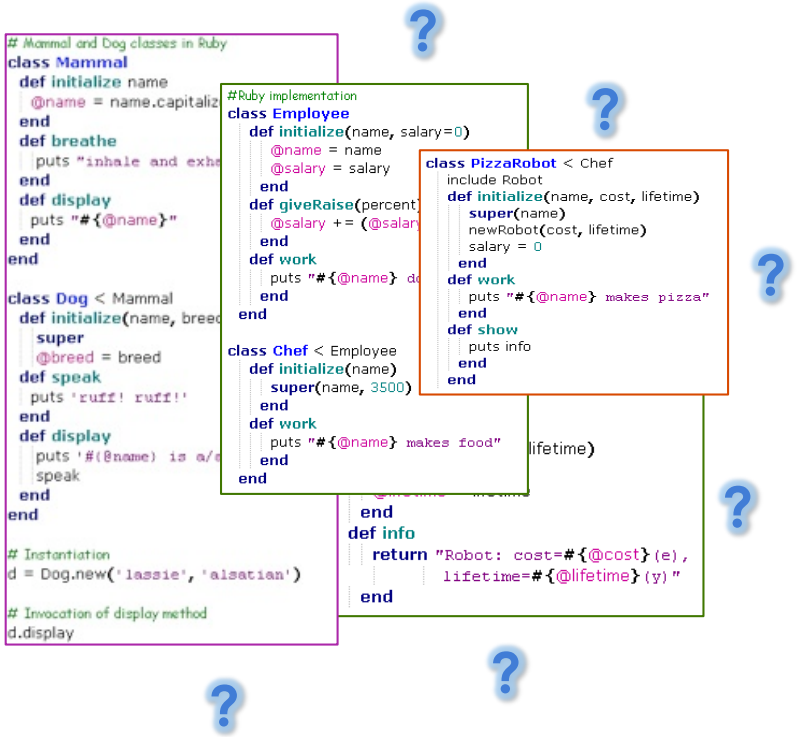
Experiments with Pro-Active Declarative Meta-Programming

Verónica Uquillas Gómez
Andy Kellens
Kris Gybels
Theo D'Hondt

International Workshop on Smalltalk Technologies
31st of August 2009, Brest - France
ESUG 2009

Software Languages Lab
Vrije Universiteit Brussel
Belgium

Problem

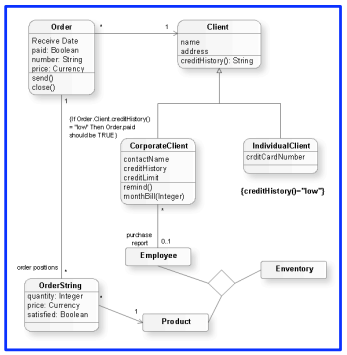


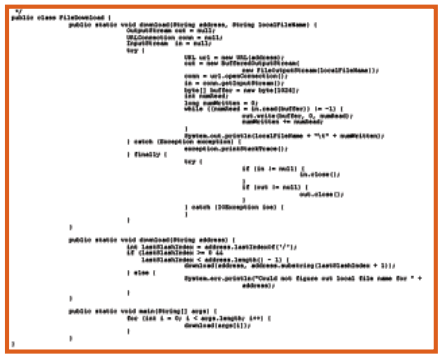
```ruby
# Mammal and Dog classes in Ruby
class Mammal
  def initialize name
    @name = name.capitaliz
  end
  def breathe
    puts "inhale and exha
  end
  def display
    puts "#{@name}"
  end
end

class Dog < Mammal
  def initialize(name, breed
    super
    @breed = breed
  def speak
    puts 'ruff! ruff!'
  end
  def display
    puts '#{@name} is a/
    speak
  end
end

# Instantiation
d = Dog.new('lassie', 'alsatian')

# Invocation of display method
d.display
```

```ruby
#Ruby implementation
class Employee
  def initialize(name, salary=0)
    @name = name
    @salary = salary
  end
  def giveRaise(percent)
    @salary += (@salary
  end
  def work
    puts "#{@name} do
  end
end

class Chef < Employee
  def initialize(name)
    super(name, 3500)
  end
  def work
    puts "#{@name} makes food"
  end
end
```

```ruby
class PizzaRobot < Chef
  include Robot
  def initialize(name, cost, lifetime)
    super(name)
    newRobot(cost, lifetime)
    salary = 0
  end
  def work
    puts "#{@name} makes pizza"
  end
  def show
    puts info
  end
end
```

```ruby
lifetime)
  end
def info
  return "Robot: cost=#{@cost}(e),
          lifetime=#{@lifetime}(y)"
  end
```





