

1

AOP using Reflection in Smalltalk The MetaclassTalk Experiment

Noury Bouraqadi
Computer Science Laboratory (CSL)
Ecole des Mines de Douai
France

2

Outline

- What is AOP?
- What is Reflection?
- What is MetaclassTalk?
- AOP Using MetaclassTalk
- Conclusion

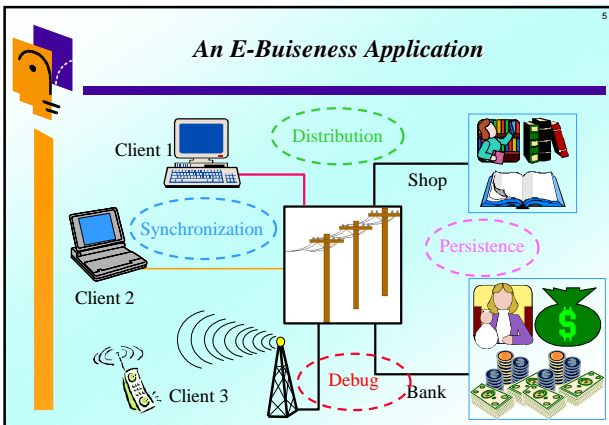
3

What is AOP?

4

From OOP to AOP

- AOP is a new paradigm for building programs
 - Led to AOSD
- AOP does not discard OOP
 - but, AOP is not bound to OOP
- AOP addresses 2 OOP limitations
 - Code tangling
 - Cross-cutting



6

Code Tangling

```

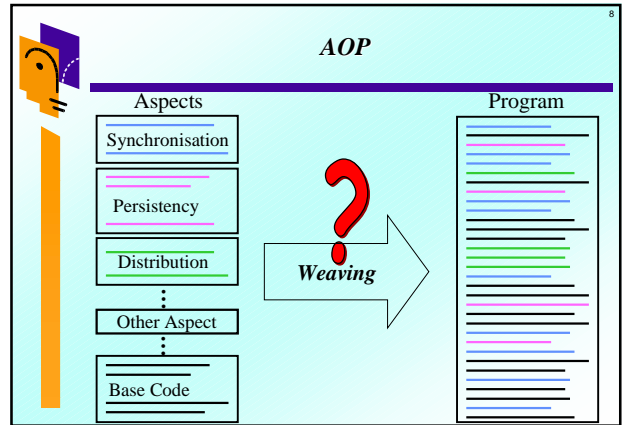
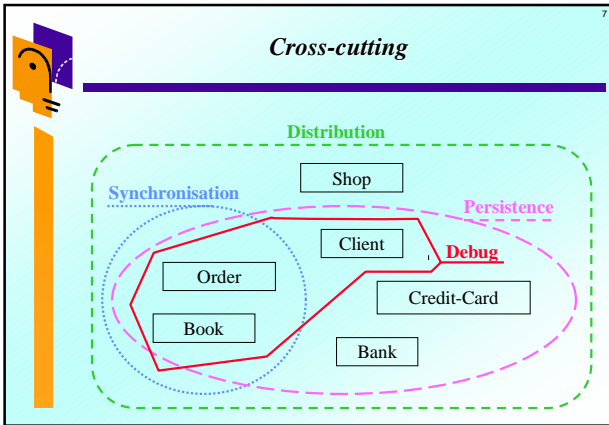
RemoteObject subclass: #Book
instanceVariableNames: 'title author price lock' ...

initialize
lock := Semaphore forMutualExclusion.
self price: ("myDataBase priceFor: self")...
super initialize
"self start port listner = remote communication + marshaling"
"register into a name registry"

price
|currentPrice|
self halt.
lock critical: [currentPrice := price].
^currentPrice

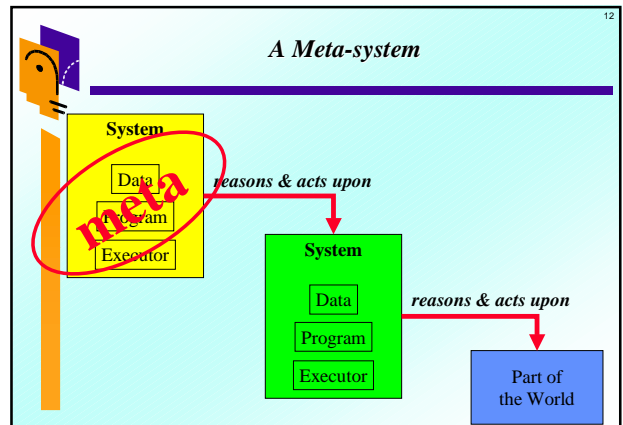
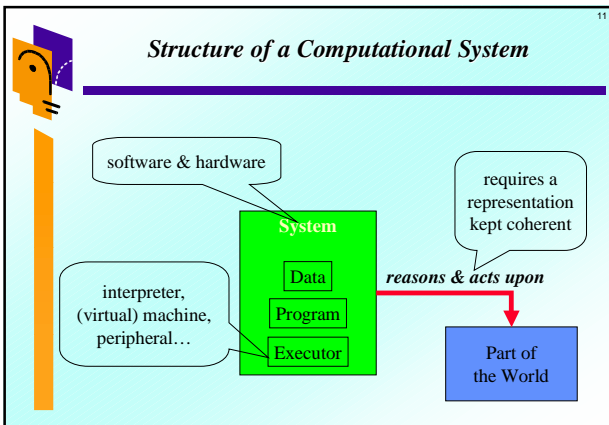
price: newPrice
lock critical: [price := newPrice.
Transcript cr; show: self printString.].
"myDataBase setPrice: newPrice for: self"
Transcript cr; show: 'DataBase price: ', ("myDataBae priceFor: self") printString.
  
