

The Reflectivity

Marcus Denker

with:

David Röthlisberger

Philippe Marschall

Nik Haldiman

Adrian Lienhard / Lukas Renggli

Eric Tanter

Stephane Ducasse

Oscar Nierstrasz

Structural Reflection

- > Structure modeled as objects
 - e.g. Classes, methods
 - Causally connected

- > Uses:
 - Development environments
 - Language extensions and experiments

Methods and Reflection

- > Method are Objects
 - e.g in Smalltalk

- > No high-level model for sub-method elements
 - Message sends
 - Assignments
 - Variable access

- > Structural reflection stops at the granularity of methods

Sub-Method Reflection

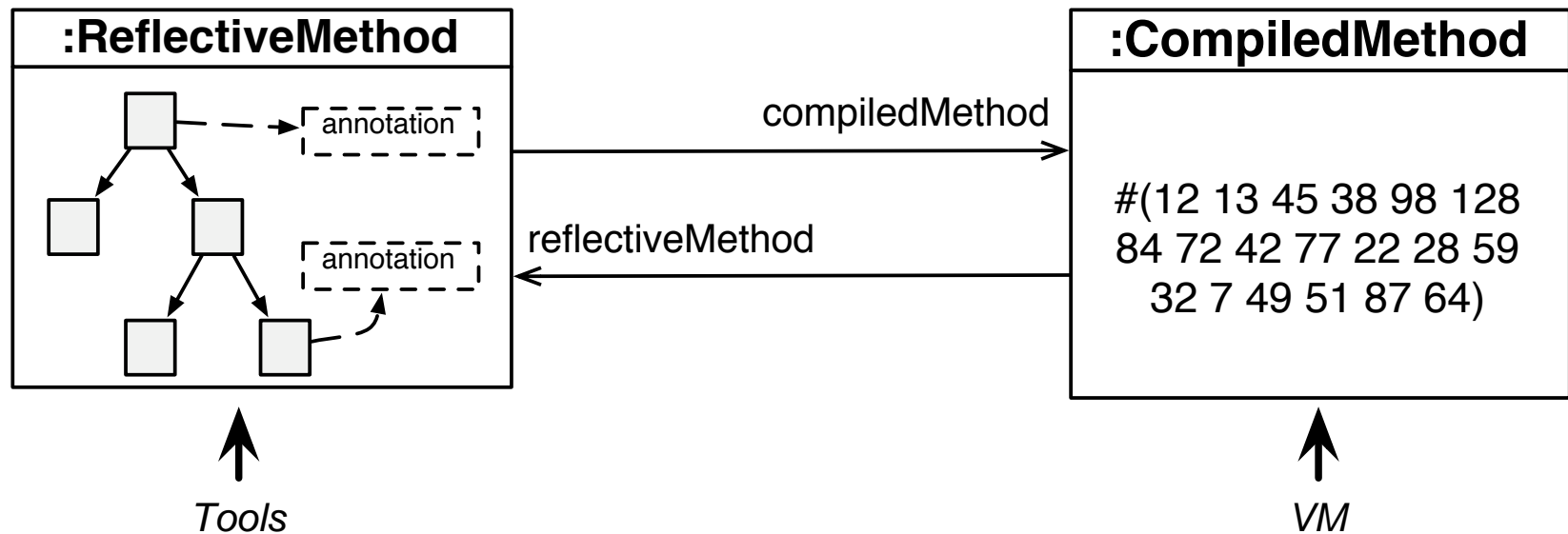
- > Many tools work on sub method level
 - Profiler, Refactoring Tool, Debugger, Type Checker

- > Communication between tools needed
 - example: Code coverage

- > All tools use different representations
 - Tools are harder to build
 - Communication not possible

Solution: Reflective Methods

- > Annotated, persistent AST
- > Bytecode generated on demand and cached



Reflectivity

- > Implementation of Reflective Methods for Squeak Smalltalk

- > Smalltalk Compiler generates Reflective Methods
 - Translated to Bytecode on demand

- > Open Compiler: Plugins
 - Generator plugin: called before code generation
 - *Transform a copy of the AST*
 - Analysis plugin: called after name analysis

DEMO I

- > Show invalidation of code

- > Show assert Demo

Reflective Methods: Annotations

- > Source visible annotations
 - extended Smalltalk syntax

(9 raisedTo: 10000) <:evaluateAtCompiletime:>

- > Source invisible annotations
 - Reflective API
 - Can reference any object
- > Every node can be annotated
- > Semantics: Compiler Plugins

