Can Domain-Specific Languages Benefit from Linguistic Symbiosis?

Kris Gybels
kris.gybels@vub.ac.be

Vrije Universiteit Brussel
The Concept

Linguistic Symbiosis

= Transparent Inter-language Communication
**Business Rules**

*Order*

- **price**
  - \(^{\text{self totalPrice} \times (100 - \text{self discount})/100}\)
- **discount**
  - \((\text{customer isLoyal}) \text{ ifTrue: } [^{^5}]\)
  - \((\text{self nrItems} > 20) \text{ ifTrue: } [^{^10}]\)

*Customer*

- **isLoyal**
  - ....

---

Business knowledge, factor out into more suitable language
Old SOUL = Prolog + Smalltalk terms

Order
price

^ self totalPrice * (100 - self discount)/100
discount

results := SoulEvaluator evaluate:
    'discount(?o, ?d)' with: ?o -> self.
^ (results bindingsFor: #d) first

Customer
isLoyal
....

discount(?order, 5) if
equals(?customer, [ ?order customer ], [ ?customer isLoyal ])
discount(?order, 10) if
equals(?nrItems, [ ?order nrItems ], larger(?nrItems, 20))

loyal(?customer) if
...

= use of “escape mechanisms” = not transparent
=> Code cluttering, overhead ...
**Order**

price

^ self totalPrice * (100 - self discount)/100

**Customer**

isLoyal

....

?order discount = 5 if
?order customer = ?customer &
?customer isLoyal

?order discount = 10 if
?order nrItems = ?nrItems &
?nrItems > 20

?customer isLoyal if
...

New SOUL = Prolog with Smalltalk-like syntax

Language interaction is absorbed into method/rule lookup

?customer isLoyal

Translation to message

isLoyal

?order discount = ?result

Translation to rule invocation
Language interaction in large-scale DSL-based software?

Approach?

SOUL = useful for *paradigmatic* fit to business rules, syntax change no problem

but many DSLs = *syntactic* fit to domain?