```

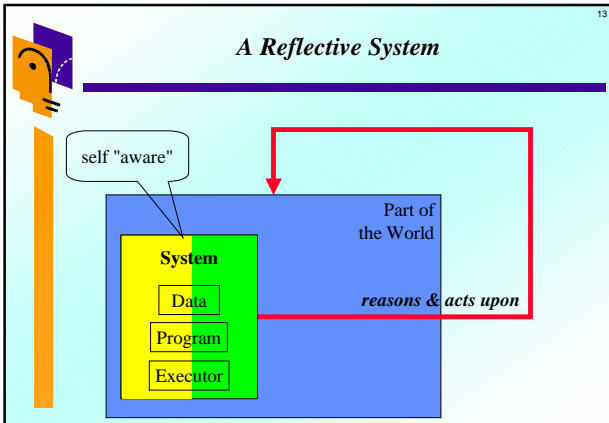
Debug
Persistence
Distribution
Synchronisation



- ### AOP Concepts
- **Base Code**
 - Most functionalities
 - **Aspect**
 - Unit of code that describes a single "global" property
 - Examples: Remote communication, Synchronization, ..
 - **Join points**
 - Points of the base code execution flow
 - Example: sending message *m* to object *o*
 - **Weaving**
 - Assembling aspects with base code at join points

What is Reflection?

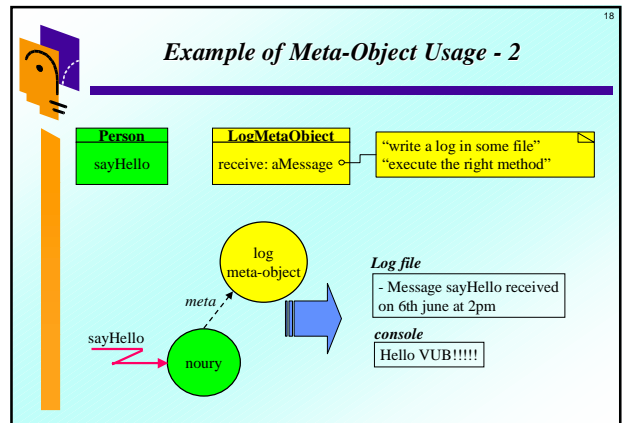
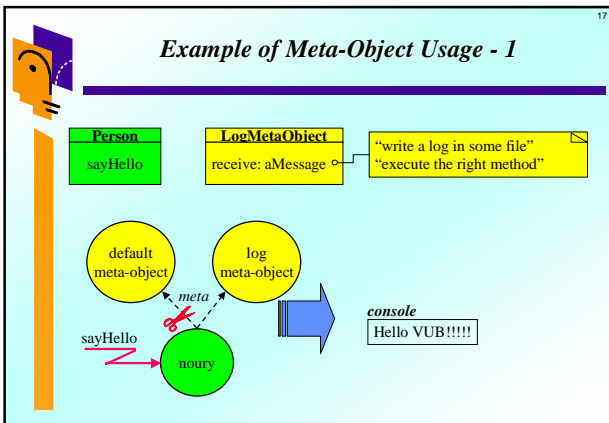




