Uniform and Safe Metaclass Composition

Stéphane Ducasse, Nathanael Schaerli and Roel Wuyts

ducasse@iam.unibe.ch
http://www.iam.unibe.ch/~ducasse/

Why Metaclasses?

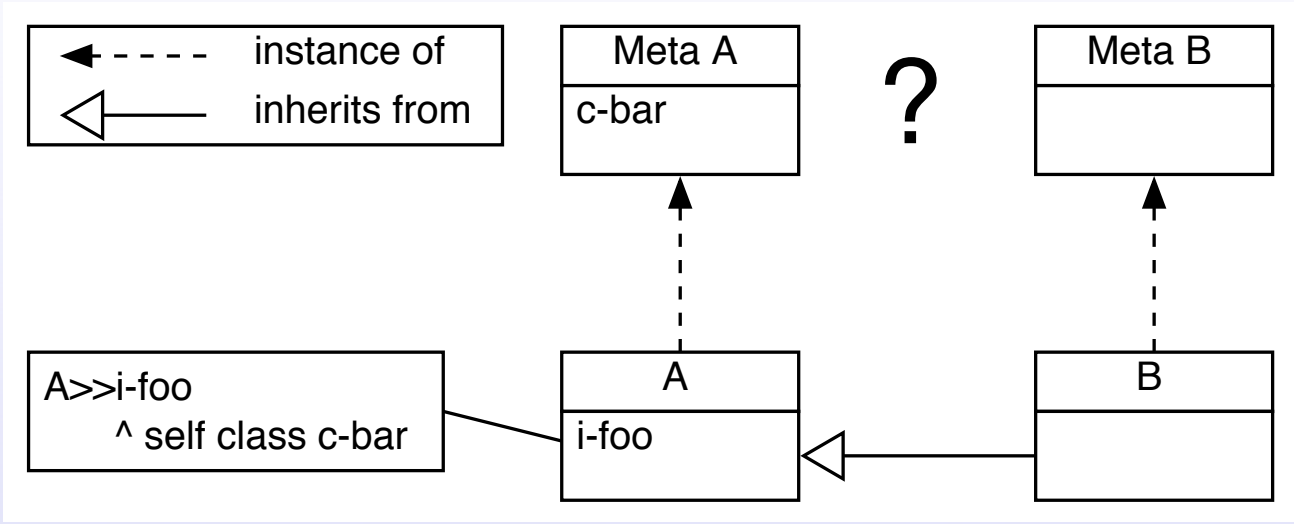
- Classes are objects too
- A class is an instance of a class called a metaclass

- Uniformity: Classes are just objects
- Control: Instance creation behavior
- Reuse of class behavior

Implicit vs. Explicit

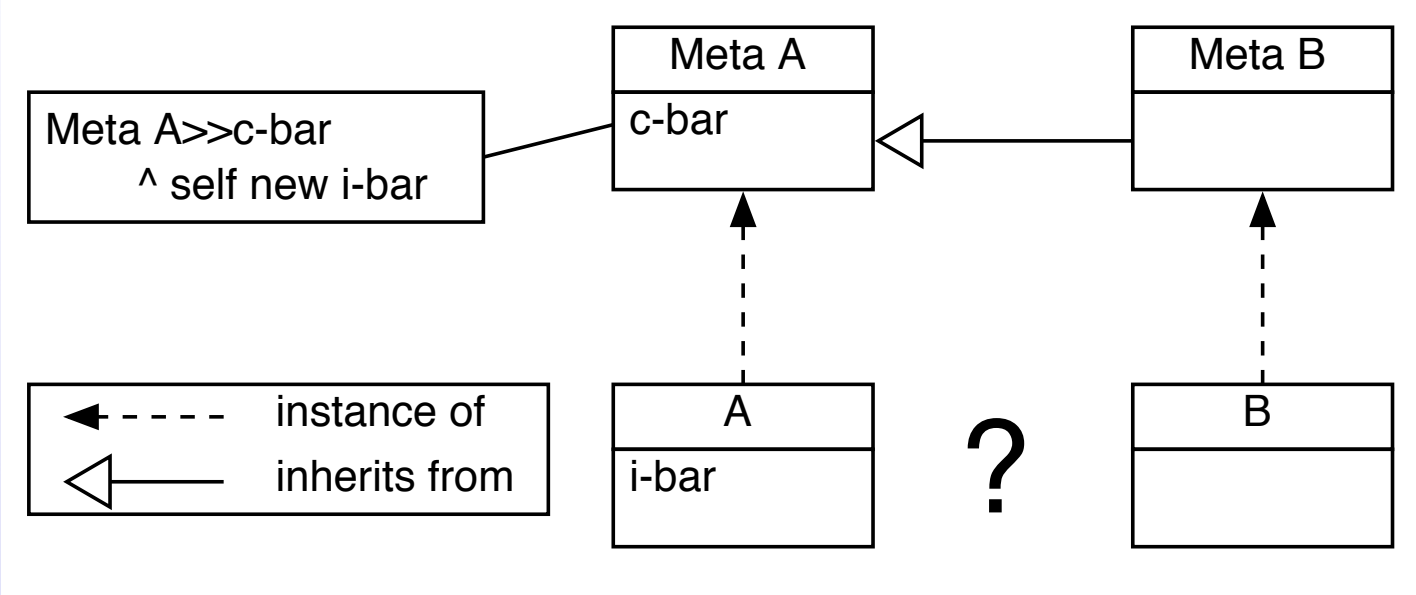
- Implicit: programmer cannot specify the class
  - Safe but
  - Limited or no reuse
- Explicit:
  - Composition metaclass is not safe
  - Ad-hoc composition mechanisms (for example: strings manipulation of neoclasstalk and dynamic class change)
- Safe?: upward and downward compatible

