UNIVERSITÄT
BERN

Model-Centric Software Adaptation

Oscar Nierstrasz

Software Composition Group
scg.unibe.ch

SVPP — 2008-08-09

Roadmap

- **Intro — Model-centric development**
- **Self-describing systems** *(Magritte)*
- **Fine-grained, unanticipated adaptation** *(Reflectivity)*
- **Bridging static and dynamic views** *(Hermion)*
- **Tracking change** *(Object flow)*
- **Scoping change** *(Changeboxes)*
- **Bringing models to code** *(Embedding DSLs)*

Roadmap

- **Intro — Model-centric development**
- Self-describing systems *(Magritte)*
- Fine-grained, unanticipated adaptation *(Reflectivity)*
- Bridging static and dynamic views *(Hermion)*
- Tracking change *(Object flow)*
- Scoping change *(Changeboxes)*
- Bringing models to code *(Embedding DSLs)*

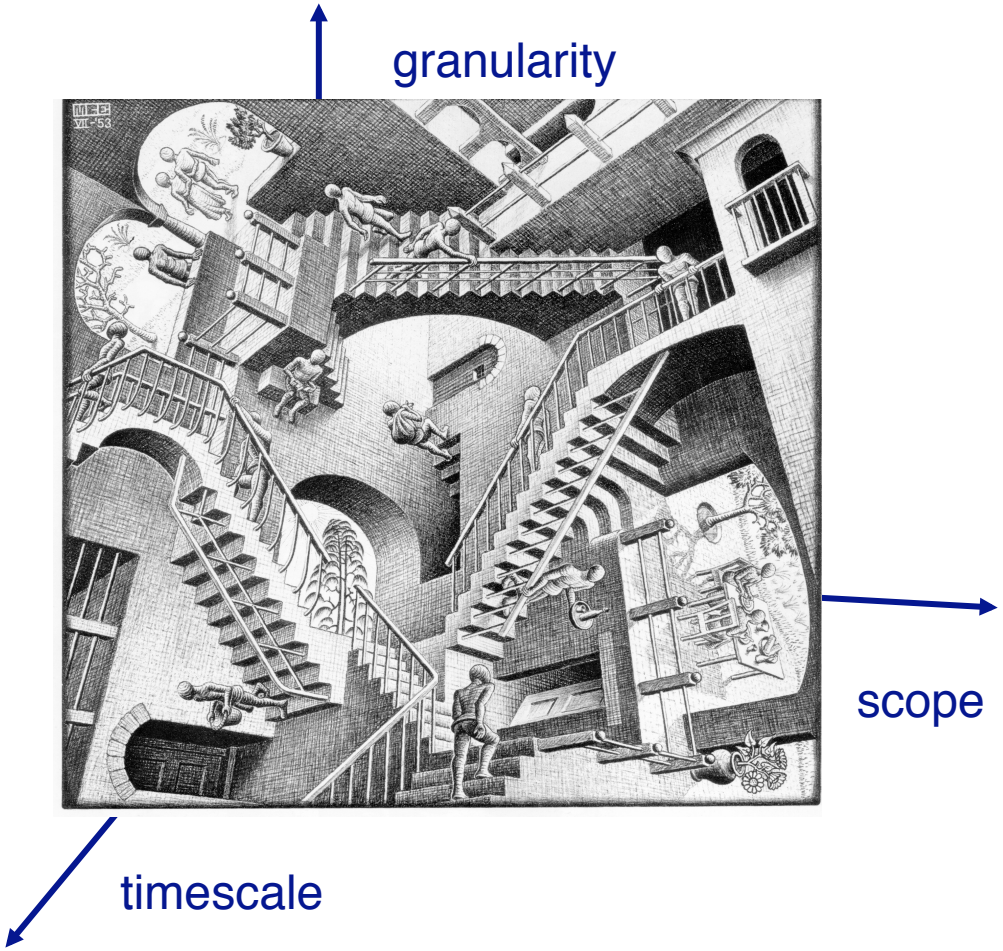
Team — scg.unibe.ch

Magritte	*Lukas Renggli*
Reflectivity	*Marcus Denker*
Hermion	*David Röthlisberger*
Object Flow	*Adrian Lienhard*
Changeboxes	*Pascal Zumkehr*
Embedding DSLs	*Lukas Renggli*
Other topics …	*Tudor Gîrba,* *Adrian Kuhn, Toon Verwaest*

Software inevitably changes, but …

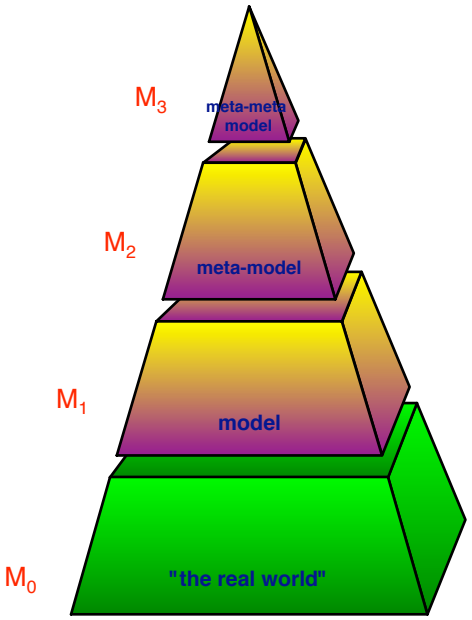
most programming languages and IDEs ***inhibit change*** rather than support it!

(Some) dimensions of change



- (Re-)configuration
- Bug fixes
- Refactoring
- New functionality
- Bridging versions
- Dynamic aspects
- Instrumentation
- Run-time adaptation
- …

