Metaprogramming and Reflection
Behavioral Reflection

Universität Bern
Marcus Denker
Hasso-Plattner-Institut Potsdam
Software Architecture Group
Prof. Dr. Robert Hirschfeld
http://www.swa.hpi.uni-potsdam.de
WS 2006/2007

Topics Covered

- Overview
- Introduction
- Open implementations
- OMG meta object facility
- CLOS metaobject protocol
- Smalltalk/Squeak
- Behavioral reflection
- Refactoring browser
- AspectS and ContextS
- Traits and mirrors
- Metacircular interpreters

Outline

- Introduction: Reflection in Squeak
- Sub-method Structure: Bytecode
- ByteSurgeon: Bytecode Transformation
- Partial Behavioral Reflection

Reflection

- Reflection: computation about computation
  - Base level / meta level
  - Causally connected
- Structural Reflection
  - Reification of structure
- Behavioral Reflection
  - Reification of execution

Reflection in Squeak

- Squeak has support for reflection
- Structural Reflection
  - Classes / Methods are Objects
  - Can be changed at runtime
- Behavioral Reflection
  - Current execution reified (thisContext)
  - #doesNotUnderstand / MethodWrappers

Can we do better?

- Structural Reflection stops at method level
  - Bytecode in the CompiledMethod: Numbers
  - Text: Just a String, needs to be compiled
- Behavior hard coded in the Virtual Machine
  - Message Sending
  - Variable Access
- Behavioral Reflection is limited in Squeak
  - We should do better!

Sub-Method Abstraction

- We need a model for method bodies
- Posibilities:
  - Text
  - Bytecode (CompiledMethod)
  - AST (Abstract Syntax Tree)

- For now: Bytecode
  - Bytecode is causally connected
  - Quite fast to edit

- Later: AST (next Lecture)

Outline

- Introduction: Reflection in Squeak
- Sub-method Structure: Bytecode
- ByteSurgeon: Bytecode Transformation
- Partial Behavioral Reflection

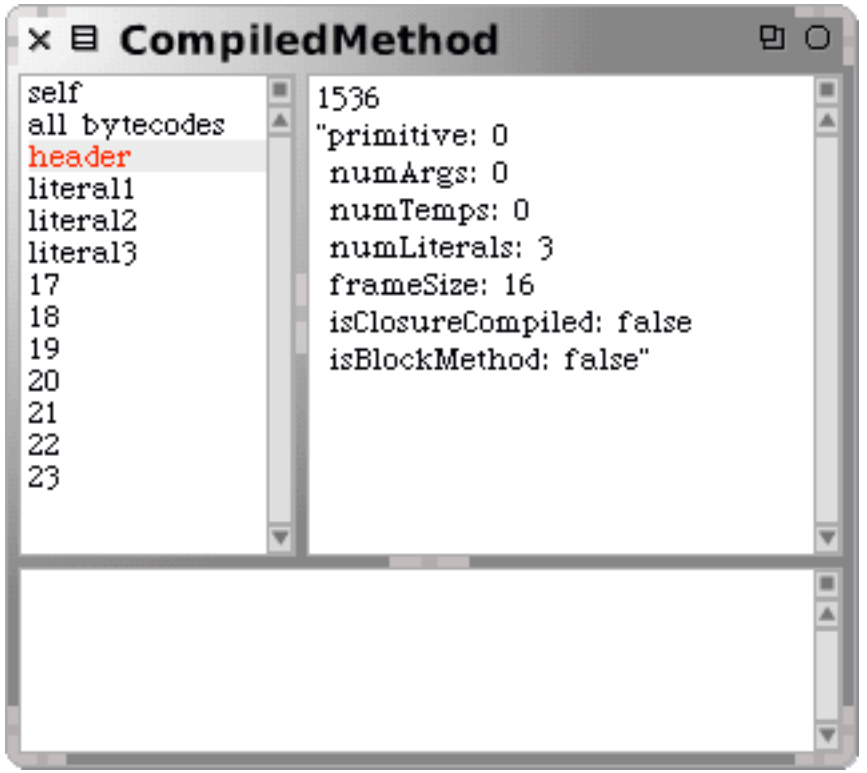
Squeak VM

- Virtual machine provides a virtual processor
- Smalltalk (like Java): Stack machine
  - easy to implement interpreters for different processors
  - most hardware processors are register machines
- Bytecode: ‘Machine code’ of the VM

CompiledMethods

- CompiledMethod format:

Header	Number of temps, literals...
Literals	Array of all Literal Objects
Bytecode	
Trailer	Pointer to Source




```
(Number>>#asInteger) inspect
```

```
(Number methodDict at: #asInteger) inspect
```