Upward Compatible



- aB i-foo => no error

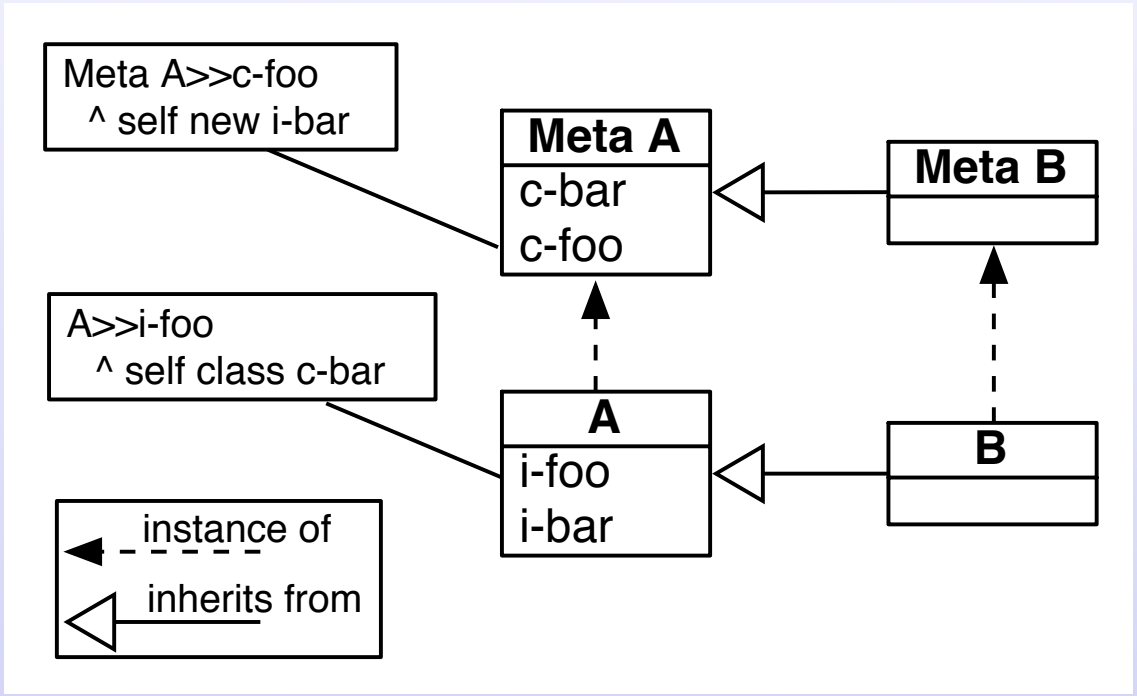
Downward Compatible



- B c-bar => no error

Smalltalk-80

- Implicit
- Safe
- Limited reuse

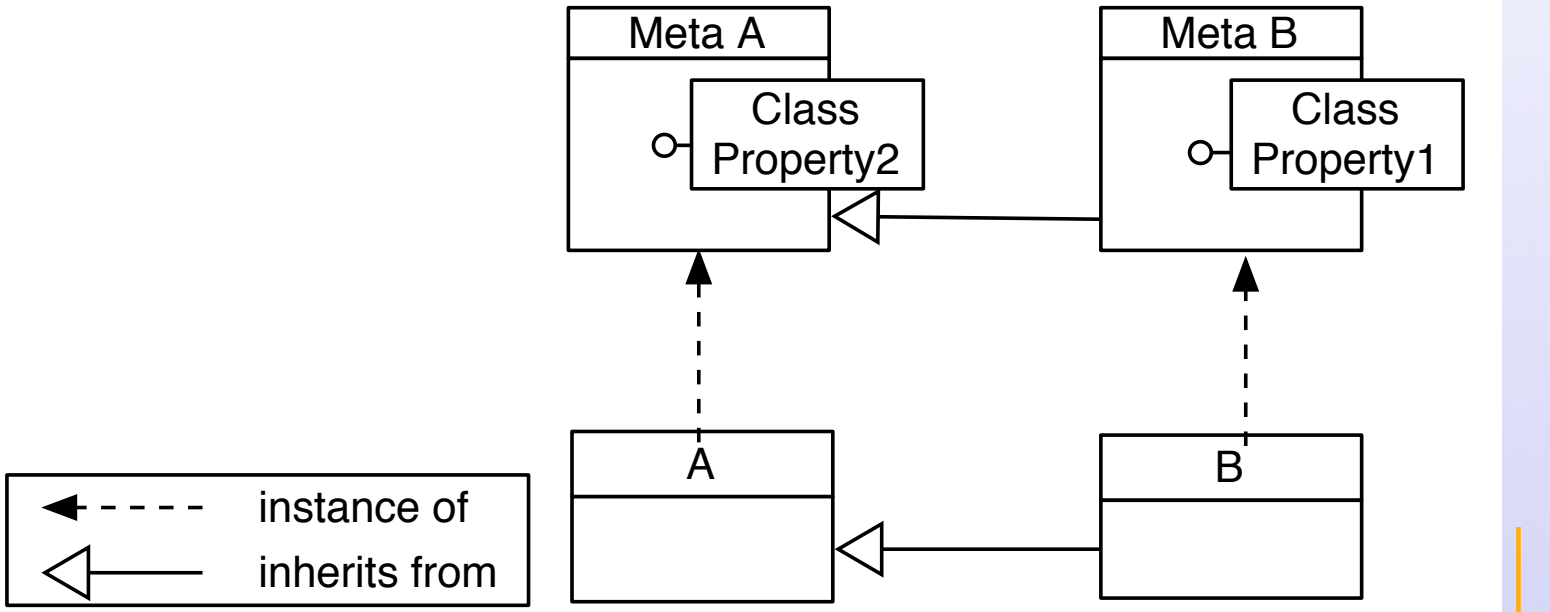


Existing Solutions

- Smalltalk-80
  - Implicit, safe
  - But limited reuse
- Clos
  - Explicit but help yourself!
- NeoClasstalk
  - Dynamic class changes
  - Metaclass behavior specified as strings
- MetaClasstalk
  - Double meta-layer with mixin composition

Our approach: traits

- Implicit metaclasses
- But traits to compose classes
- Reuse of traits



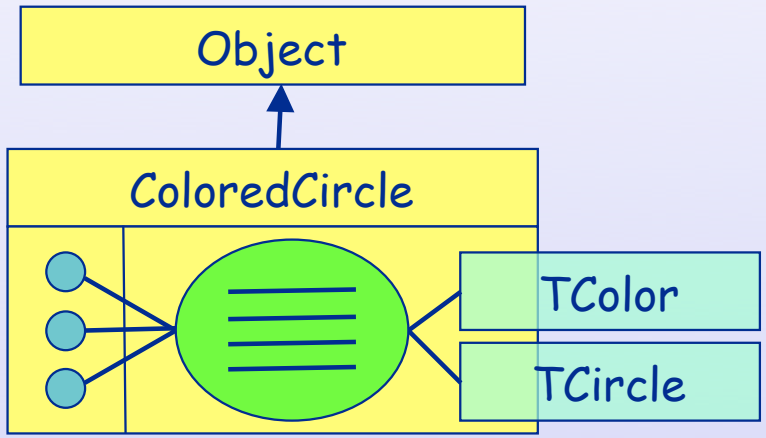
What are Traits?