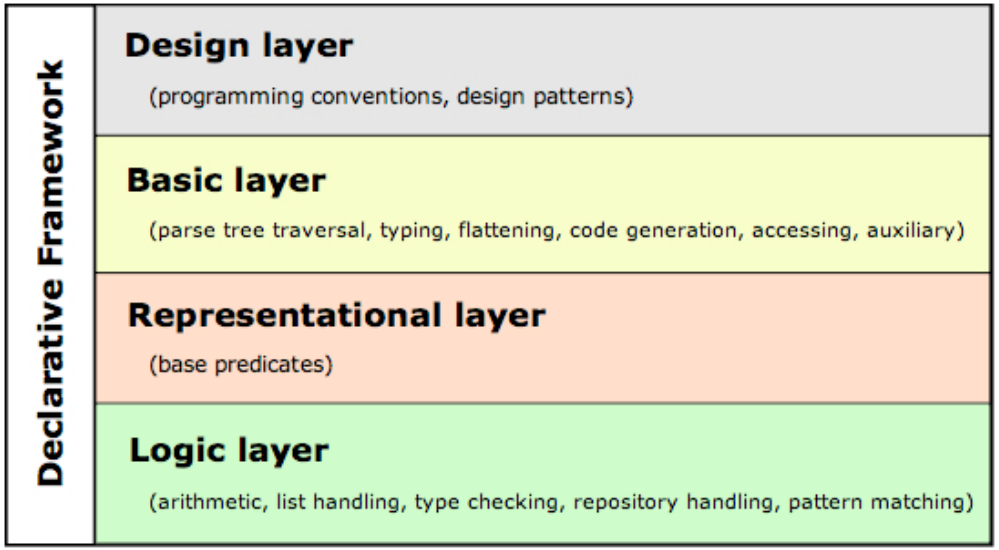
Co-evolution

SOUL

- Developer-driven (☒ pro-active)
- Snapshot-based reasoning (☒ time)

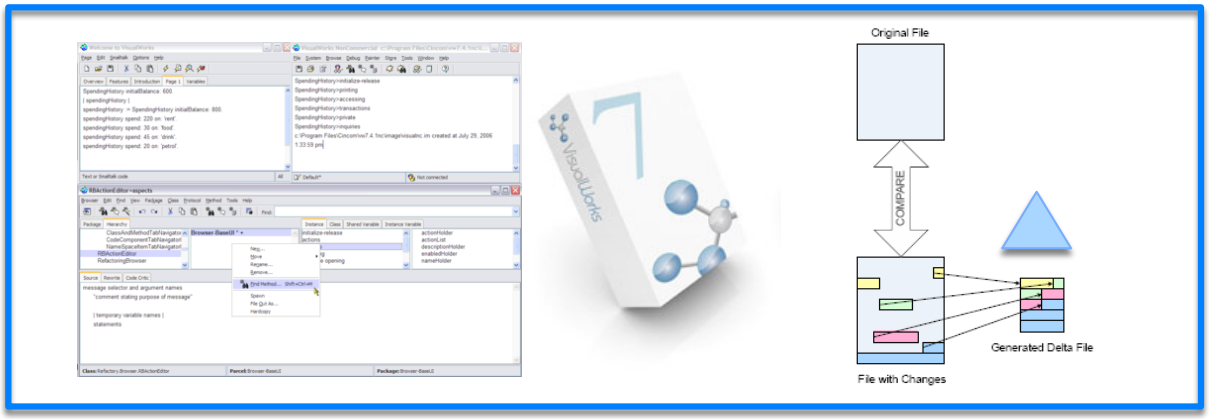
SOUL: Smalltalk Open Unification Language