Not model-driven, but *model-centric*



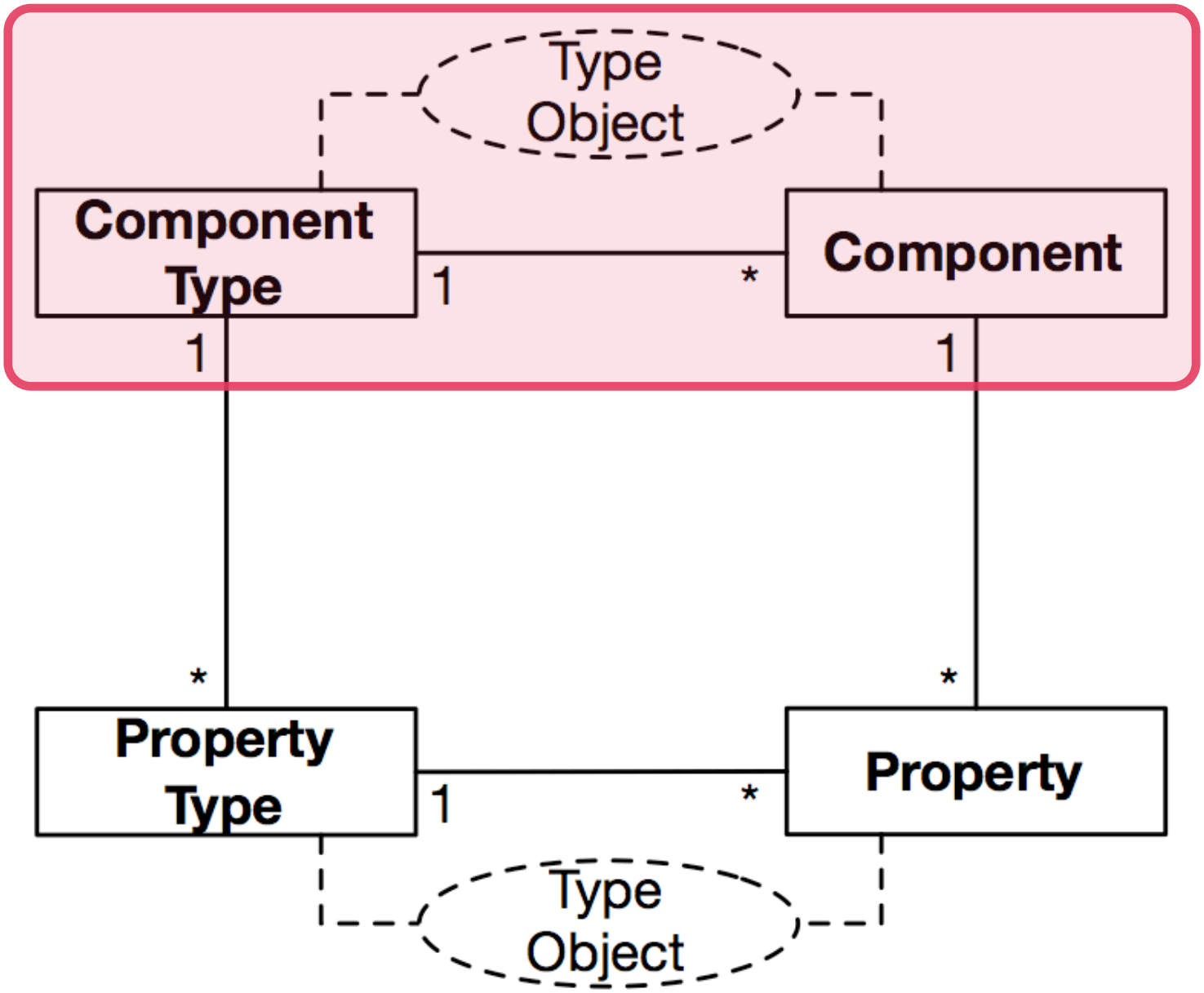
Not static, but *context-aware*

Roadmap

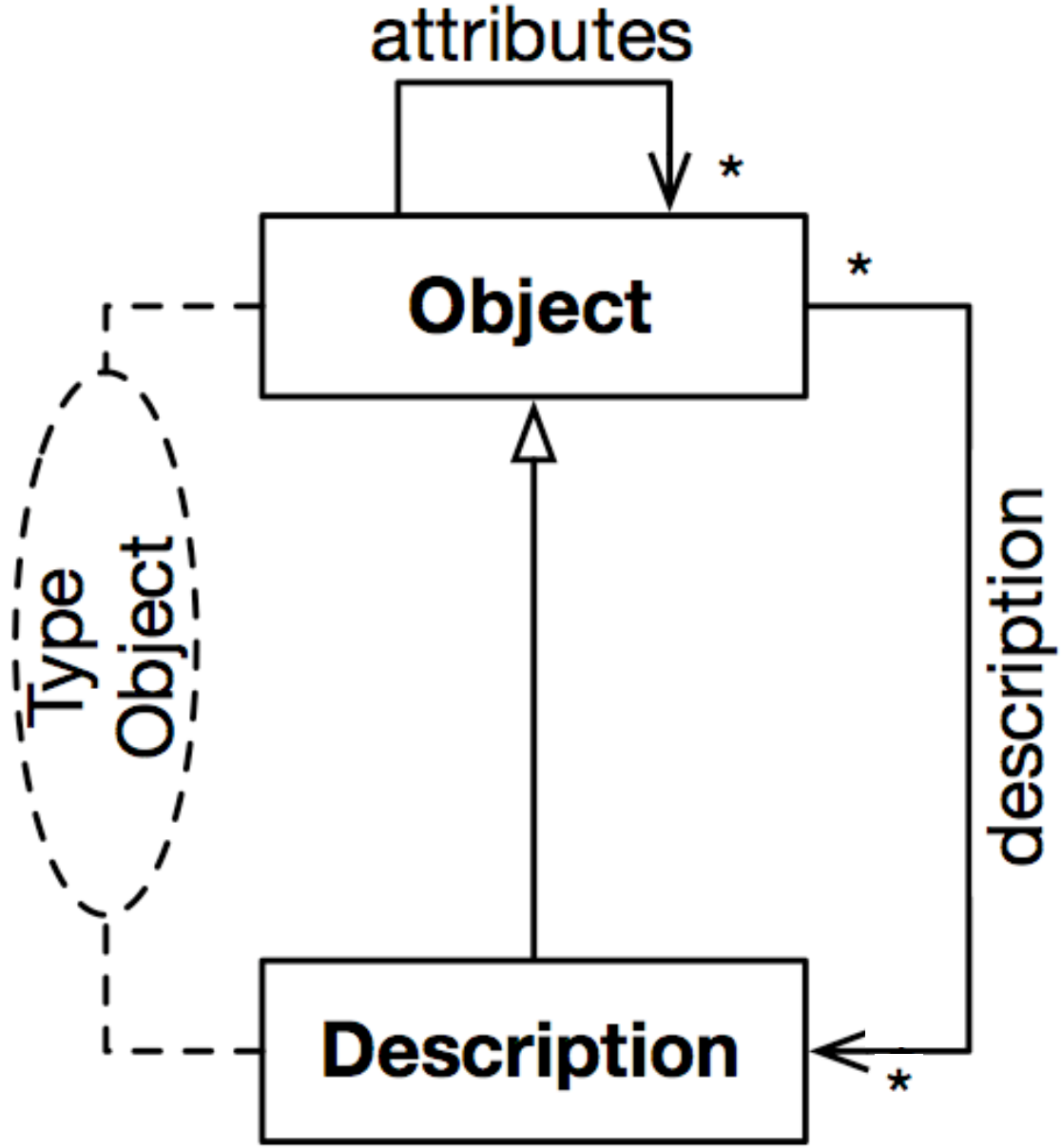
- Intro — Model-centric development
- **Self-describing systems *(Magritte)***
- Fine-grained, unanticipated adaptation *(Reflectivity)*
- Bridging static and dynamic views *(Hermion)*
- Tracking change *(Object flow)*
- Scoping change *(Changeboxes)*
- Bringing models to code *(Embedding DSLs)*

Magritte

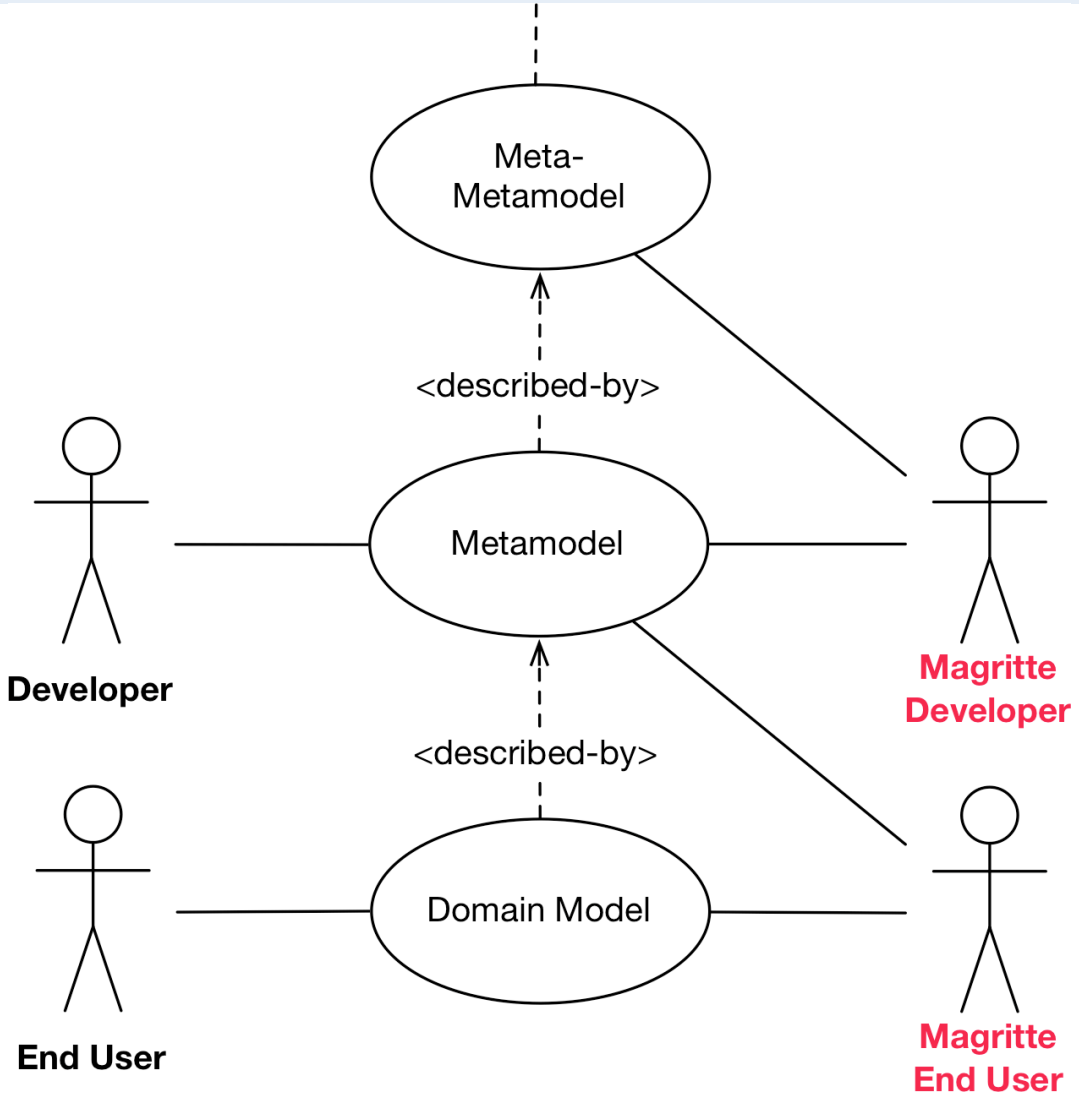

Magritte: A Meta-Driven Approach to Empower Developers and End Users
http://localhost:8080/seaside/pier/introduction
Google
Welcome > Introduction
Search Site
Introduction , Environment
Magritte
A Meta-Driven Approach to Empower Developers and End Users
Lukas Renggli, Stéphane Ducasse, Adrian Kuhn
[René Magritte, 1966] Decalcomania
View | Edit | Edit Meta