- Traits are parameterized behaviors
  - Traits provide a set of methods
  - Traits require a set of methods
  - Traits are purely behavioral
  - Traits do not specify any state

TCircle	
area bounds diameter hash	radius radius: center center:

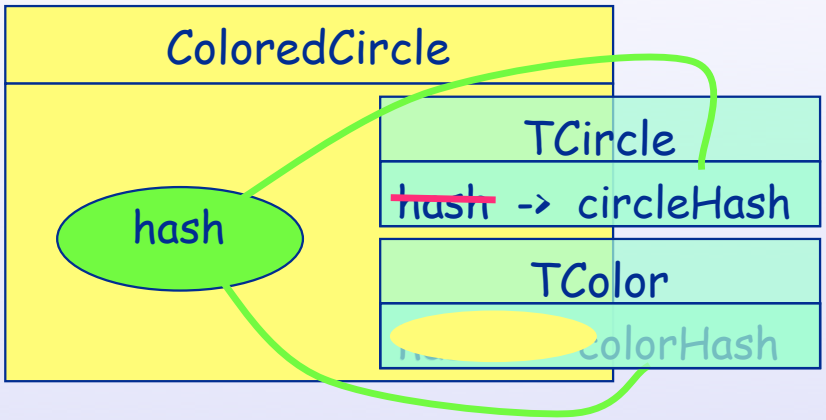
How are Traits Used?

- Traits are the behavioral building blocks of classes
- Class = Superclass + state + traits + glue methods