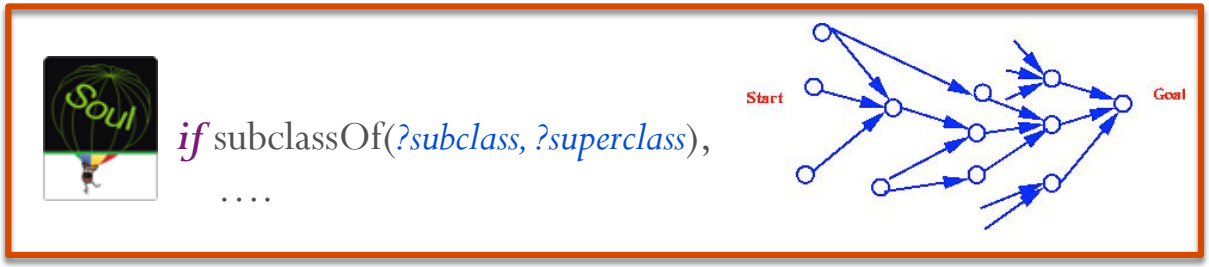
- Query language
- Prolog-like
- Smalltalk, Java, C(++), Cobol

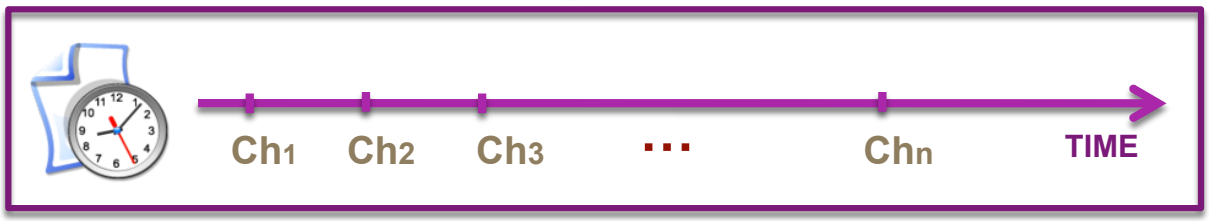



```
class(?class) if
    member(?class, [Smalltalk allClasses])
```

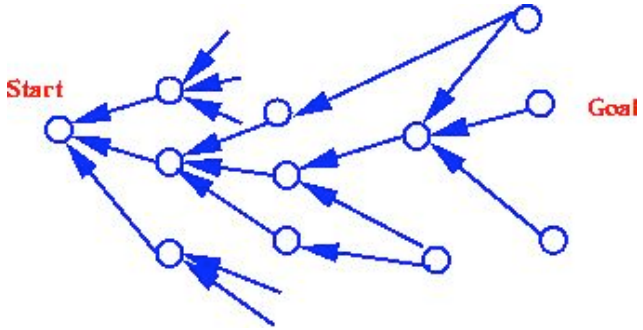
PARACHUT: Programming And Reasoning About CHanges Using Time



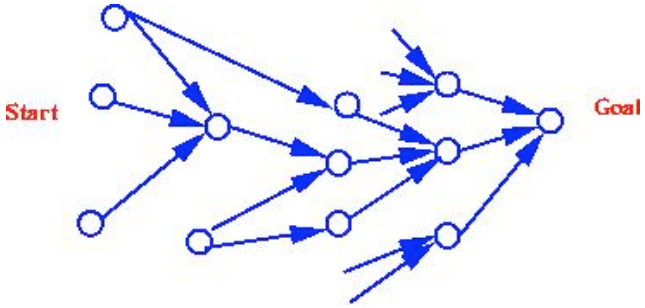




Backward Chaining vs. Forward Chaining



Goal-driven



Data-driven

- More possible start states or goal states?
- What kind of events triggers problem-solving?