[Yoder *et al*, 2001] Architecture and design of adaptive object models


attributes
*
Object
*
Type
Object
description
Description
*

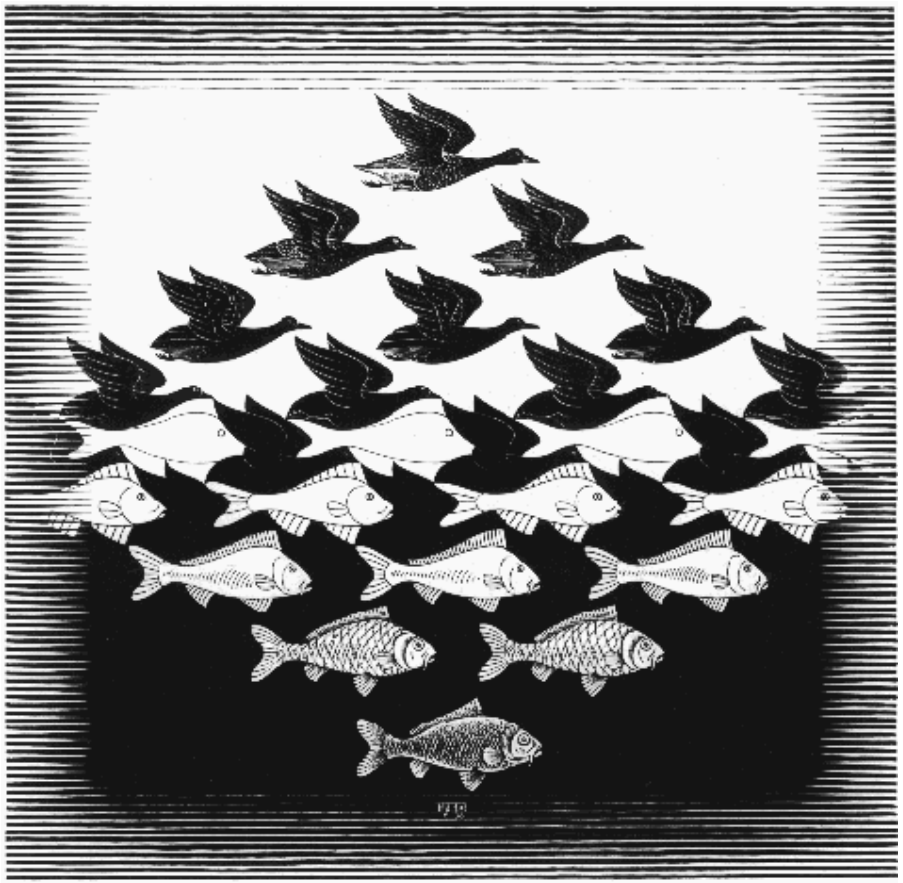
Magritte — meta-descriptions enable dynamic change



Roadmap

- Intro — Model-centric development
- Self-describing systems *(Magritte)*
- **Fine-grained, unanticipated adaptation *(Reflectivity)***
- Bridging static and dynamic views *(Hermion)*
- Tracking change *(Object flow)*
- Scoping change *(Changeboxes)*
- Bringing models to code *(Embedding DSLs)*

How to change a running system?