- ### Definitions
- **Reflection : Ability of a System to**
 - observe itself
 - Reason on itself
 - change itself
 - Alter its structure (program)
 - Alter its behavior (executor)
 - **Reify : Represent some concept as an explicit entity**
 - Building blocks of program and executor
 - **A Reflective Language**
 - Language which constructs and "interpreter" are reified

- ### A System in OOPs
- **Executor = (virtual) machine, interpreter, ...**
 - language semantics
 - program loading & compilation
 - memory management
 - ...
 - **Program**
 - classes, methods, fields, messages ...
 - **Data**
 - objects : bank, clients, accounts, ...
- } more or less reified
↓
Degree of Reflection

- ### OO Reflective Languages
- **Reification = representing entities as objects**
 - e.g. classes = instance of other classes: **Meta-Classes**
 - **Meta-object**
 - is an object
 - **controls** one or more base-objects
 - i.e. responsible of the execution of messages, field accesses, ...
 - MOP = Meta-Objects Protocols
 - **Two programming levels**
 - executor → meta-level → **meta-objects**
 - program → base-level → **base-objects**



19

The Meta Link

- **Meta-link = link objects to their meta-objects**
 - Granularity = object
 - Sibling objects linked to different meta-objects
- **An object can be linked to many meta-objects**
 - Meta-objects should cooperate
- **A meta-object can be shared**
 - i.e. linked to many base-objects

20

Reflective Towers & Infinite Regression

- **Meta-meta-objects**
 - Meta-objects
 - control other meta-objects
- **Meta-meta-objects are objects**
 - controlled by meta-meta-meta-objects
 - Infinite tour
- **Stopping the infinite regression**
 - Primitive/Default meta-object

21

Reflection is Useful

- **Development tools**
 - Browser, Debugger, Code Generators, ...
- **Run-time Flexibility**
 - Quality of Service, Unplanned Evolution, ...
- **Adapt and Extend the Language**
 - Mixin based inheritance, Asynchronous Communication, Lazy memory allocation...
- **Ease Software Development**
 - Generic Code ⇌ Reuse
 - Separation of Concerns

22

What is MetaclassTalk?