Example: hash – equal rule

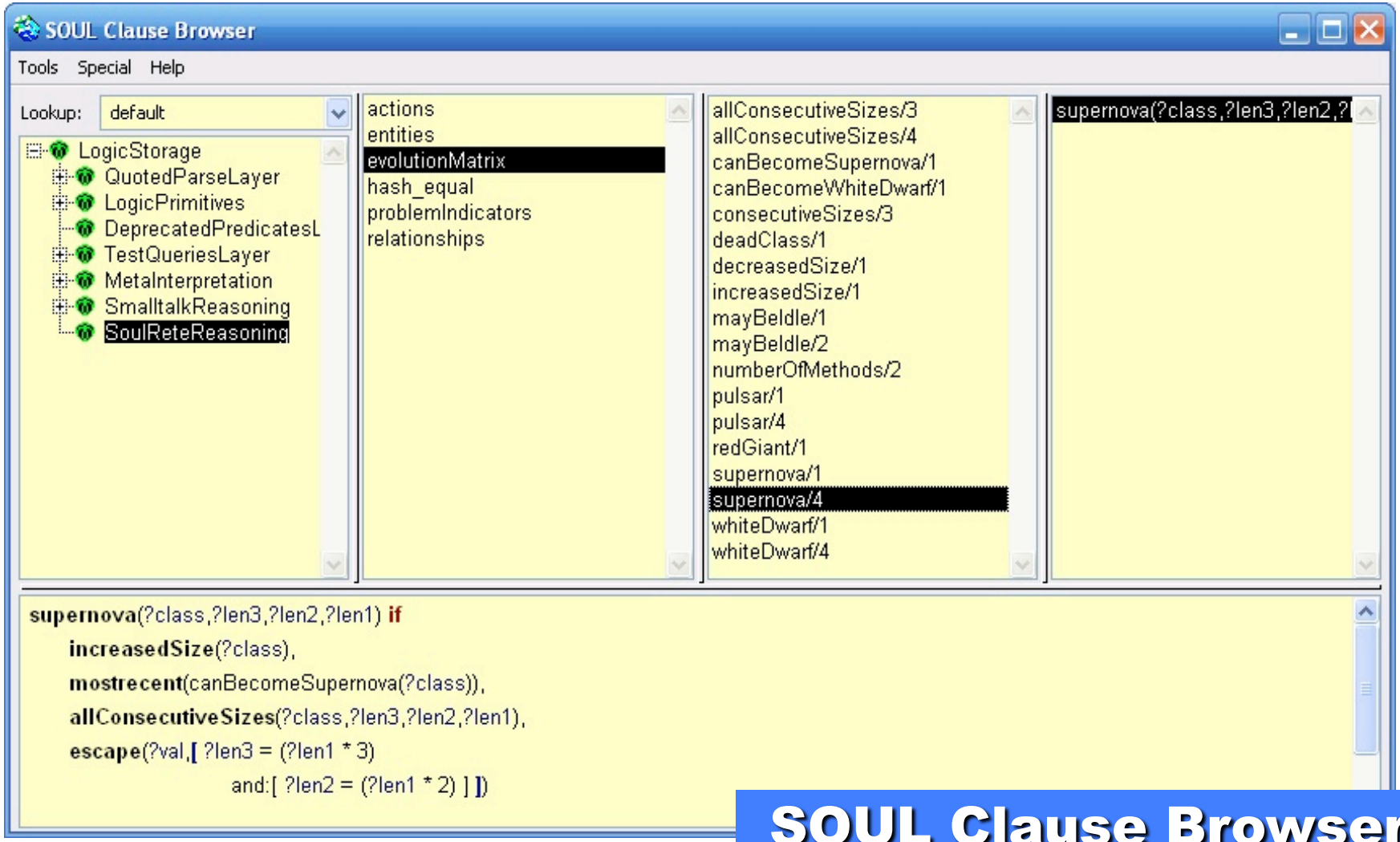
"re-implementing the = method implies re-implementing the hash method"

classNeedsHash(*?class*) *if*
    methodInClass(*?class*, =),
    not(methodInClass(*?class*, hash))

classNeedsEqual(*?class*) *if*
    methodInClass(*?class*, hash),
    not(methodInClass(*?class*, =))