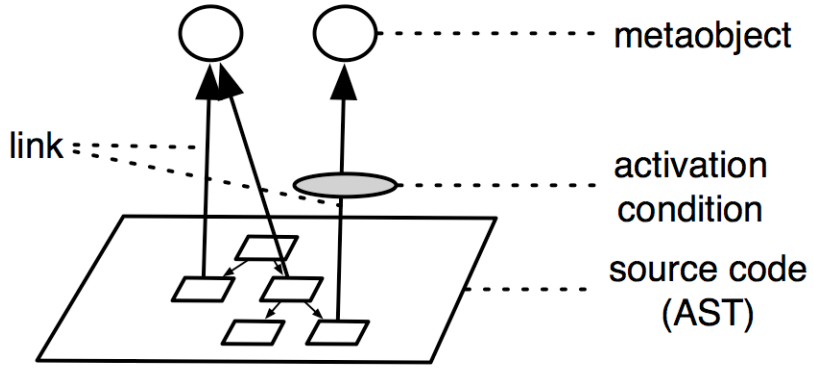


Unanticipated

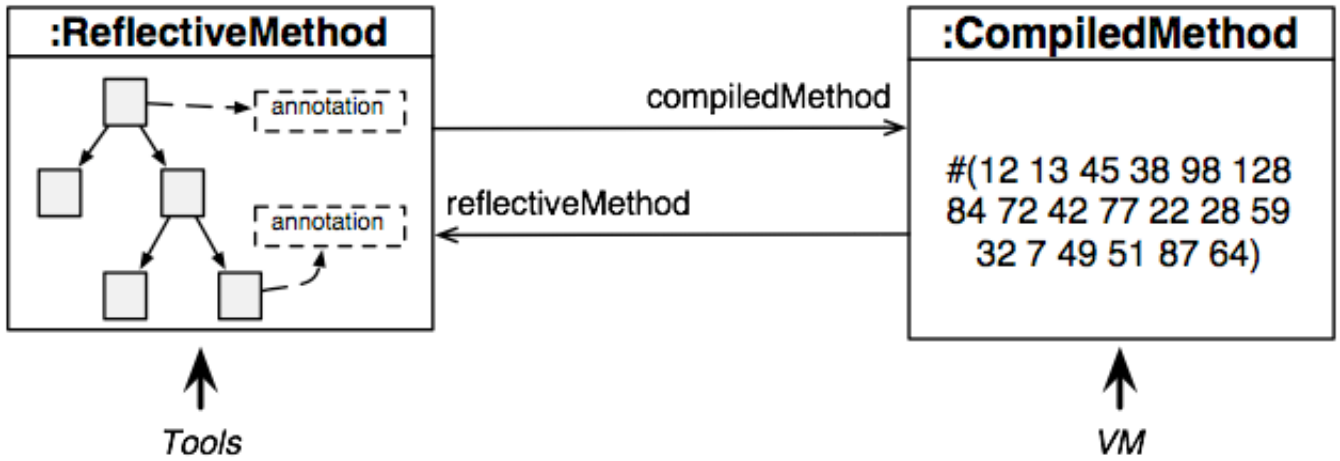
Arbitrary granularity

Geppetto — dynamic adaptation through partial behavioural reflection

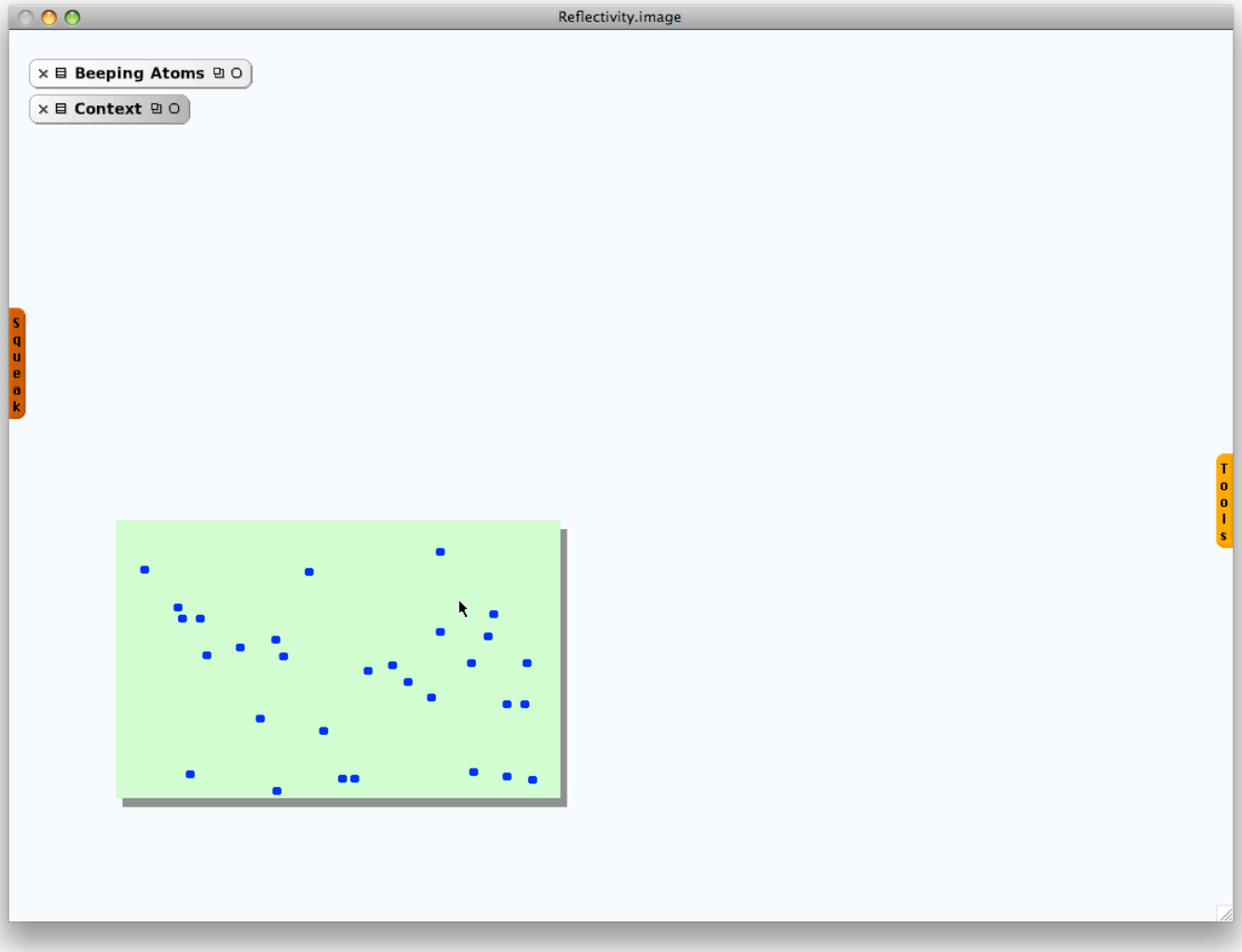
Partial behavioural reflection



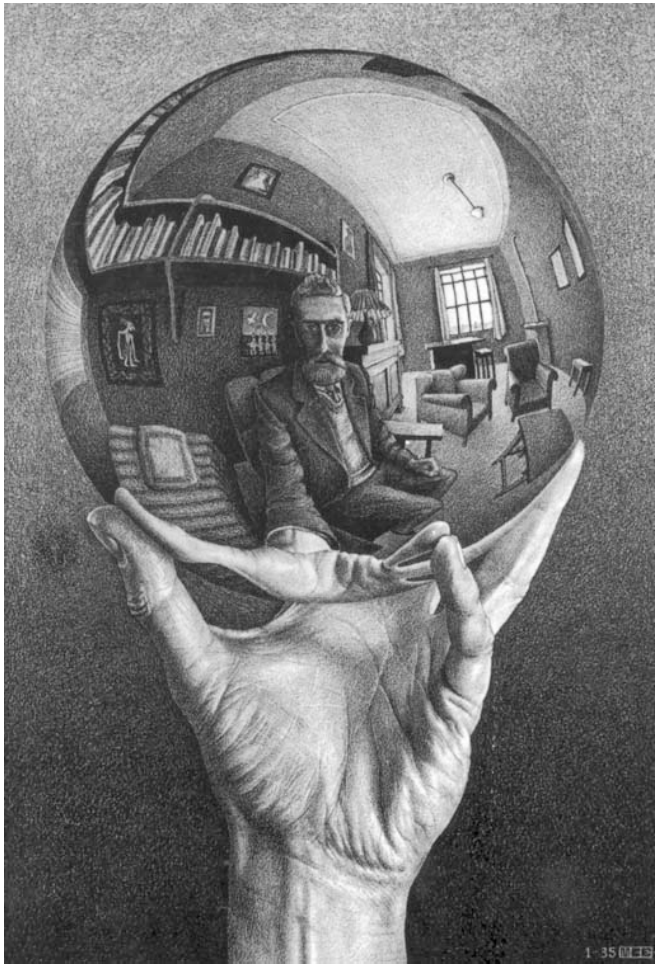
"Evil twin"



Reflectivity and Geppetto



Context

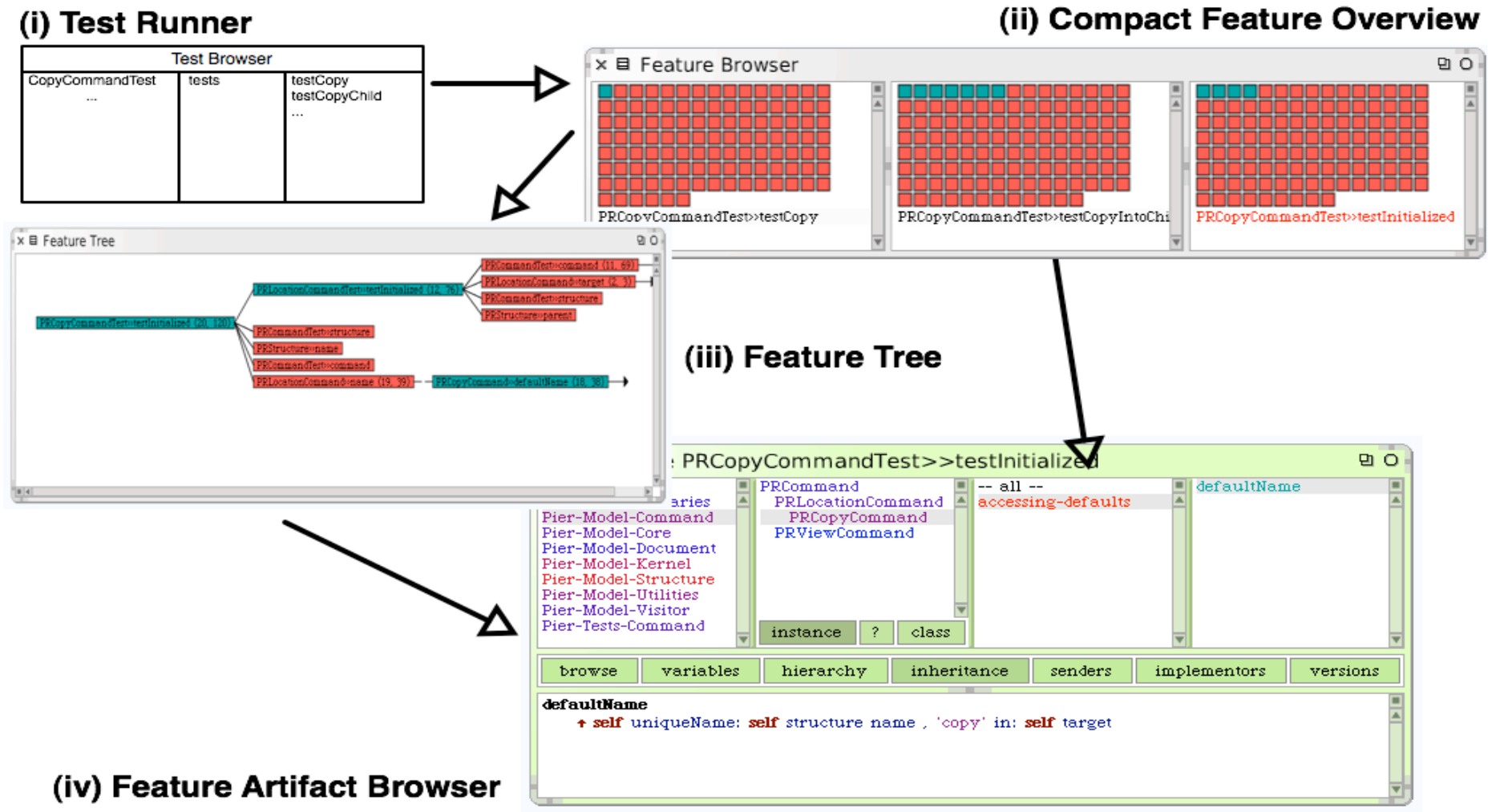


Reflection can be *scoped*
to the base level
(or to the meta-level …)