Example: Pluggable Type-System

- > Example for textual annotations

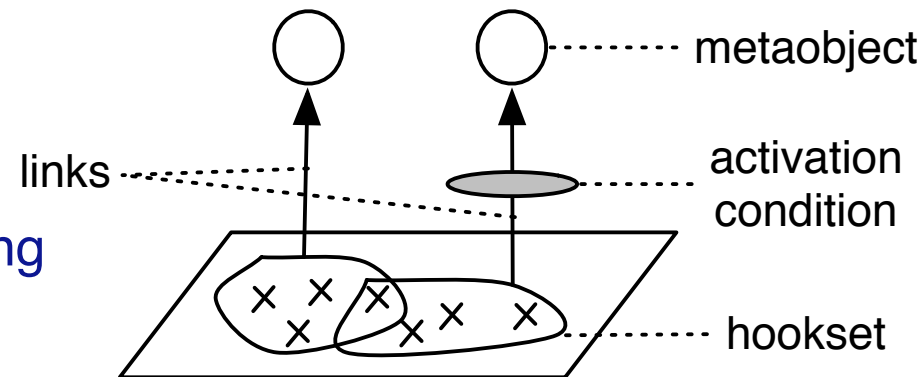
```
bitFromBoolean: aBoolean <:type: Boolean :=  
^ (aBoolean ifTrue: [1] ifFalse: [0]) <:type: Integer :=
```

- > Optional, pluggable type-system
- > Types stored as annotations in the Reflective Methods

Reflex: Partial Behavioral Reflection

- > Hooksets: collection of operation occurrences
- > Links
 - Bind hooksets to metaobjects
 - Define Protocol between base and meta

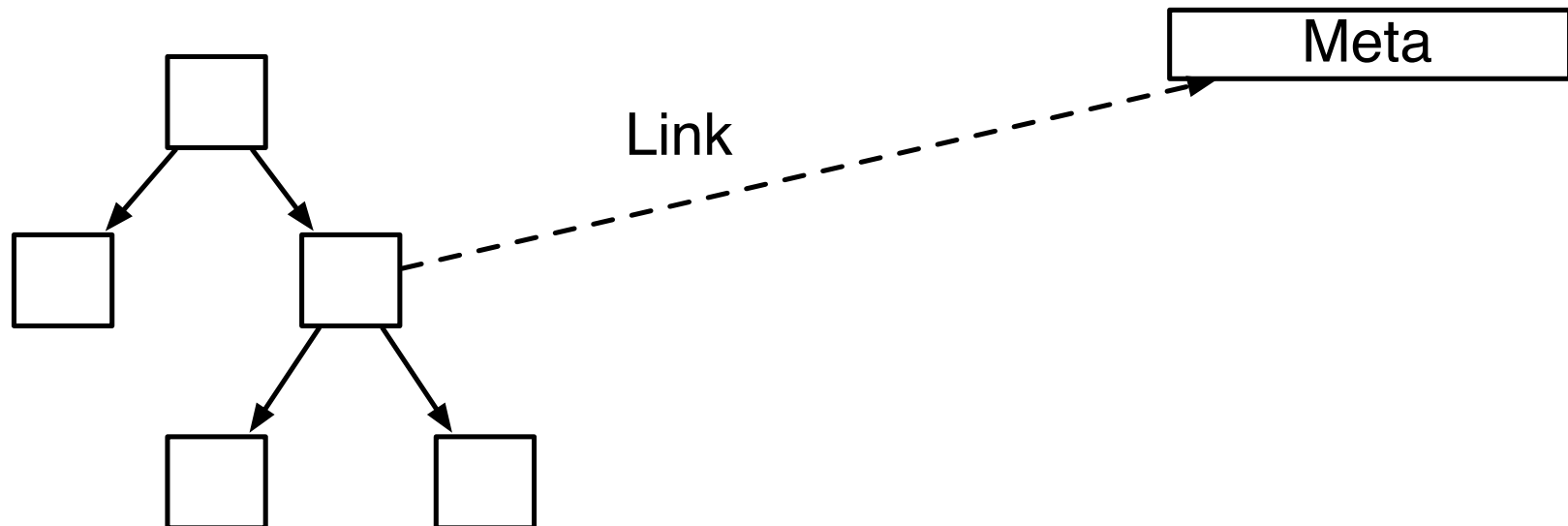
- > Goals
 - Highly selective reification
 - Flexible metalevel engineering
 - *Protocol specification*
 - *Cross-cutting hooksets*



Tanter, OOPSLA03

Links as Annotations

- > Links can be annotations on the AST



Demo II: Geppetto

- > Show Bounce Demo

- > Show Coverage Demo

Future Work

- > Optimize Size of AST Representation
 - Simpler AST
 - AST Compression

- > Contextual Reifications
 - Context depended Links

- > Beyond Text
 - Store only AST (no text)
 - Build text from annotated AST