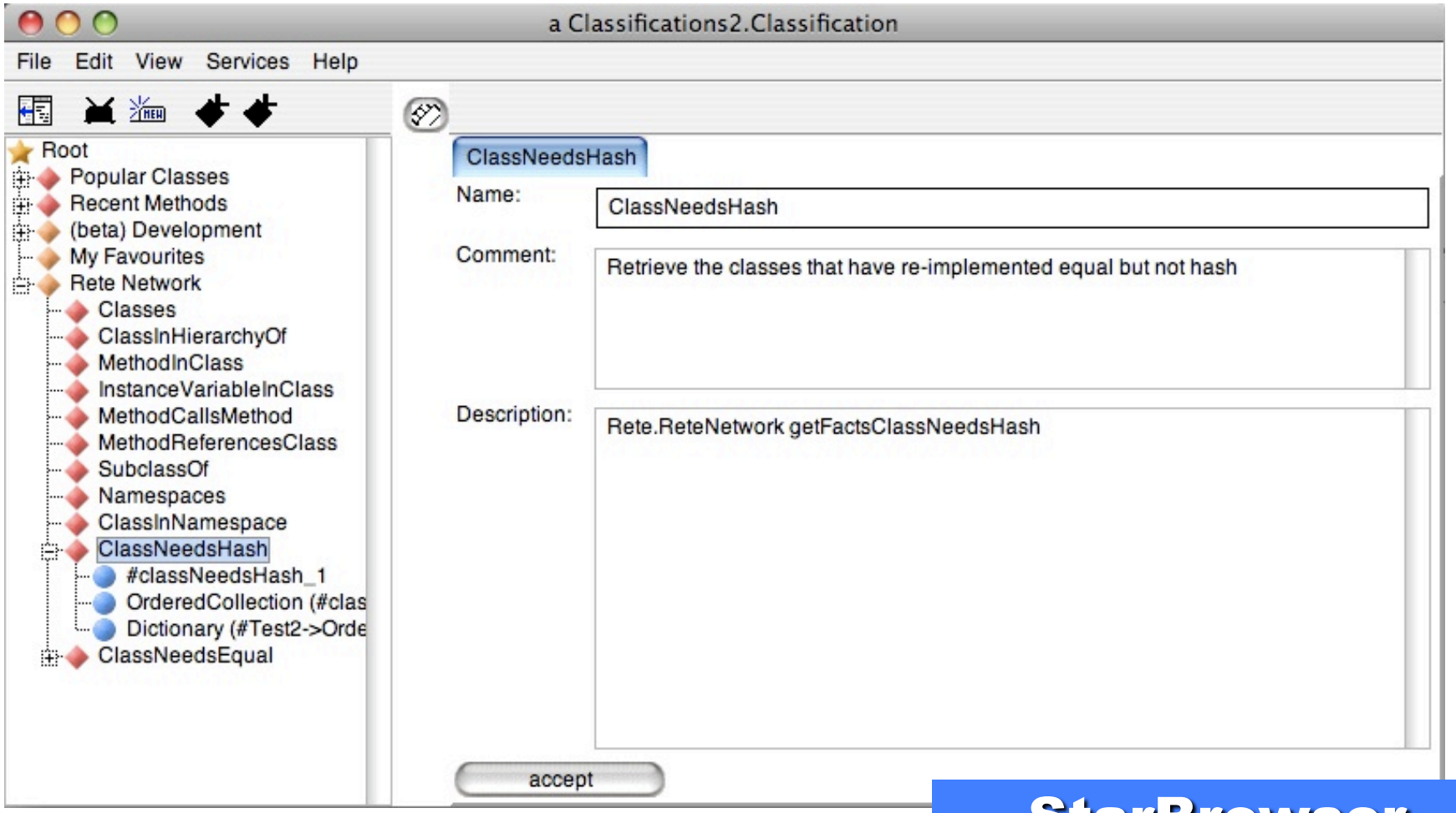
- class
- instanceVariableInClass
- methodInClass
- methodCallsMethod
- classInPackage
- subclassOf
- classInHierarchyOf