23

A Reflective extension of Smalltalk

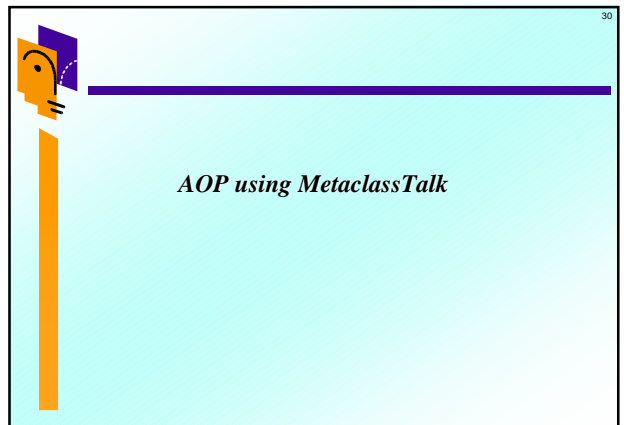
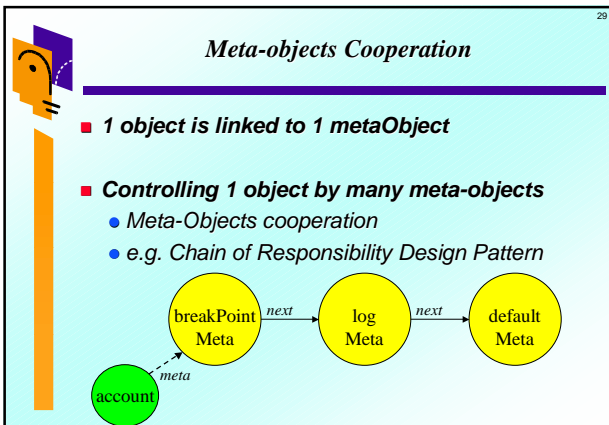
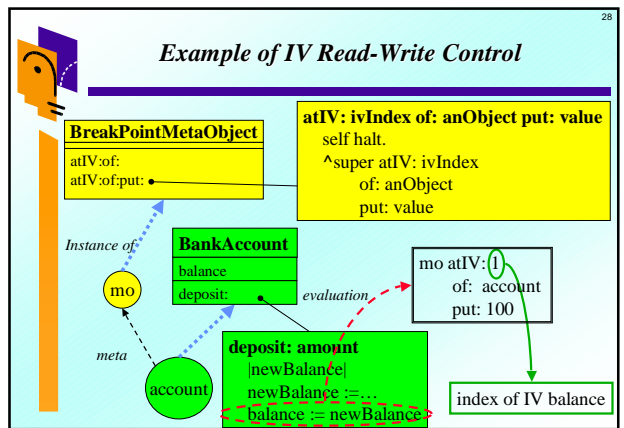
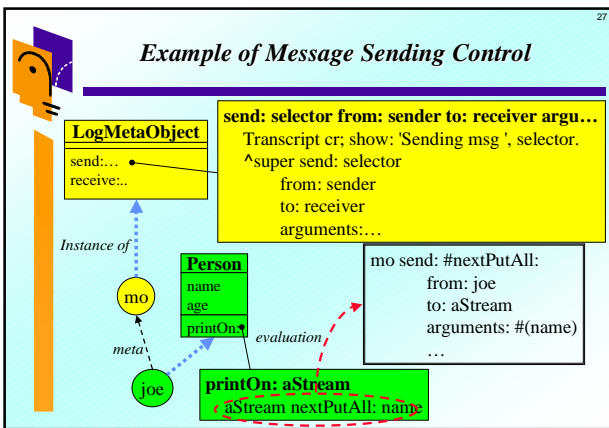
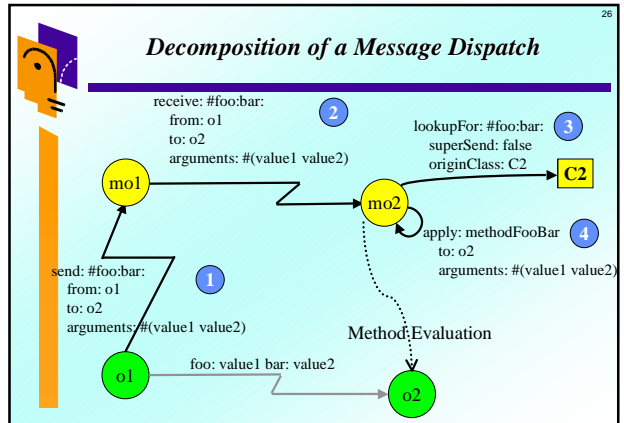
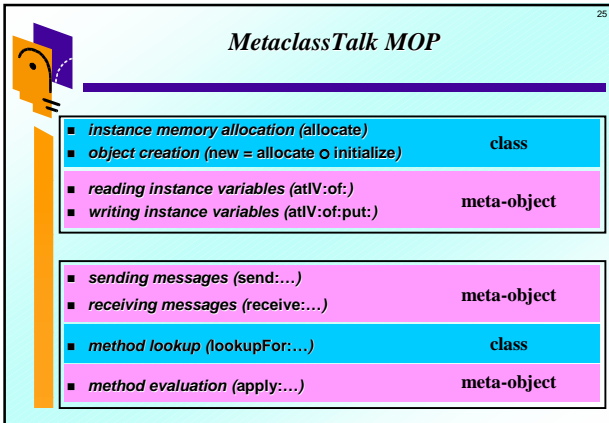
- **Explicit metaclasses**
 - New kinds of classes
 - Class properties
- **Meta-object (MOP)**
 - Objects structure management
 - Message dispatch
- **Goal**
 - Easing Experiments
 - Various Programming Paradigms

