New Implementation of Traits in Squeak

The new clean and stable traits implementation - a short overview

Adrian Lienhard, Software Composition Group, Berne, Switzerland

Requirements

- Simple installation and usage
  - "One-click-installation" into current Squeak image
  - Should run on unmodified VM
  - Clean design and unit tests
  - Traits Browser, Monticello, FileIn/-Out, …

Design Considerations

Adding traits to a class:

a) Change method lookup
   - DNU-mechanism
   - Modify VM

b) Add trait methods directly to method dictionary
   - “Flattening property” (no modification of the lookup needed) but…
   - Manage method dictionary and keep track of which methods are locally defined
   - Update method dictionary whenever a relevant method in a trait has changed (=>update-mechanism)

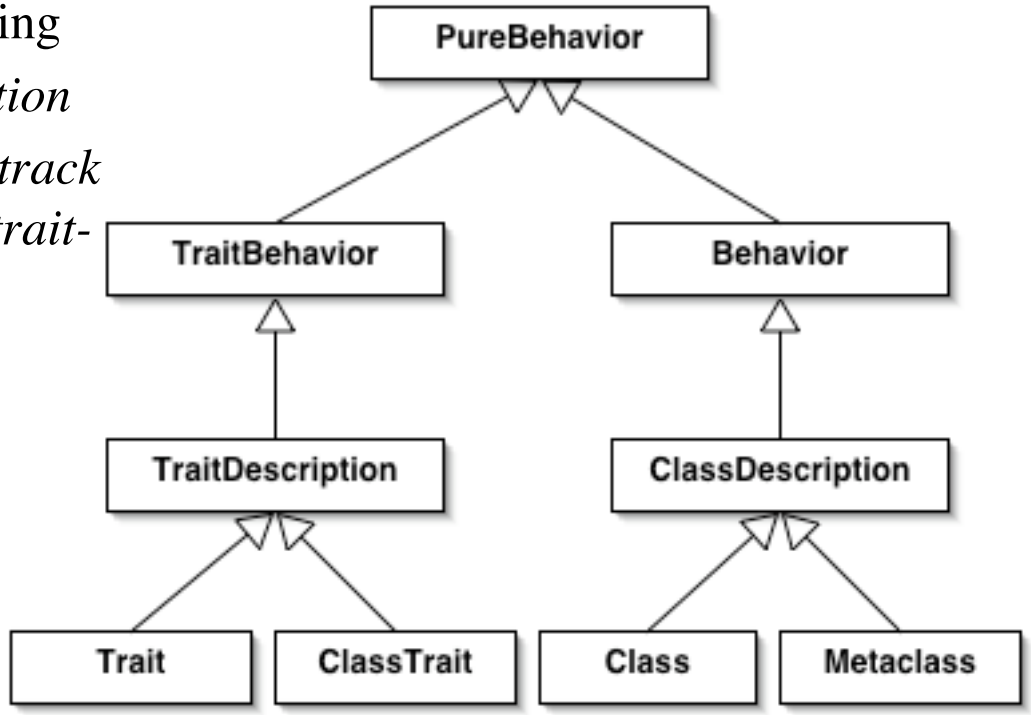
Architecture: Modelling Traits

Traits and Classes have in common:

- method-dictionary, compiling
- *setting a new trait composition*
- *update-mechanism to keep track of changes in traits of it's trait-composition*

- categorizing methods
- fileIn/Out

- name
- class-side (classtrait vs. metaclass)

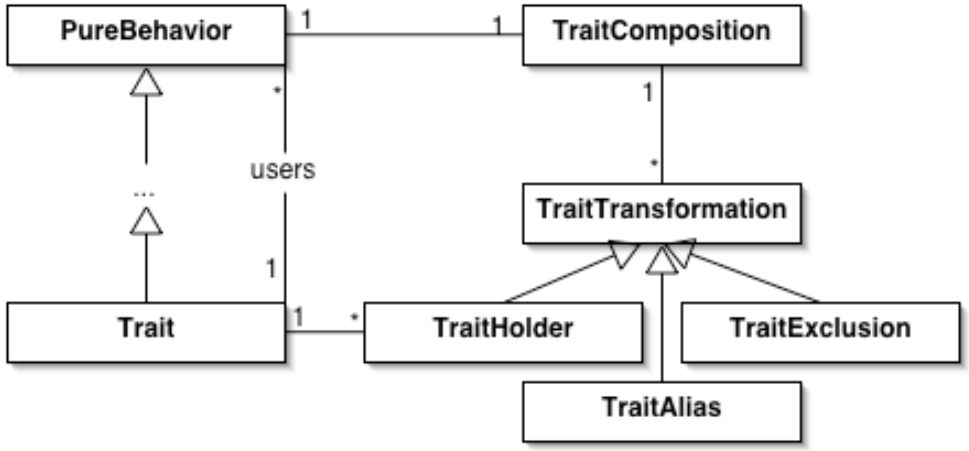


Problems

- Parallel hierarchy
  - Problem: Code duplication in *Class-/TraitDescription*
  - Alternative: *Trait* inherits from *ClassDescription*. But too much class specific behavior, e.g., #new
- Modification of kernel classes:
  - VM accesses instance-variables (superclass, methodDict, format) by index => i-vars can not be added to a superclass of *Behavior*

Architecture: Composing Traits

- Trait-composition is constructed by a Smalltalk expression (using message sends)
- *PureBehavior*s can set a trait- composition
- Traits keep track of users for notification of changes



Example:

```
Object subclass: #MyClass
        uses: Ta @ {y->x} - {#x} + Tb
        instanceVariableNames: ''
        ...
```

Refactoring the Traits Kernel

Tackle code duplication:

Refactor *Class-/TraitDescription* using traits itself

Solution:

- 9 common traits for: compiling, categorizing, testing, fileout, …

- Total methods in traits: 78
- Local methods in ClassDescription: 59
- Local methods in TraitDescription: 35

Lessons Learned and Future Work

- 80% of the functionality developed in 20% of the time
- Squeak: open and flexible. But still, a lot of detail work and refactoring was needed (KCP)
- Future enhancements/fixes to Squeak kernel will have to be handled manually
- Traits Browser, Monticello and FileIn/Out
- Future work
  - Further refactoring of the kernel
  - Bootstrapping