PARACHUT: defining the declarative queries



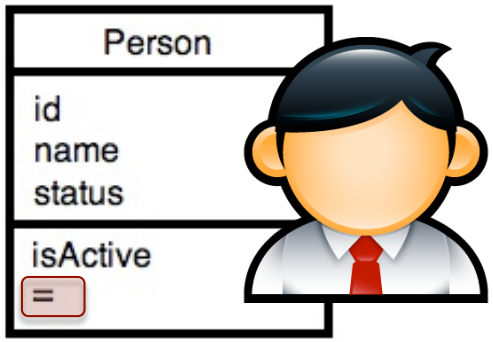
SOUL Clause Browser

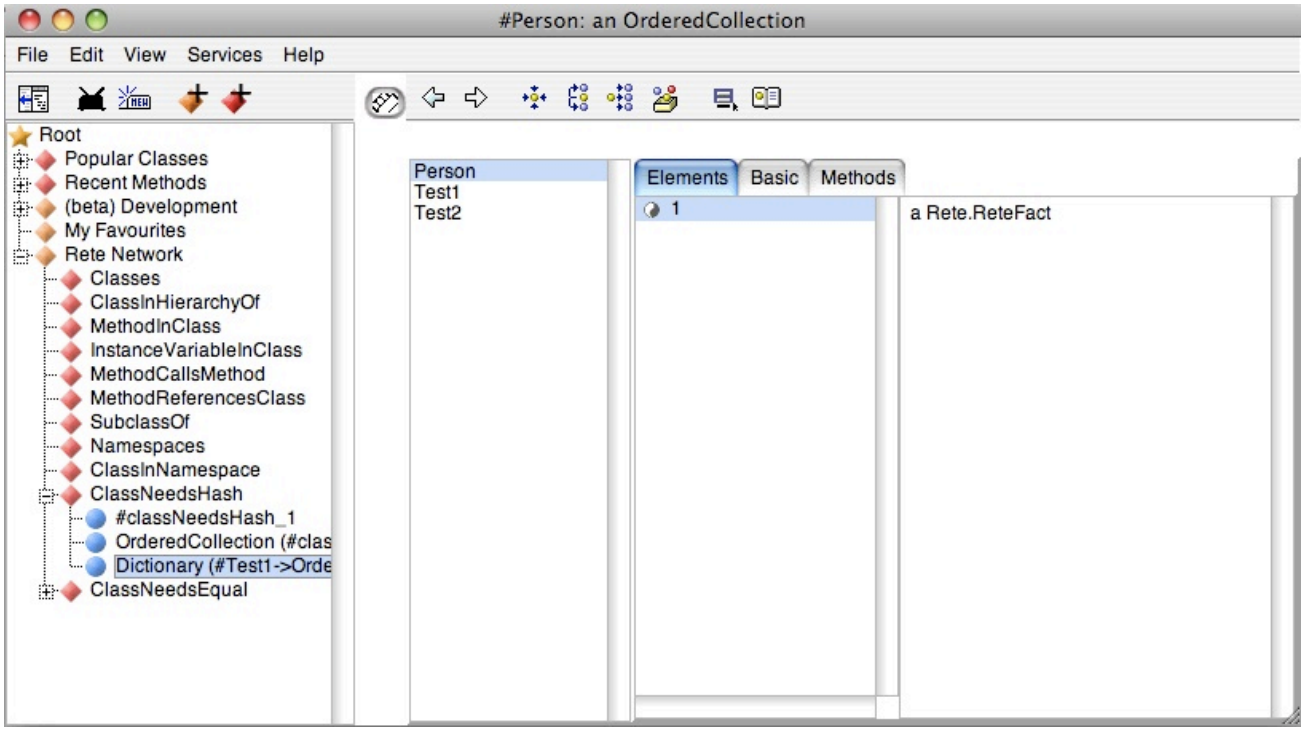
PARACHUT: outputs to developers (I)



StarBrowser

PARACHUT: outputs to developers (II)

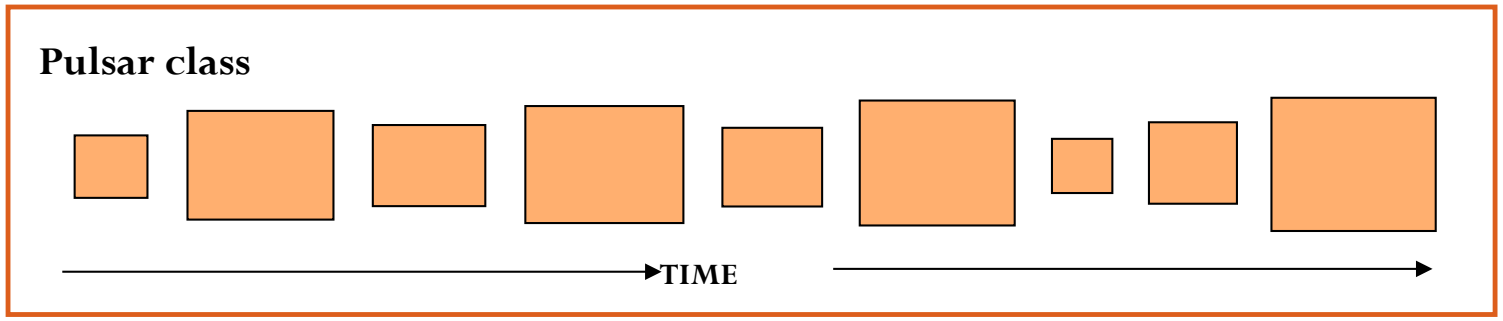




Other Examples

- Pulsar class pattern
- Possible refactoring opportunities

Pulsar Class Pattern



pulsar(*?class*) ***if***

pulsar(*?class*, *?len3*, *?len2*, *?len1*)

