Object-Centric Instrumentation with Pharo

Steven Costiou

Square Bracket tutorials
February 14, 2019

Layout and typography based on the `sbabook` LaTeX class by Damien Pollet.

Contents

Illustrations

CHAPTER 1

Anonymous subclasses

Anonymous classes are nameless classes that are inserted between an object and its original class [FJ89, HJJ93]. The object is migrated to that new class, which takes the original object's class as its superclass. Methods from the original class can be redefined and reimplemented in the anonymous class, having the effect to change the behavior of that single object. Original behavior that is not redefined in the anonymous behavior is preserved.

1.1 Example

Talents are based on traits. Objects can answer to the `#addTalent:` messages, which takes a `Trait` as parameter. All behavior defined in the trait is flattened in the object. In the following illustration, we instantiate an anonymous trait, and we compile a method in this trait. That method is an instrumented version of the original `name` method of the class `Person`. This new method replaces the original one, until the talent is removed from the object.

```
|person anonClass|
  person := Person new.
  anonClass := anObject class newAnonymousSubclass.
  anonClass
    compile:
      'name: aName
        self tag: aName.
        name := aName'.
  anonClass adoptInstance: person. "migrates the object to its new
    class"
  anonClass superclass adoptInstance: "migrates back the object to
    its original class"
  ^anonClass
```

1.2 Evaluation

Manipulated entity: Trait. Behavioral variations are expressed using traits. It can be Traits defined in the image or anonymous trait instances in which specific behavior is manually compiled by the developer.

Reusability: Yes. A trait can be added as a Talent to any number of objects.

Flexibility: Partial. Using anonymous traits forces the user to manually compile code in the method. This is however necessary to achieve a sub-method granularity. Conflicts must be resolved manually when Traits are composed.

Granularity: Method. Traits add, remove or alter (through aliasing) the behavior of a method. It can be done at a sub-method level (*e.g.* inserting a statement in the body of a method), but that requires manual rewriting of the method in the Trait.

Integration: Partial. The object is migrated to an anonymous subclass, which does not break system tools. However, it may break libraries that uses classes and class names as a discriminator.

Bibliography

[FJ89] Brian Foote and Ralph E Johnson. Reflective facilities in smalltalk-80. In *ACM Sigplan Notices*, volume 24, pages 327–335. ACM, 1989.

[HJJ93] Bob Hinkle, Vicki Jones, and Ralph E Johnson. Debugging objects. In *The Smalltalk Report*. Citeseer, 1993.