Squeak Bytecode

- 256 Bytecodes, four groups:
  - Stack Bytecodes
    - Stack manipulation: push / pop / dup
  - Send Bytecodes
    - Invoke Methods
  - Return Bytecodes
    - Return to caller
  - Jump Bytecodes
    - Control flow inside a method

Editing Bytecode

- Why Bytecode?
  - No source needed
  - Performance
  - Language independent (e.g. Ruby-on-Squeak)

- Problems of direct Bytecode editing
  - Hard to edit directly (e.g jump offsets)
  - Programmers don‘t know bytecode
  - Easy to make errors

Outline

- Introduction: Reflection in Squeak
- Sub-method Structure: Bytecode
- ByteSurgeon: Bytecode Transformation
- Partial Behavioral Reflection

ByteSurgeon

- Library for bytecode transformation in Smalltalk
- Full flexibility of Smalltalk: Runtime
- Provides high-level API
- For Squeak, but portable

[Main Source] Marcus Denker, Stéphane Ducasse and Éric Tanter. *Runtime Bytecode Transformation for Smalltalk.* Journal of Computer Languages, Systems and Structures, vol. 32, no. 2-3.

Examples

- Counts the number of Bytecodes:

```
InstrCounter reset.
Example instrument: [:instr  | InstrCounter increase]
```

- Counts the number of Sends:

```
InstrCounter reset.
Example instrumentSend: [:instr|InstrCounter increase]
```

- Introspection:

```
(Example>>#aMethod) instrumentSend: [:send |
      Transcript show: send selector printString]
```

Transformations

- Modification: inlining of code
  - insertBefore:, insertAfter:, replace:

```
(Example>>#aMethod) instrumentSend: [ :send |
      send insertAfter: '[InstrCounter increase]'
```

```
(Example>>#aMethod) instrumentSend: [:send |
      send insertAfter: 'Transcript show:',
                        send selector printString].
```

Meta Variables

- Goal: extend a send with after logging
- Problem: How to access receiver and args?
- Solution: metavariables

```
Example instrumentSend: [:s |
    s insertAfter: 'Logger logSendTo: <meta: #receiver>']
```

- #receiver, #arguments, #argn, #result.... #value, #newvalue

ByteSurgeon

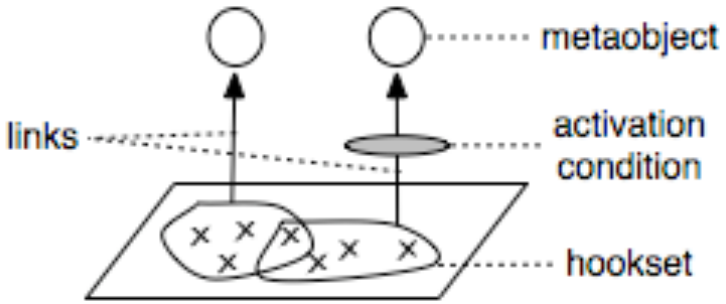
- Provides simple model for submethod structure
  - Too low level?
- Used in a number of projects
  - Trace Debugger
  - Test Coverage
- Next Step: Behavioral Reflection
  - Bytesurgeon just concerned with inlining of bytecode
  - We want to have reifications + meta objects

Outline

- Introduction: Reflection in Squeak
- Sub-method Structure: Bytecode
- ByteSurgeon: Bytecode Transformation
- Partial Behavioral Reflection

Partial Reflection: Reflex

- Hooksets: collection of operation occurrences
- Links
  - Bind hooksets to metaobjects
  - Define Protocol between base and meta
- Goals
  - Highly selective reification
  - Flexible metalevel engineering
    - *Protocol specification*
    - *Cross-cutting hooksets*



Geppetto

- Partial Behavioral Reflection pioneered in Java
  - Code transformation at load time
  - Not unanticipated (it's Java...)
- Geppetto: Partial Behavioral Reflection for Smalltalk
  - For Squeak 3.9 with ByteSurgeon
  - but portable to other dialects
- Let's see an example!