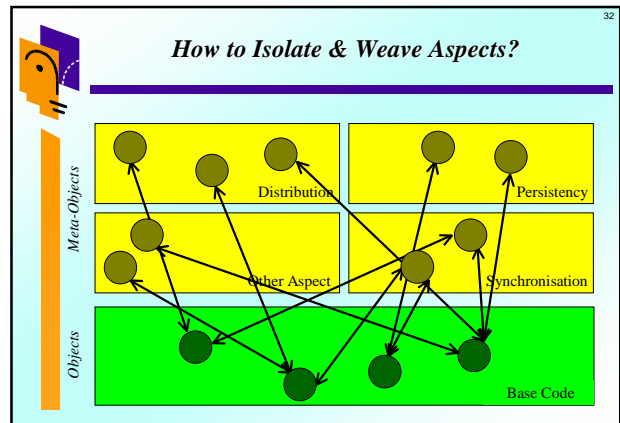
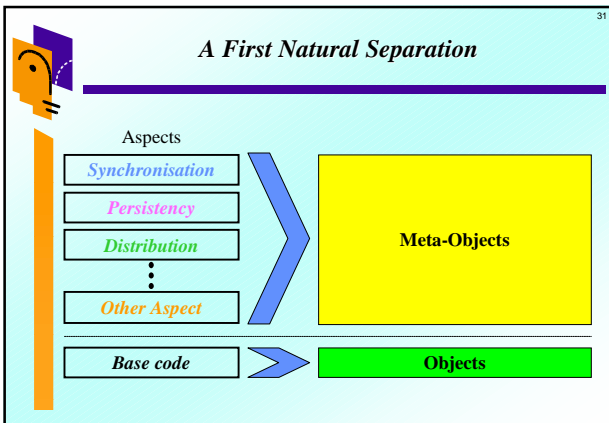
24

MetaclassTalk "Conceptual" Kernel

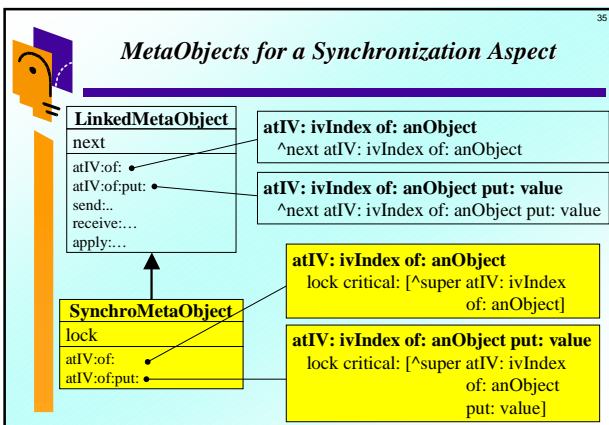
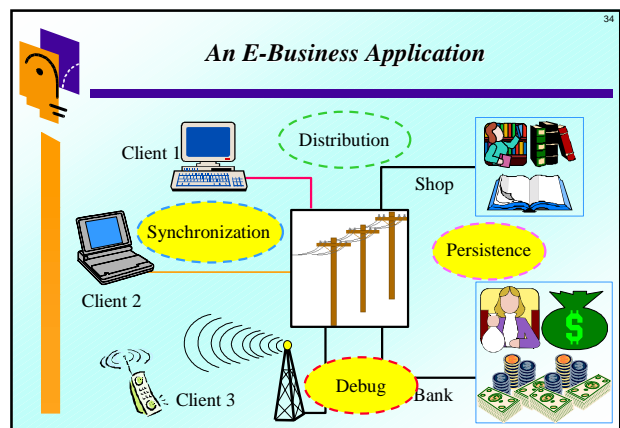
```

graph TD
    Object[Object] -- inherits from --> StandardClass[StandardClass]
    StandardClass -.->|is instance of| MetaObject[MetaObject]
    MetaObject -- default meta-object --> Default((default meta-object))
    Default -.->|meta| Object
  
```



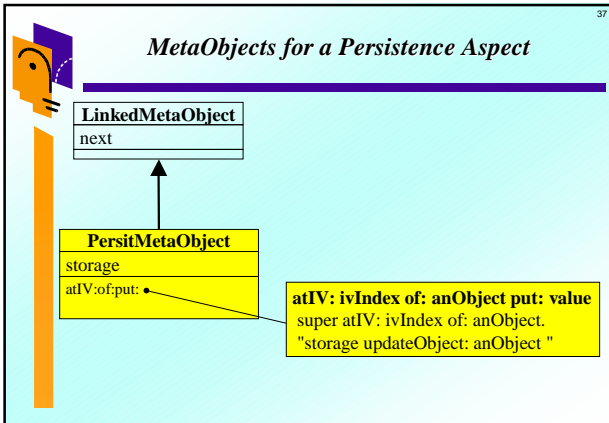
- ### Issues & Solutions
- **How to define isolated aspects?**
 - 1 Set of Meta-Objects per aspect
 - 1 meta-object participate to 1 aspect
 - **How to weave?**
 - The meta link
 - Meta-objects cooperation
- } Configuration Scripts