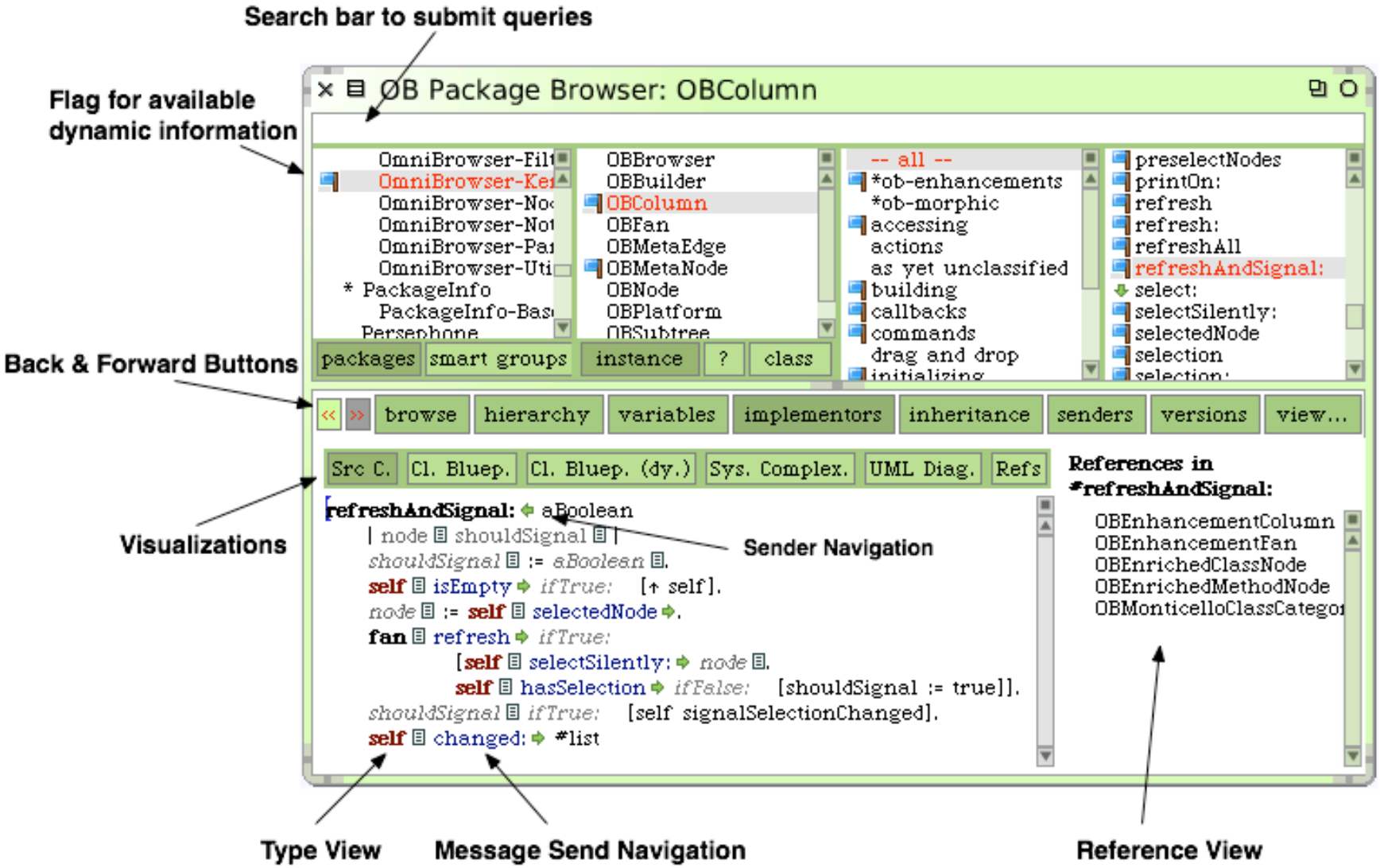
Roadmap

- Intro — Model-centric development
- Self-describing systems *(Magritte)*
- Fine-grained, unanticipated adaptation *(Reflectivity)*
- **Bridging static and dynamic views *(Hermion)***
- Tracking change *(Object flow)*
- Scoping change *(Changeboxes)*
- Bringing models to code *(Embedding DSLs)*

Hermion — combining static and dynamic information in the IDE



Enriching source artifacts with dynamic information

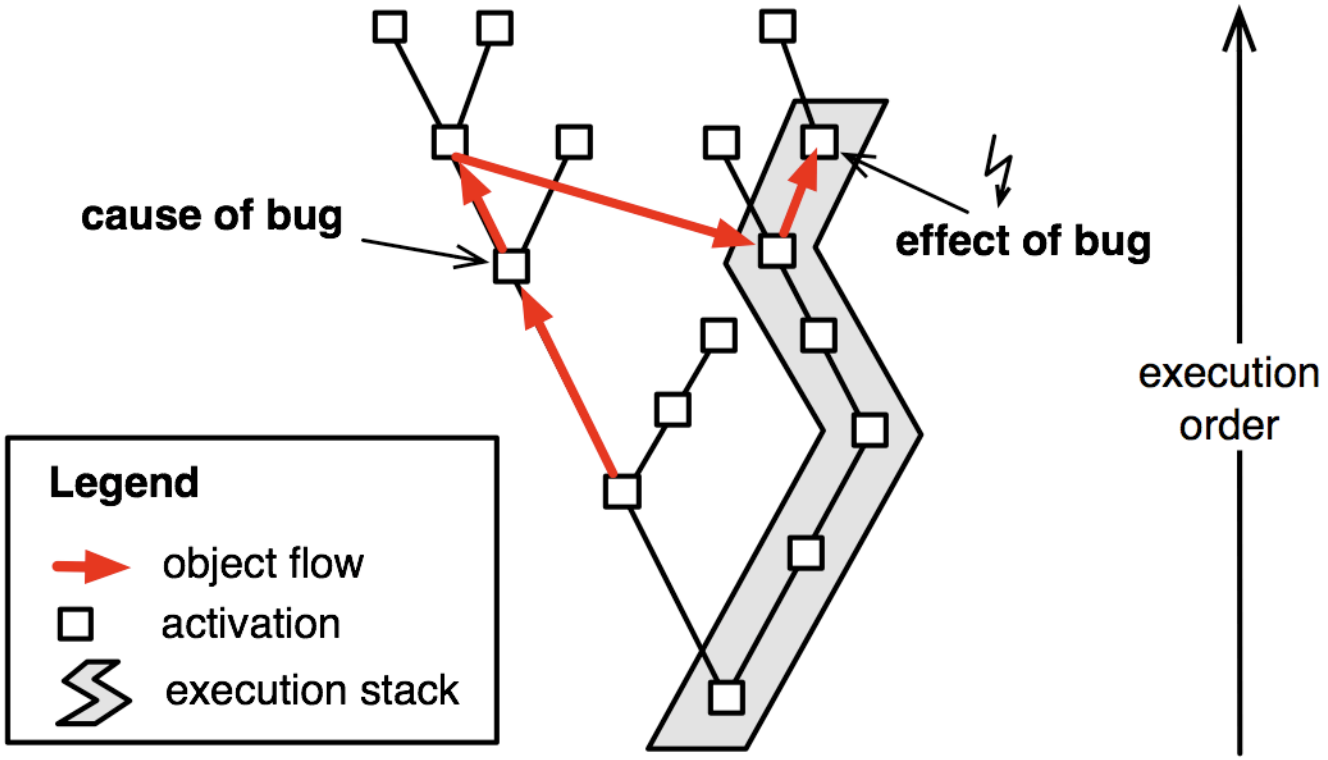


Roadmap

- Intro — Model-centric development
- Self-describing systems *(Magritte)*
- Fine-grained, unanticipated adaptation *(Reflectivity)*
- Bridging static and dynamic views *(Hermion)*
- **Tracking change *(Object flow)***
- Scoping change *(Changeboxes)*
- Bringing models to code *(Embedding DSLs)*

