Can Programming be Liberated from the Two-Level Style?

by Thomas Kühne and Daniel Schreiber
presented at OOPSLA'07

Software Engineering Seminar
Reto Mock, 24.03.2009
Motivation

- 3 (or even more) levels of ontological classification
- only 2 levels supported by OO model: classes and objects
- what do we do?
Super vs Meta

- '2001: A Space Odyssey' is an instance of a DVD
- but DVD is a specialisation of Product
  \[ \text{subset (}\subset\text{)} \]
- whereas a DVD is an instance of a ProductType
  \[ \text{elementhood (}\in\text{)} \]
Workarounds

- Item Description Pattern
Diagram showing relationships between different elements:

- **ProductType**
  - `taxRate : Integer`

- **Product**
  - `price : Float`

- **Book**
  - `taxRate = 7`

- **mobyDick**
  - `price = 9.95`

- **DVD**
  - `taxRate = 19`

- **2001**
  - `price = 19.95`
Problems

- inheritance has to be emulated
- built-in language instantiation vs. `isOfType` relationship
- loss of typesafety
- mismatch of models

» accidental complexity
Solution: DeepJava

• clabjects instead of classes and objects
• deep instantiation
• potency value defines on which level a certain field exists (default is 1)
  ➔ deep characterisation
public class ProductType extends ProductCategory {
    public ProductType(String categoryName, int categoryCount, int taxRate) {
        super(categoryName, categoryCount);
        taxRate(taxRate);
    }

    int taxRate;
    public void taxRate(int t) { taxRate = t; }
    public int taxRate() { return taxRate; }

    private float netPrice;
    public void price(float p) { netPrice = p; }
    public float price() {
        return netPrice * (1 + type.taxRate / 100f );
    }
}
Language Integration

```java
public void soldOne() {
    categorySoldCount++;
    superType().categorySoldCount++;
}

DVD aso = new DVD();
aso.price(19.95f / (1 + aso.type().taxRate() / 100f));
aso.name("2001: A Space Odyssey");
aso.promoProduct(haChi_779);

ProductCategory{"Software Items", 333}
    SoftwareItem extends Product^0; // create SoftwareItem
DigitalMedium{DVD_Player, "DVDs", 222, 19}
    DVD extends SoftwareItem { // create DVD
        "public String toString () { return name() + " (" + " \\
+ "type .categoryName() + \\
)" + \\
(promoProduct() == null ? \\
" -> \\
+promoProduct());}"}}
```
Compiler Prototype

- Polyglot used as parser generator rather than JavaCC
- based on Java 1.4 grammar
- two compile phases
  - DeepJava sources are translated in Java sources which implement DeepJava semantics
  - regular javac produces byte-code
- not final/ not available for download
My Opinion

- Interesting concept
- Does the world really need it?
- No special VM needed
- Dynamic type and feature creation has to be used with care
- Performance?
- Tool Support? e.g. Debugger
Questions?