- Class methods take precedence over trait methods

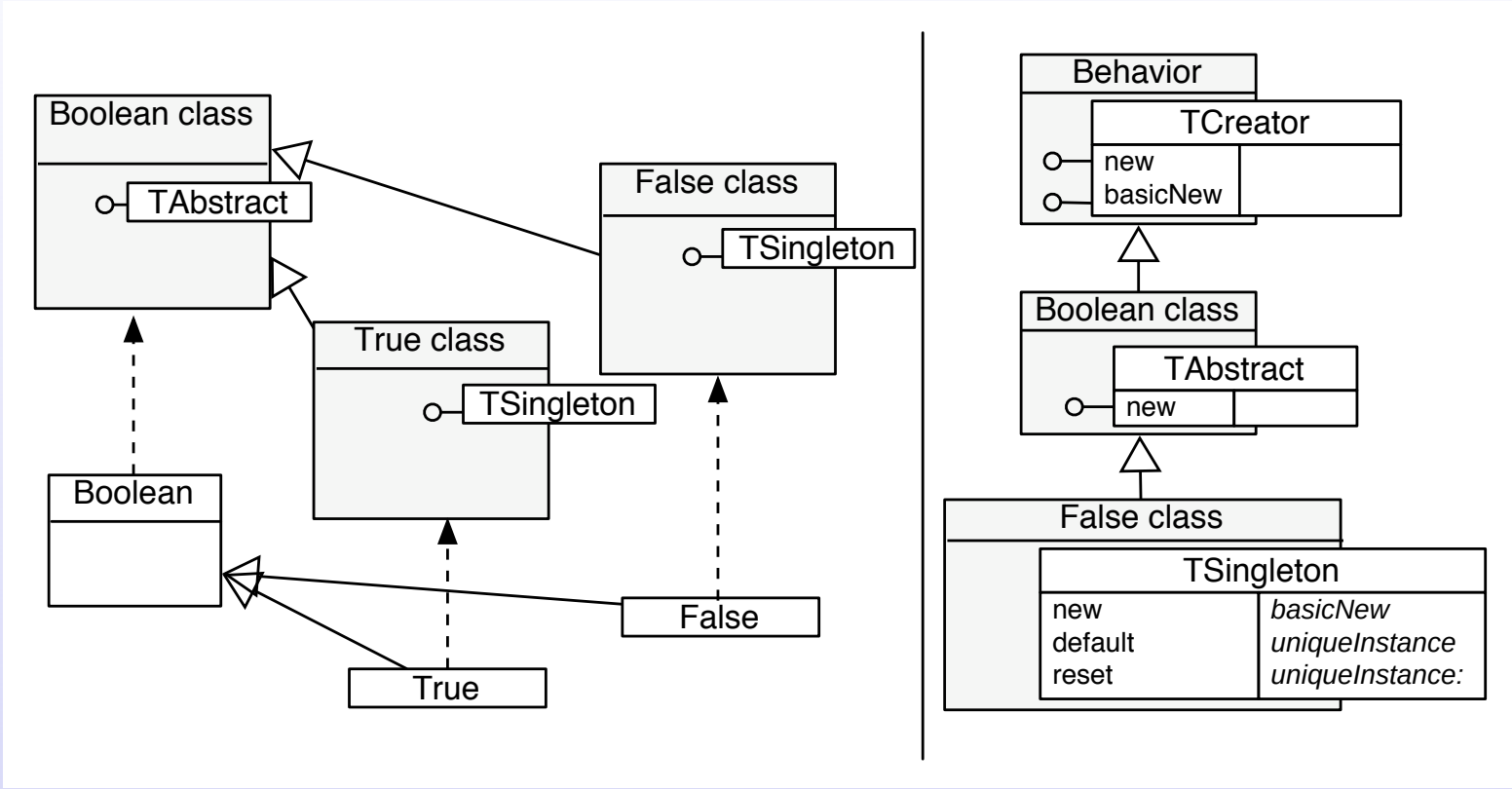
Conflicts *Explicitly* Resolved