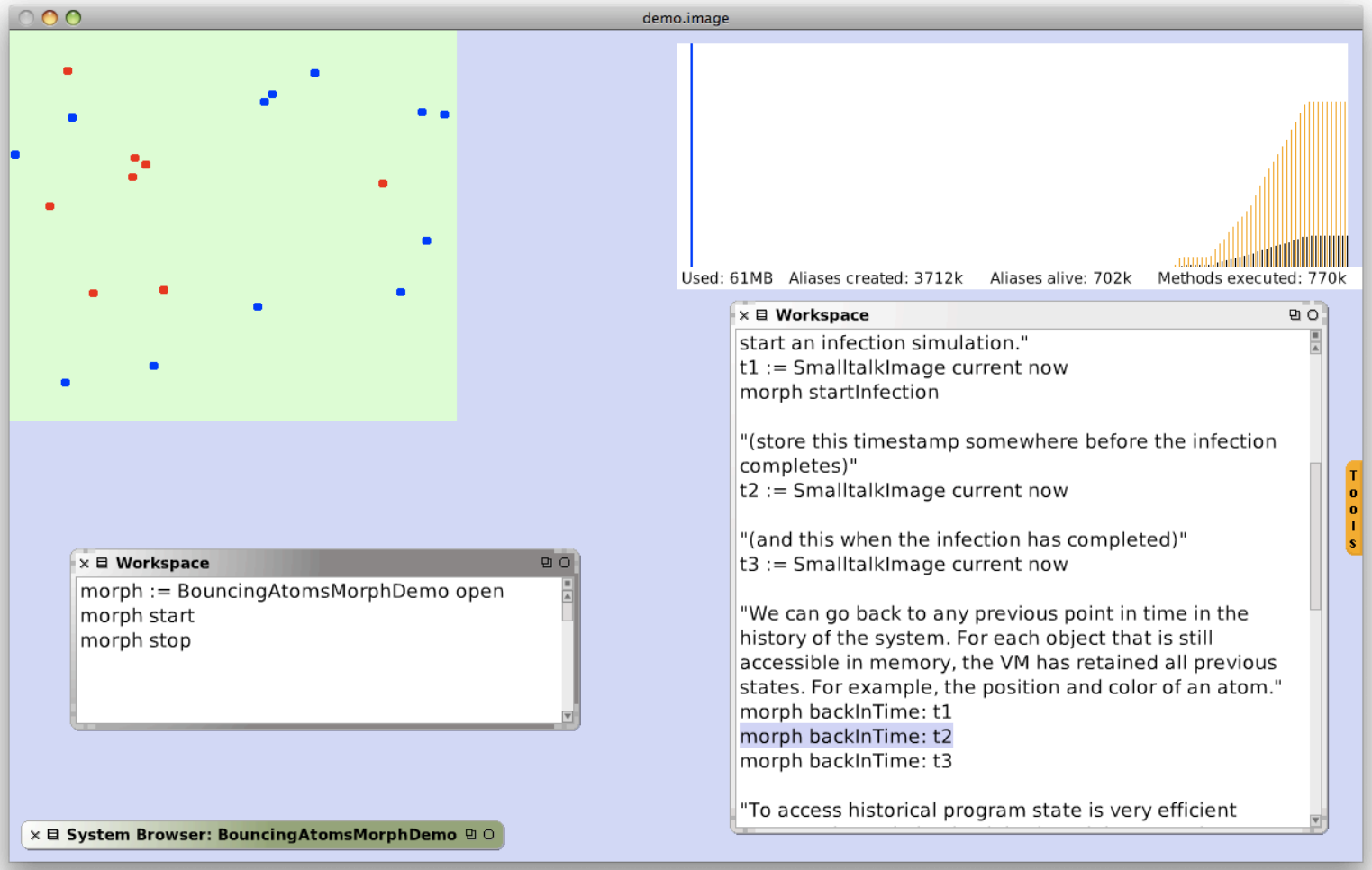
How to track down defects when the offending context is gone?

Object Flow Analysis

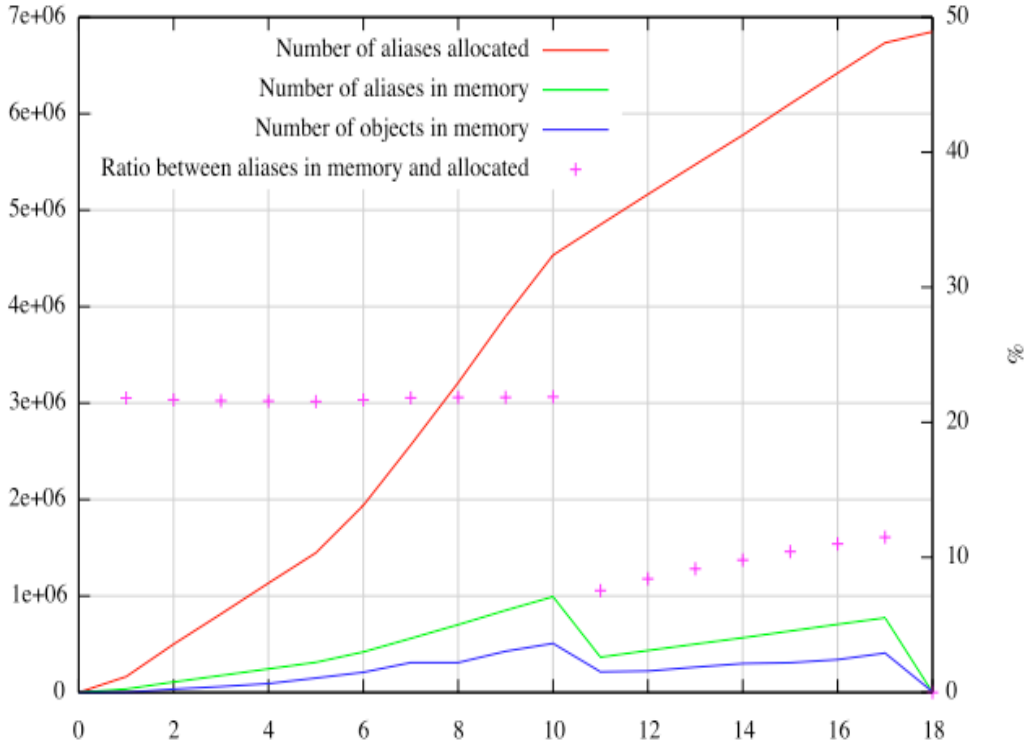


Use ***first-class aliases*** to track object flow

A back-in-time VM with object flow analysis



Selective memory



Remember only what is needed!

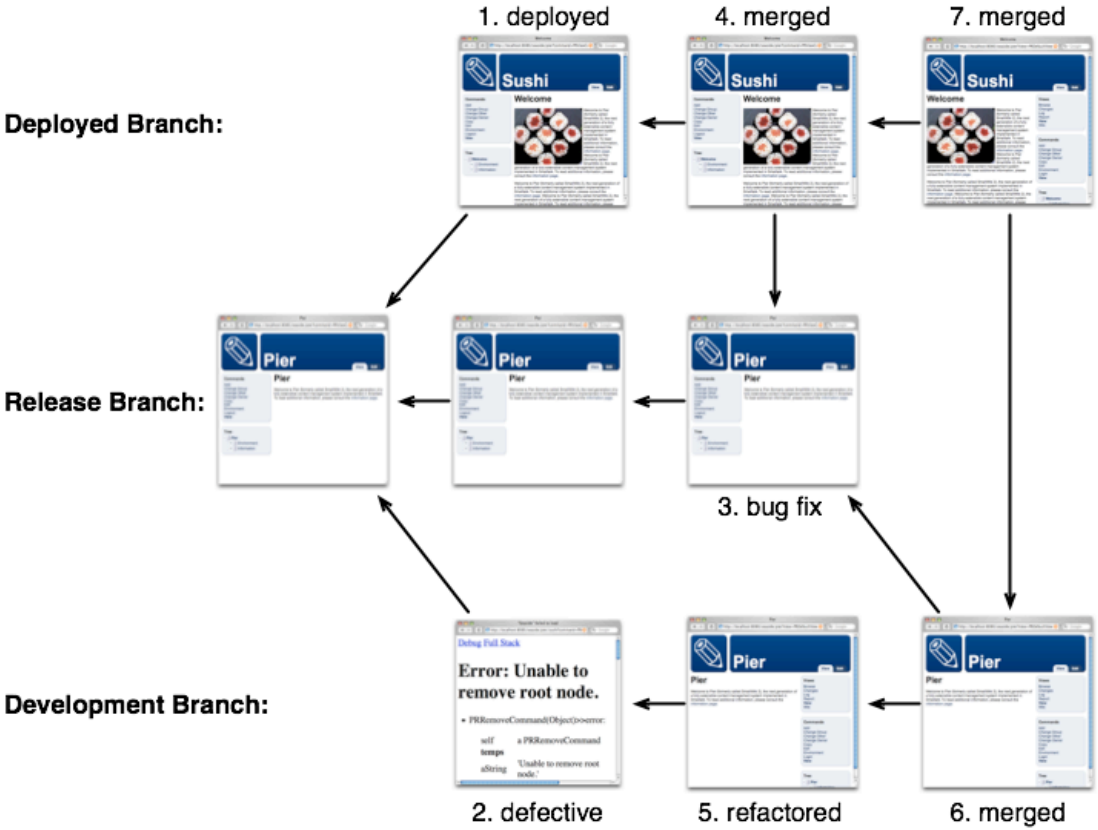
> record aliases and past states as regular objects