[Main Source] David Röthlisberger, Marcus Denker and Éric Tanter. Unanticipated Partial Behavioral Reflection: Adapting Applications at runtime. to appear

Example

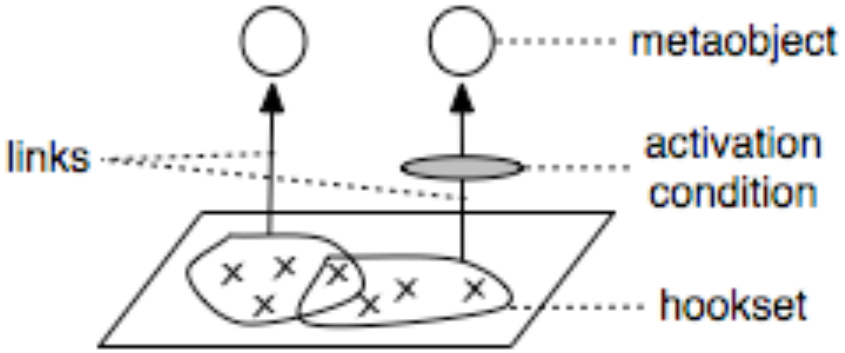
- Typical Web Applications (e.g. Wiki)
- Shows performance problem under high load
- Goals:
  - Profile and fix the problem
  - No restart / interruption of service

Towards a Solution

- Analyze the problem
  - Install Profiler
  - Analyze
  - Retract Profiler
- Solve the Problem
  - Introduce a caching mechanism
  - Experiment with different solutions

Solution with Reflection

- Operation
  - Method Evaluation
- Hookset
  - All method executions in the wiki
- Metaobject
  - A profiler tool



Profiler: Hookset + Link

Hookset

```
allExecs := Hookset new.
allExecs inPackage: 'Wiki'; operation: MethodEval.
```

Link

```
profile := Link id: #profiler
   hookset: allExecs
   metaobject: Profiler new. profile
   control: Control around.
```

Profiler: Protocol

Protocol

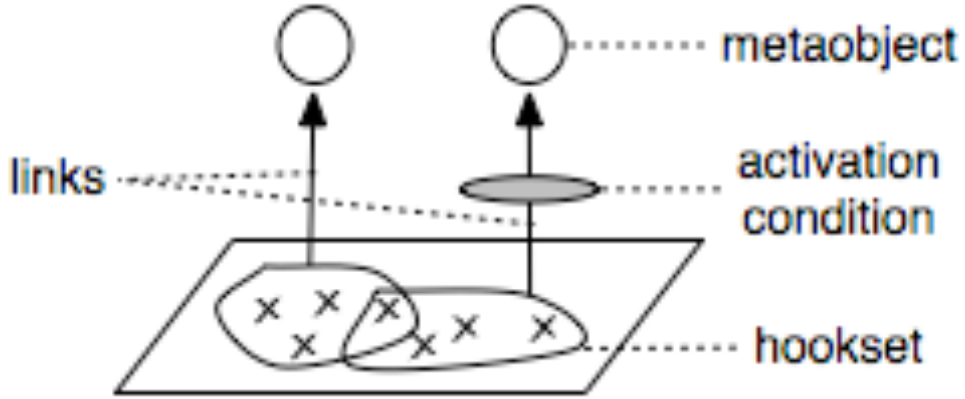
```
profile callDescriptor:
   (CallDescriptor
      selector: #profileMethod:in:withArguments:
      parameters:{ Parameter selector.
                   Parameter self.
                   Parameter arguments.}
      passingMode: PassingMode plain).
```

Install / Retract

```
profile install.
profile uninstall.
```

Solving the Problem: Caching

- Operation:
  - Method Execution (Around)
- Hookset
  - The one slow method (`#toughWork:`)
- Metaobject
  - The cache



Cache: Hookset and Link

Hookset:

```
toughWorks := Hookset new.
toughWorks inClass: Worker;
           inMethod: #toughWork:;
           operation: MethodEval.
```

Link:

```
cache := Link id: #cache
   hookset: toughWorks
   metaobject: Cache new.
cache control: Control around.
cache callDescriptor:(CallDescriptor
      selector: #cacheFor:
      parameters: {Parameter arg1}
      passingMode: PassingMode plain).
```

Geppetto

- Operations
  - MethodEval
  - MessageSend, InstVarAccess, TempAccess
- Control
  - Before, After, Around, Replace
- Activation condition per Link

Future Work

- Pluggable Backends
  - Bytecode
  - AST
  - VM Support?
- AST: integrate with annotation framework
- Better tool support

Conclusion

- Introduction: Reflection in Squeak
- Sub-method Structure: Bytecode
- ByteSurgeon: Bytecode Transformation
- Partial Behavioral Reflection

- Next Lecture: Refactoring Engine

Conclusion

- Introduction: Reflection in Squeak
- Sub-method Structure: Bytecode
- ByteSurgeon: Bytecode Transformation
- Partial Behavioral Reflection

- Next Lecture: Refactoring Engine

Questions?