Towards Structural Decomposition of Reflection with Mirrors

Nick Papoulias
Noury Bouraqadi
Marcus Denker
Stephane Ducasse
Luc Fabresse
why

- Reflection as a pluggable sub-system for:
  - Remote Programming
  - Resources
  - Security
  - Cleaner OO Design
Problems with Reflection

- Breaks encapsulation
- Mixed base and meta access
- Resource intensive
- Security / Integrity threat
- No OO design (supporting multiple implementations)

Smalltalk Example
Meta level functionality is indistinguishably mixed in program code.

`mvCar := Car new.`
Smalltalk Example

Meta level functionality is indistinguishably mixed in program code.

```
myCar := Car new.
nNumberOfDoors := myCar numberOfDoors.
carClass := myCar class.
anotherCar := carClass new.
carSuperClass := carClass superclass.
```
numberOfDoors := myCar numberOfDoors.
carClass := myCar class.
anotherCar := carClass new.
carSuperClass := carClass superclass.

Mirrors  (Bracha & Unghar 2004)
Functional decomposition of Reflection
  • Encapsulation
  • Stratification
  • Ontological Correspondance

Smalltalk Example - with Mirrors
Meta level functionality is indistinguishably mixed in program code.

myCar := Car new.
numberOfDoors := myCar numberOfDoors.
carMirror := Mirror on: myCar.
carClassMirror := carMirror class.
Smalltalk Example - with Mirrors

Meta level functionality is indistinguishably mixed in program code.

myCar := Car new.
numberOfDoors := myCar numberOfDoors.
carMirror := Mirror on: myCar.
carClassMirror := carMirror class.
carClassSuperMirror := carClassMirror superclass.
But what about meta-information?
I represent an x-y pair of numbers usually designating a location on the screen.
impact on:

- Encapsulation/Stratification
- Resources
- Security
What is Base and what is Meta?
There is a powerful inherent ambiguity!

"With great power comes great responsibility.."
--Uncle Ben--

or Voltaire!
Our goal

<table>
<thead>
<tr>
<th>Reflective Functionality</th>
<th>Smalltalk</th>
<th>Classic mirror-based systems</th>
<th>Our goal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>FIX</strong></td>
<td><strong>DISCARDABLE</strong></td>
<td><strong>DISCARDABLE</strong></td>
</tr>
<tr>
<td>Meta-Data</td>
<td><strong>FIX</strong></td>
<td><strong>FIX</strong></td>
<td><strong>DISCARDABLE</strong></td>
</tr>
<tr>
<td>Base-Level</td>
<td><strong>FIX</strong></td>
<td><strong>FIX</strong></td>
<td><strong>FIX</strong></td>
</tr>
</tbody>
</table>

all the power, without the responsibility
Our proposition

Mirrors as the storage entities of meta-information

Inclusion explicitly through mirrors can only be done through mirrors
every object has a mirror

mirrors hold all meta-information

dynamic addition of behaviour can only be done through mirrors

all other entities provide reflection explicitly through mirrors

the power, without

the responsibility
Validation & Prototype

MetaTalk

Whole system: VM, Compiler, Object Model implemented on top of Pharo.

Sound execution of base level functionality both:
- In the presence and
- In the absence of mirrors
- Validated complete stratification of meta-information when they are not needed.

Related Work

- Bracha and Unghar 2004
- Lorenz and Vlissides 2003
- Declarative model for Smalltalk
Related Work

- Bracha and Unghar 2004
- Lorenz and Vlissides 2003
- Declarative model for Smalltalk
- Resilient
- Mirages and AmbientTalk
Future Work

- Metrics
- Behavioral Reflection
- Remote Programming (case studies)
Metrics
Behavioral Reflection
Remote Programming (case studies)

http://squeaksourcemetatalk
Thank you!