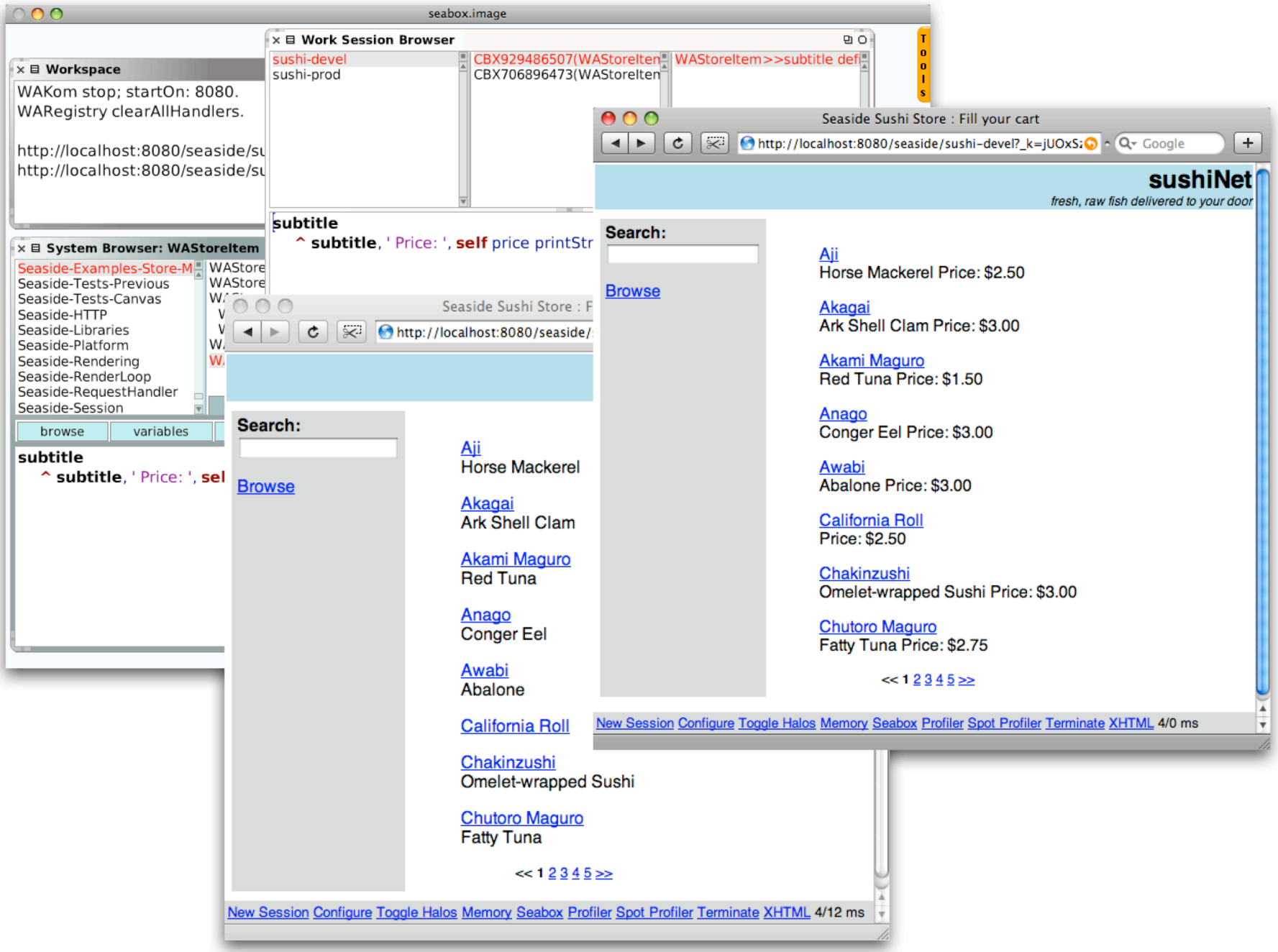
> GC forgets them when no longer needed

Roadmap

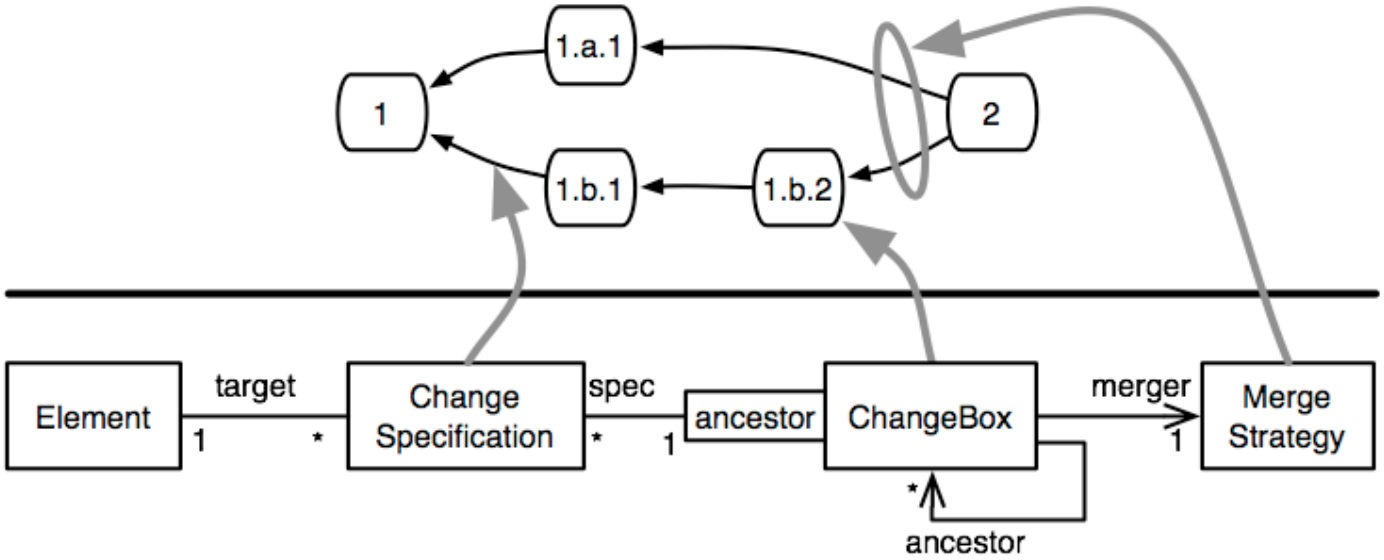
- Intro — Model-centric development
- Self-describing systems *(Magritte)*
- Fine-grained, unanticipated adaptation *(Reflectivity)*
- Bridging static and dynamic views *(Hermion)*
- Tracking change *(Object flow)*
- **Scoping change *(Changeboxes)***
- Bringing models to code *(Embedding DSLs)*

Changeboxes — encapsulate and manage change in a running system




seabox.image
Workspace
WAKom stop; startOn: 8080.
WARegistry clearAllHandlers.
http://localhost:8080/seaside/su
http://localhost:8080/seaside/su
Work Session Browser
sushi-devel
sushi-prod
CBX929486507(WAStoreIten
CBX706896473(WAStoreIten
WAStoreItem>>subtitle def
Tools
subtitle
^ subtitle, ' Price: ', self price printStr
System Browser: WAStoreItem
Seaside-Examples-Store-M
Seaside-Tests-Previous
Seaside-Tests-Canvas
Seaside-HTTP
Seaside-Libraries
Seaside-Platform
Seaside-Rendering
Seaside-RenderLoop
Seaside-RequestHandler
Seaside-Session
browse
variables
subtitle
^ subtitle, ' Price: ', sel
Seaside Sushi Store : F
http://localhost:8080/seaside/
Search:
Browse
Aji
Horse Mackerel
Akagai
Ark Shell Clam
Akami Maguro
Red Tuna
Anago
Conger Eel
Awabi
Abalone
California Roll
Chakinzushi
Omelet-wrapped Sushi
Chutoro Maguro
Fatty Tuna
<< 1 2 3 4 5 >>
New Session Configure Toggle Halos Memory Seabox Profiler Spot Profiler Terminate XHTML 4/12 ms
Seaside Sushi Store : Fill your cart
http://localhost:8080/seaside/sushi-devel?_k=jUOxSz
Google
sushiNet
fresh, raw fish delivered to your door
Search:
Browse
Aji
Horse Mackerel Price: $2.50
Akagai
Ark Shell Clam Price: $3.00
Akami Maguro
Red Tuna Price: $1.50
Anago
Conger Eel Price: $3.00
Awabi
Abalone Price: $3.00
California Roll
Price: $2.50
Chakinzushi
Omelet-wrapped Sushi Price: $3.00
Chutoro Maguro
Fatty Tuna Price: $2.75
<< 1 2 3 4 5 >>
New Session Configure Toggle Halos Memory Seabox Profiler Spot Profiler Terminate XHTML 4/0 ms