- ### Configuration Script for Synchronization Aspect
- **Link every book and order to a new synchronisation meta-object**

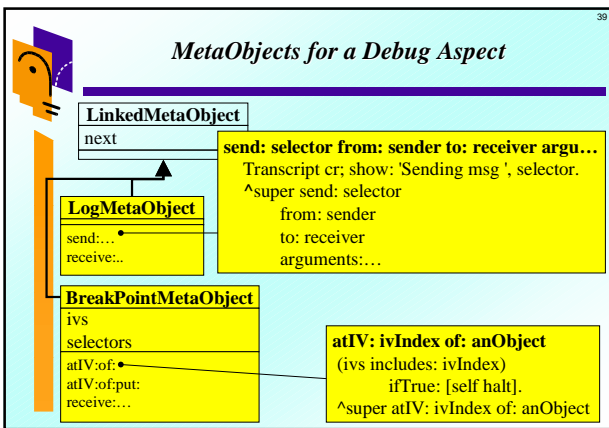
```
Book addMetaObjectClass: SynchroMetaObject.
Order addMetaObjectClass: SynchroMetaObject.
```

 - i.e. when a newInstance (book or order) is created
 1. |syncMeta|
 2. syncMeta := SynchroMetaObject new.
 3. newInstance addMetaObject: syncMeta



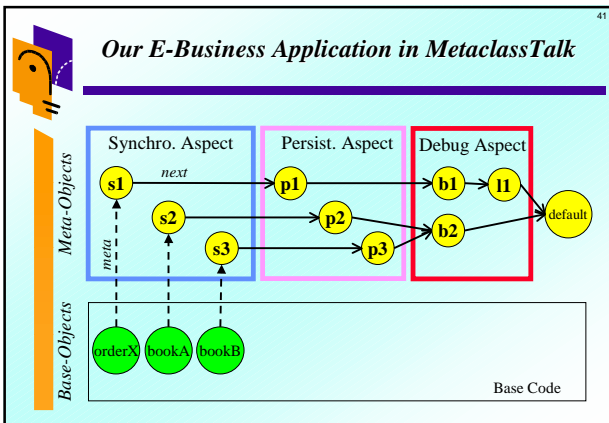
Configuration Script for Persistence Aspect

- **Link every book and order to a new persistence meta-object**
Book addMetaObjectClass: PersistMetaObject.
Order addMetaObjectClass: PersistMetaObject.
- i.e. when a newInstance (book or order) is created
 1. |persistMeta|
 2. persistMeta := PersistMetaObject new.
 3. newInstance addMetaObject: persistMeta



Configuration Script for a Debug Aspect

- **Link orderX to a log meta-object**
|b1|
b1 := BreakPointMetaObject new.
b1 ivs: #(items) selectors: #(totalPrice printOn:).
orderX addMetaObject: b1;
addMetaObject: LogMetaObject new
- **Link all books to a single break point meta-object**
|b2|
b2 := BreakPointMetaObject new.
b2 ivs: #(price) selectors: #(printOn:).
Book addInstanceMetaObject: b2.
Order addMetaObjectClass: PersistMetaObject.
- i.e. when a newBook is created
 - newBook addMetaObject: b2




Conclusion

43

Summary

- **Reflection Does Support AOP**
 - NO new language construct
 - Reusable aspects: generic meta-objects
 - Flexibility: dynamic meta-objects/aspects change
 - Aspects conflicts = meta-object composition problem
- **MetaclassTalk eases experiments**
 - AOP (using Meta-Objects)
 - Mixins (using Metaclasses)
 - ...
- **An implementation is available for Squeak 3.2**




44

Some Future Works

- **Other experiments**
 - Distribution
 - Software Components
 - Mutli-Agents Systems
 - (Strong) Mobility?...
- **Enhancing MetaclassTalk**
 - Improving performance (Code inlining?, Specific VM?)
 - Refactoring the full Smalltalk library
 - Migration to the latest Squeak release

45

Thanks for your attention Questions? Comments?



MetaclassTalk
Reflection & Meta-Programming
powered by Smalltalk

Documents & Download
<http://csl.ensm-douai.fr/MetaclassTalk>