?len3 at t3
?len2 at t2
?len1 at t1
t3 > t2 > t1

pulsar(*?class*, *?len3*, *?len2*, *?len1*) ***if***

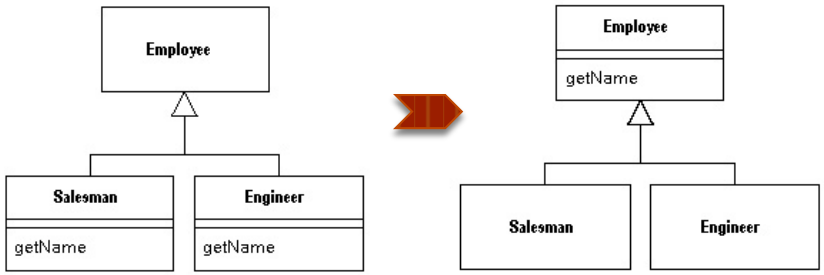
allConsecutiveSizes(*?class*, *?len3*, *?len2*, *?len1*),

escape(*?val*, [(*?len1* * 2) = *?len2*

and:[*?len1* = *?len3*]])

e.g. 10 – 20 – 10

Detecting possible refactoring opportunities



duplicateSelector(*?classA*, *?classB*, *?method*) *if*

addedMethodInClass(*?classB*, *?method*),

mostrecent(addedMethodInClass(*?classA*, *?method*))

refactoringOpportunity(*?superclass*, *?method*) *if*

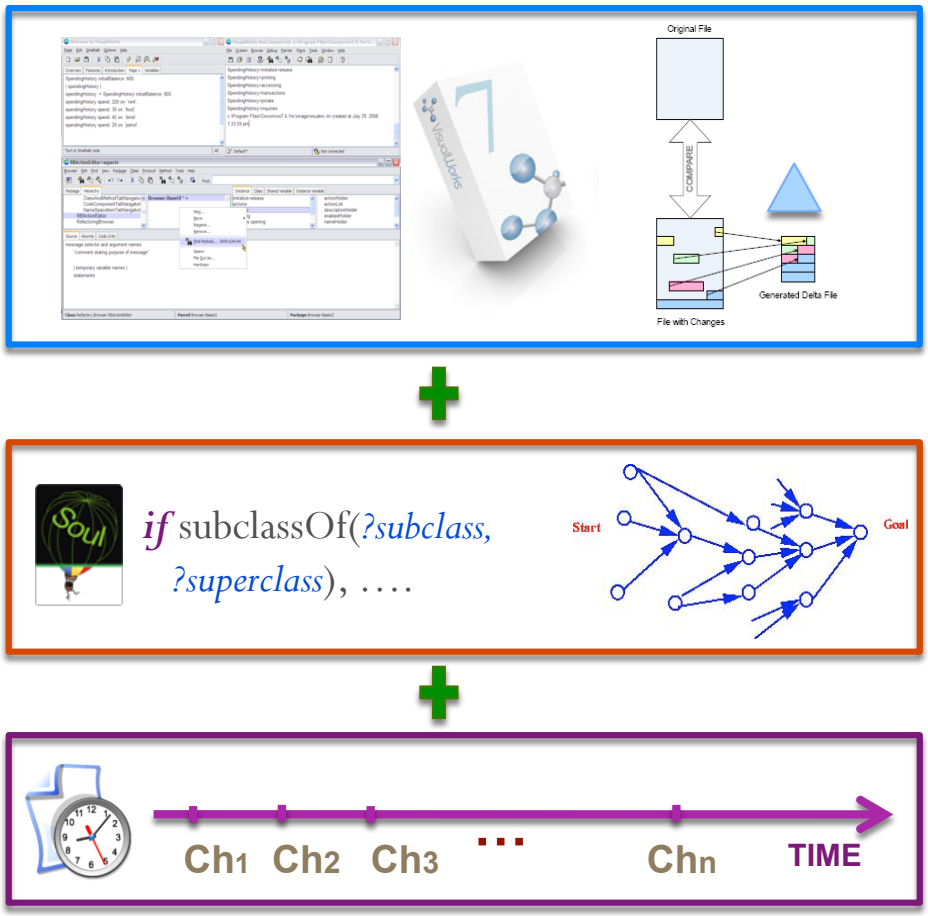
duplicatedSelector(*?classA*, *?classB*, *?method*),

subclassOf(*?classA*, *?superclass*),

subclassOf(*?classB*, *?superclass*),

not(addedMethodInClass(*?superclass*, *?method*))

Conclusions



Verónica Uquillas Gómez
vuquilla@vub.ac.be