Pharo: A Reflective System

Marcus Denker

http://rmod.lille.inria.fr
What is it?

- Language + Environment
- Simple Language (based on Smalltalk)
- Object-Oriented, Dynamic, Reflective
  - Explore + Change running systems

- The Ultimate Programming Environment!
Pharo: Open Source

- MIT license
- Mac, Linux, Windows (Android, iOS)
- Great community
- Improving steadily
- Many excellent libraries
Simple Language
“Objective-C without C”
Reflection
Code (description)
Description

Running Program
Description

Running Program
Definition:

A **reflective system** is a system which incorporates causally connected structures representing (aspects of) itself.

(Patty Maes, OOPSLA 87)
Introspection

- Introspection
  - Self-representation can be queried
- Intercession
  - Self-representation can be changed

- Reflection = Introspection + Intercession
Structure and Behavior

- **Structural** Reflection
  - Concerned with static structure
  - packages, classes, methods...

- **Behavioral** Reflection
  - Concerned with program execution
  - method execution, assignment, execution stack...
Why?
Future Systems...

- **Complex**: Runtime can not be ignored
  - Analyse at runtime
  - Live Programming

- **“Eternal”** Evolution
  - Development does not stop after deployment
  - Enable evolution in deployment
Live Programming: Development at Runtime
Reflection is the basis
Classes, Methods, Packages are Objects
The tools manipulate these Objects
Inspector
Demo: Inspect World
Demo: Changing a class at runtime
Demo: Reflective Execution Stack
Reflective Challenges
Three Examples

- Partial Behavioral Reflection
- Controlling Reflection
- Reflection and the Virtual Machine
Advanced Reflection

- Partial Behavioral Reflection
- Associate MetaObject with structural object
  - e.g. methods, slots…
- AST nodes
Partial Behavioral Reflection

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

But without changing the program code!
Partial Behavioral Reflection

- Meta-object
- Activation condition
- Source code (AST)
- Links
Uses...

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

- Reflection is dangerous
- Everything is possible

- Can we control the **effect** of Reflective Change?
System - as - Objects

- Put “virtualization” in the language
- We already use “Images”
- Make the Image a first class concept in the language
Pharo JIT Compiler

- “traditional JIT”: per method translation to native code
- Already fast, but slower than e.g JVM

- Next Step: Type feedback runtime optimiser in the VM
  - A lot of work ==> not realistic
Reflective Optimization

- We can stop execution
- We can manipulate the stack
- And then continue
Type Feedback JIT

Sista

Work with E. Miranda
Future: Beyond Text

- Methods in Pharo are still Strings

- Why not use a reflective model for methods?

- Text editor modifies directly the objects, not text
Open Pharo Sprints

May 2008 Bern
July 2009 Bern
October 2009 Lille
November 2009 Buenos Ares
March 2010 Bern
May 2010 Buenos Ares
June 2010 Bern
June 2010 Bruxelles
July 2010 London
September 2010 **Barcelona**
September 2010 Lille
January 2011 Lille
July 2011 Lille
October 2011 Bruxelles
February 2012 Bern
April 2012 Lille
September 2012 Ghent
October 2013 Lille
November 2013 Buenos Aires
The next one is **tomorrow**!

14h Building B in front of B31
Picture on Slide 7:

Source code ON PAPER, available under Creative Commons

http://www.flickr.com/photos/toolmantim/6170448143/