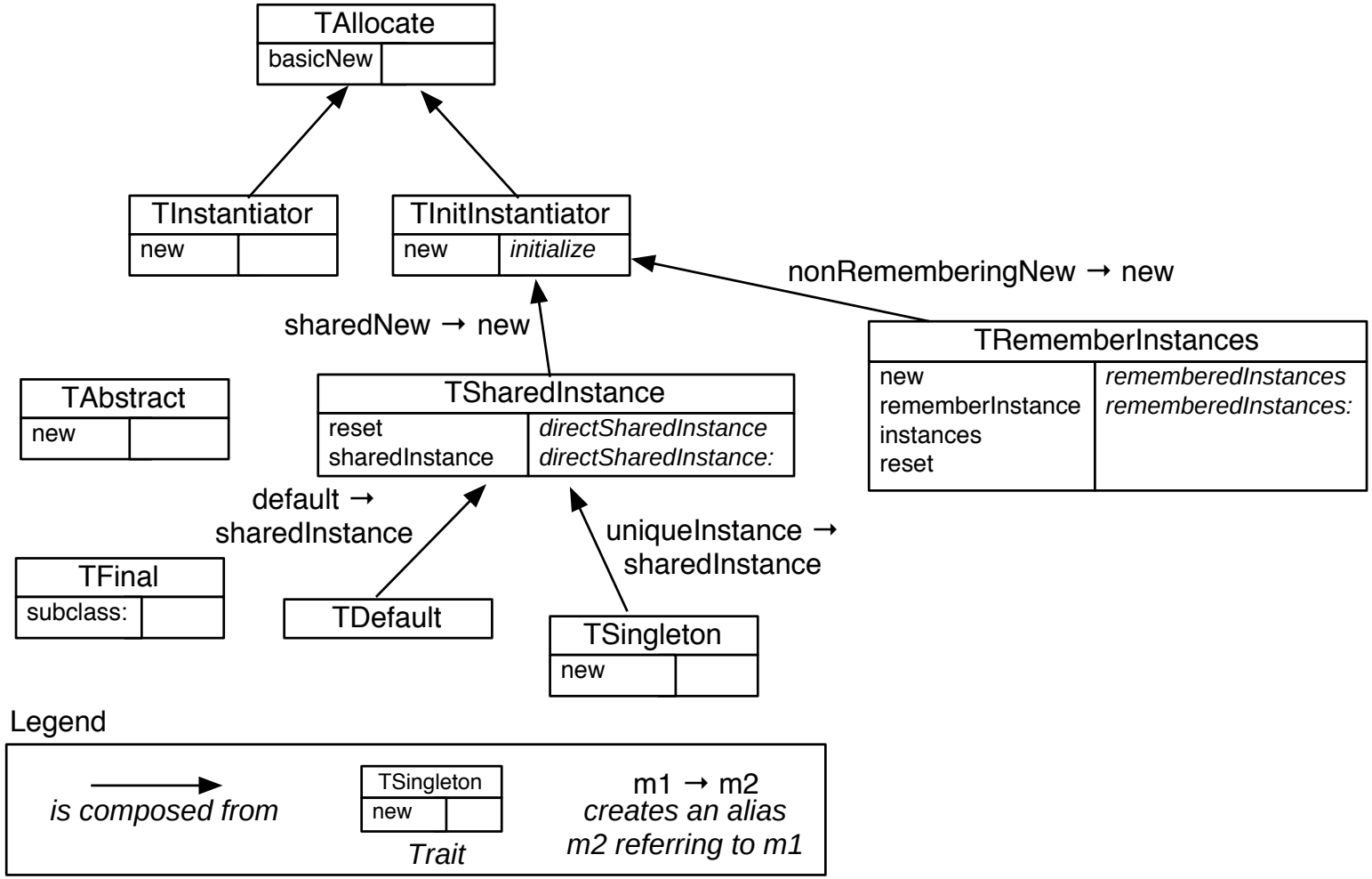
I) Override the conflict with a glue method
- Aliases provide access to the conflicting methods

