Categorical design patterns

FOP Away Day, 17 Jan 2007

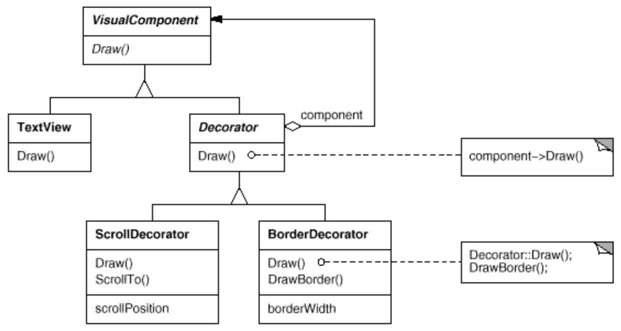
Ondřej Rypáček

University of Nottingham, UK

Design Patterns

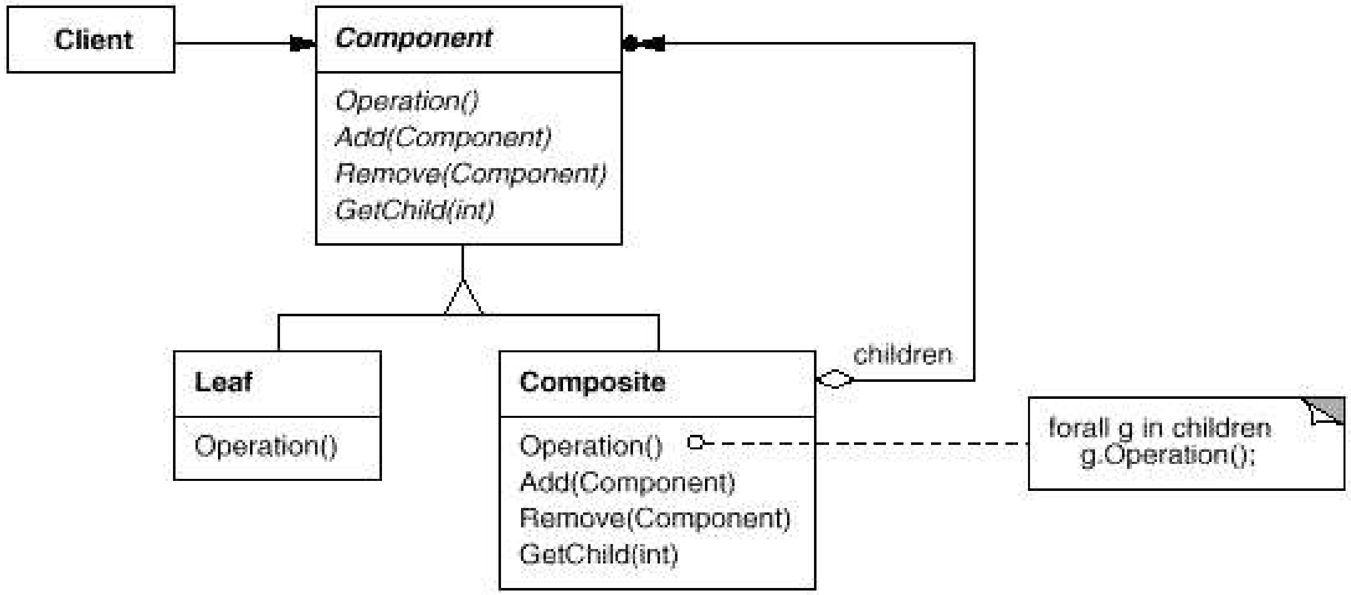
Design Patterns : Elements of Reusable Object-Oriented Software (Gamma, Helm, Johnson, Vlissides)

- describe the “good” designs in OOP
- informal, ambiguous



Decorator Pattern:
. . . The decorator conforms to the interface component it decorates so that its presence transparent to the component’s clients. The decorator forwards requests to the component may perform additional actions before or after forwarding. . .

Composite pattern



Formal Design Patterns

- formal objects in the language
- support reasoning about programs
- replace inheritance and lots of hand-coding by formally defined refinement steps

$$\begin{tikzcd}
O \arrow[r, "m"] \arrow[d, "c"'] & F.O \arrow[r, "zip"] & F.O \arrow[d, "F.c"] \\
O \arrow[rr, "m"] & & F.O
\end{tikzcd}$$

Category of “simple objects”

Objects: parametric object types (signatures)

Arrows: freely generated from *constructors*, *method calls*, *pairs*, *composition*

Objects interpreted directly, not via functional models and **Set** [Reichel, Jacobs, Pierce, Hoffman]

Thanks to ***Command*** and ***Visitor*** patterns, the category has **exponents** and **co-products**

(weakly) **terminal co-algebras** correspond to abstract object types and abstract methods

$$O \xrightarrow{m} F.O$$

Decorator vs *Composite*

“*Decorator* is a singleton *Composite*”

Composite pattern – formally

$$\begin{tikzcd} O \arrow[r, "m"] & F.O \end{tikzcd}$$

Composite pattern – formally

$$\begin{tikzcd} C.O \arrow[r, "C.m"] & C.F.O \arrow[r, "zip"] & F.C.O \end{tikzcd}$$

$$\begin{tikzcd} O \arrow[r, "m"] & F.O \end{tikzcd}$$

Composite pattern – formally

$$\begin{tikzcd}
C.O \arrow[r, "C.m"] \arrow[d, "c"'] & C.F.O \arrow[r, "zip"] & F.C.O \arrow[d, "F.c"] \\
O \arrow[rr, "m"] & & F.O
\end{tikzcd}$$

Decorator pattern – formally

$C \equiv I$

$$\begin{tikzcd}
I.O \arrow[r, "I.m"] \arrow[d, "c"'] & I.F.O \arrow[r, "zip"] & F.I.O \arrow[d, "F.c"] \\
O \arrow[rr, "m"] & & F.O
\end{tikzcd}$$

Conclusion and further steps

- The approach is very promising
- We already have some new results
  - natural interpretation of terminal co-algebras as abstract object-types
  - natural *zips* correspond to rearrangement of inputs and outputs in an object
  - discovered a relation between *composite*, *decorator* and *adapter*
  - formalised the relation of *Composite* and initial algebras – recursive structure traversals
- Future: lot of work and more results