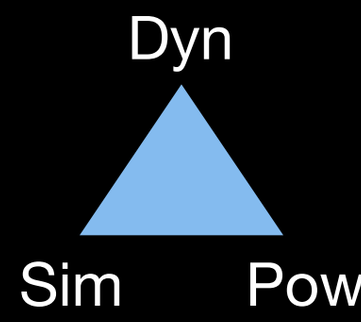
- Global and static

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 ...

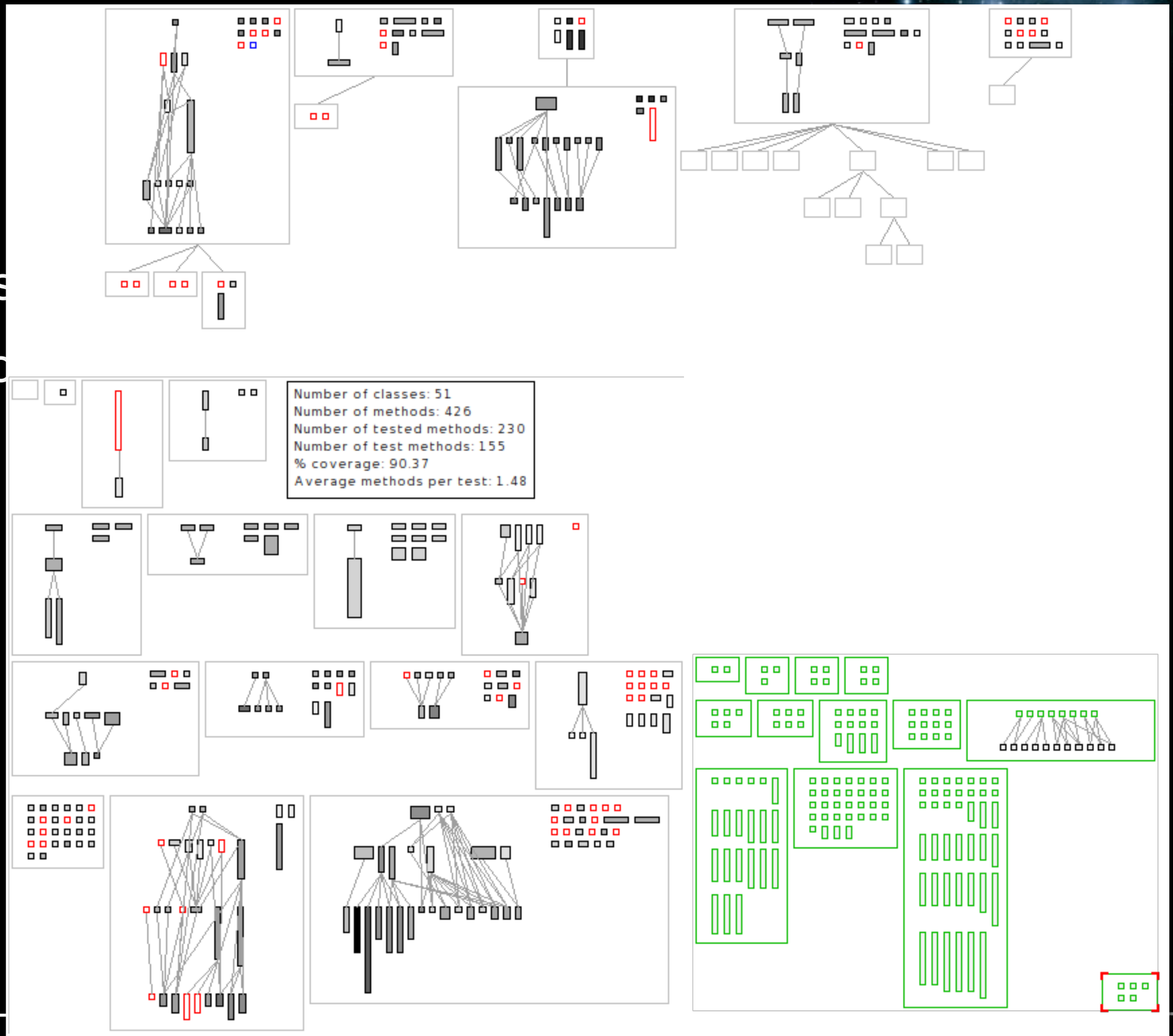
TDD

- 155 tests
- 90+% coverage

Not in the paper ...

TDD

- 155 tests
- 90+% co



Future work

Optimize the implementation

Compile PHANtom code

Infrastructure for Domain-Specific Aspect Languages

http://pleiad.cl/PHANtom