Changeboxes in a nutshell

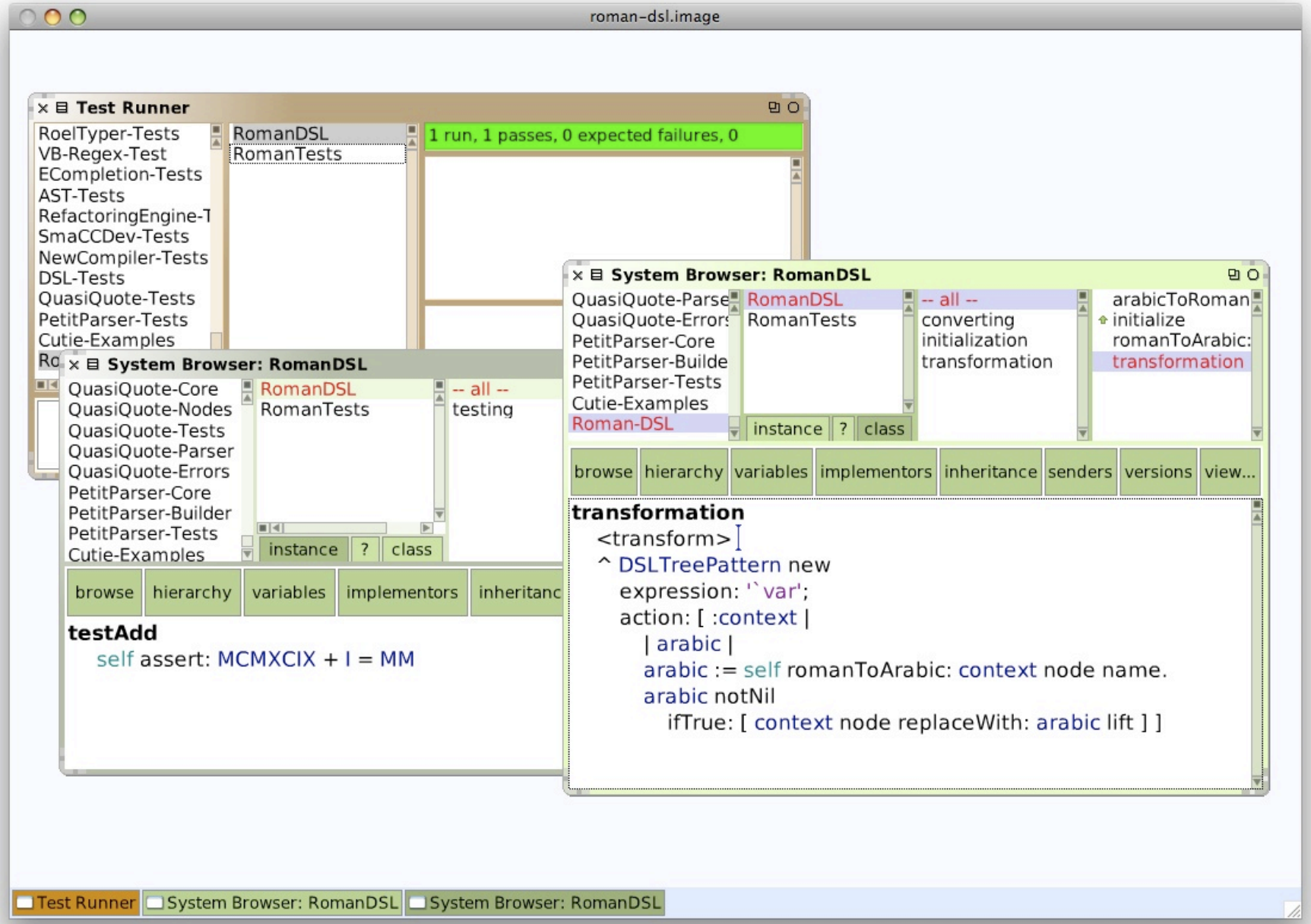


Roadmap

- Intro — Model-centric development
- Self-describing systems *(Magritte)*
- Fine-grained, unanticipated adaptation *(Reflectivity)*
- Bridging static and dynamic views *(Hermion)*
- Tracking change *(Object flow)*
- Scoping change *(Changeboxes)*
- **Bringing models to code *(Embedding DSLs)***

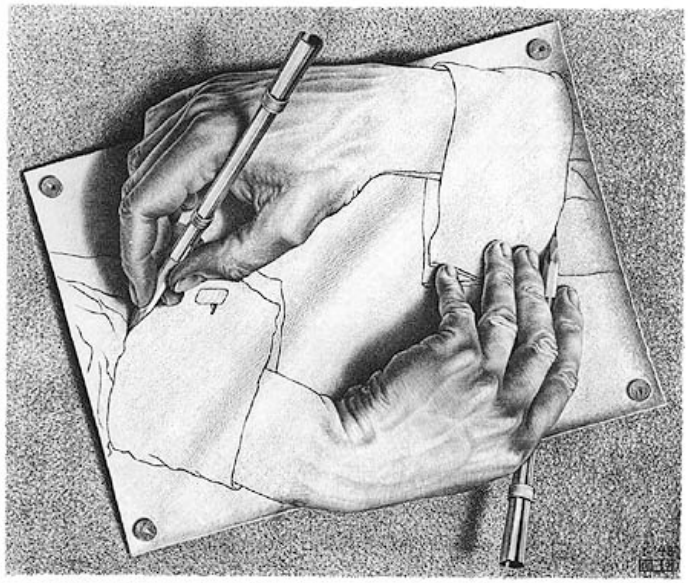
Embedding Domain Models in Code

Make DSLs ***first class citizens*** of their host language


roman-dsl.image
Test Runner
RoelTyper-Tests
VB-Regex-Test
ECompletion-Tests
AST-Tests
RefactoringEngine-T
SmaCCDev-Tests
NewCompiler-Tests
DSL-Tests
QuasiQuote-Tests
PetitParser-Tests
Cutie-Examples
RomanDSL
RomanTests
1 run, 1 passes, 0 expected failures, 0
System Browser: RomanDSL
QuasiQuote-Core
QuasiQuote-Nodes
QuasiQuote-Tests
QuasiQuote-Parser
QuasiQuote-Errors
PetitParser-Core
PetitParser-Builder
PetitParser-Tests
Cutie-Examples
RomanDSL
RomanTests
-- all --
testing
instance
?
class
browse
hierarchy
variables
implementors
inheritanc
testAdd
self assert: MCMXCIX + I = MM
System Browser: RomanDSL
QuasiQuote-Parse
QuasiQuote-Errors
PetitParser-Core
PetitParser-Builde
PetitParser-Tests
Cutie-Examples
Roman-DSL
RomanDSL
RomanTests
-- all --
converting
initialization
transformation
arabicToRoman
initialize
romanToArabic:
transformation
instance
?
class
browse
hierarchy
variables
implementors
inheritance
senders
versions
view...
transformation
<transform>
^ DSLTreePattern new
expression: '`var';
action: [:context |
| arabic |
arabic := self romanToArabic: context node name.
arabic notNil
ifTrue: [context node replaceWith: arabic lift]]
Test Runner
System Browser: RomanDSL
System Browser: RomanDSL

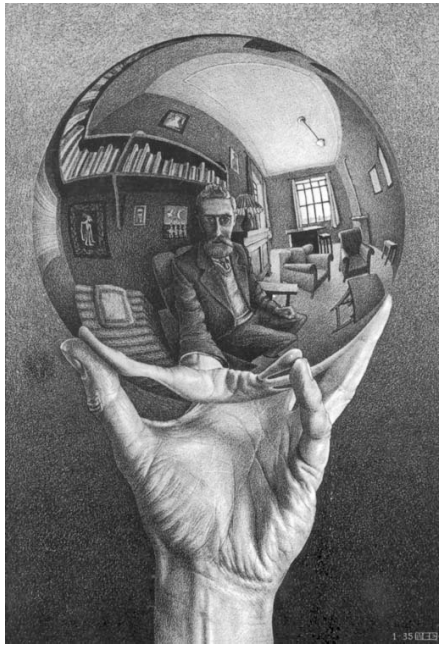
Conclusions

Systems that support change need to be *model-centric* and *context-aware*



First-class meta-descriptions
High-level, fine-grained reflection
Run-time annotations

Scope changes to:
- base/meta levels
- individual clients
- …



Where do we go from here?

From model-centric to virtual worlds?