II) Avoid the conflict
- Exclude the conflicting method from one trait

Hierarchies Revisited...

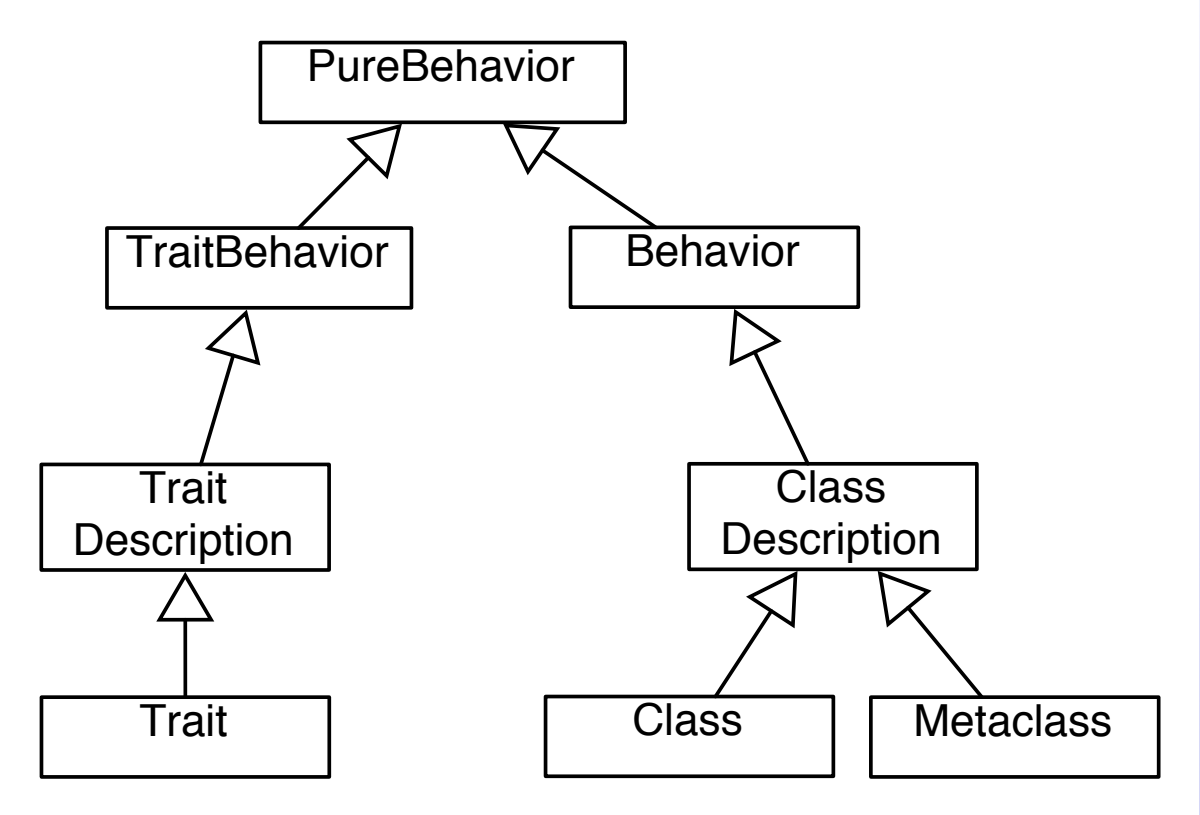


Metalevel Reengineering



Traits Implementation

- Available in Squeak 3.7
- Bootstrapping the kernel



Conclusion

- Uniform
- Traits are instance and class level!
- Explicit Composition
- Reuse but Safe