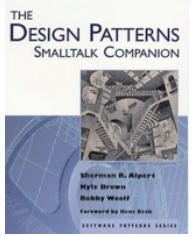
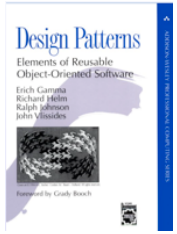
INRIA

Composite

Stéphane Ducasse
stephane.ducasse@inria.fr
http://stephane.ducasse.free.fr/

Source





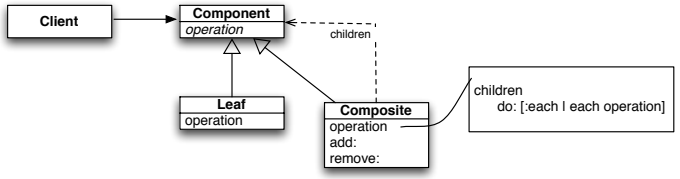
12 of 12 people found the following review helpful:

★★★★★ **Easier to understand than the original GoF**, February 4, 2000
Reviewer: **Nicolas Weidmann** (Zurich, Switzerland) - See all my reviews
This book gives you a better understanding of the patterns than in its original version (the GoF one). I am not a SmallTalk programmer but a 9 years C++ one. At work I had to use the GoF book and never liked reading it. In contrast to this, the SmallTalk companion is easy to read and you can understand the patterns within the first few lines of their description. Take the Bridge pattern and compare their discussions in the two books. If you really like the Gof one then buy it. But according to me, it would be a big mistake buying the GoF in favour of the SmallTalk companion. Trust a C++ programmer :-)

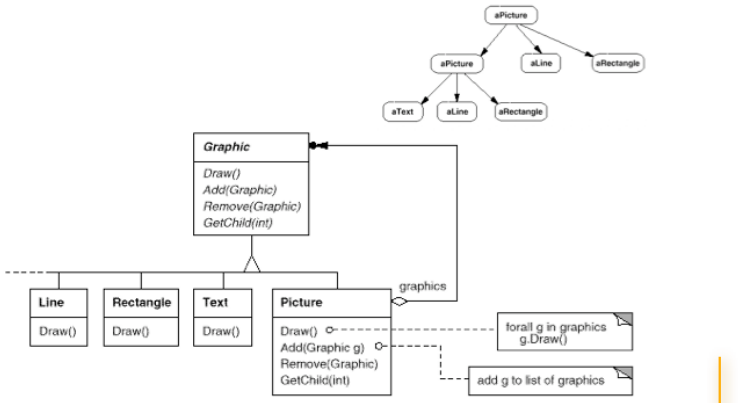
Was this review helpful to you? Yes No (Report this)

Composite Intent

- Compose objects into tree structures to represent part-whole hierarchies.
- Composite lets ***clients*** treat individual objects and compositions of objects ***uniformly***



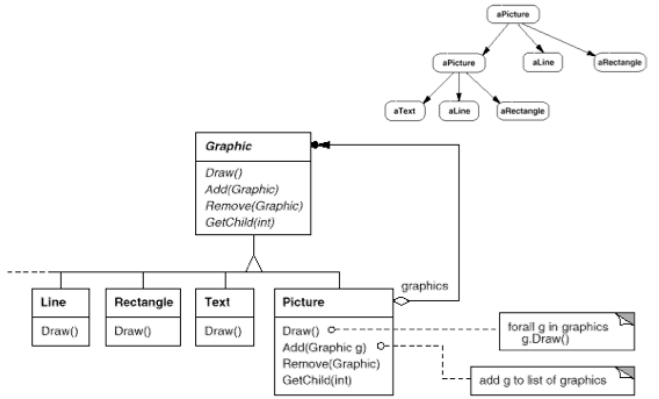
Composite Pattern Motivation



Composite Pattern Applicability

- Use the Composite Pattern when :
  - you want to represent part-whole hierarchies of objects
  - you want clients to be able to ignore the difference between compositions of objects and individual objects. Clients will treat all objects in the composite structure uniformly

Composite Pattern Possible Design



Composite Pattern Participants

- Component (Graphic)
  - declares the interface for objects in the composition
  - implements default behavior for the interface common to all classes, as appropriate
  - declares an interface for accessing and managing its child components
- Leaf (Rectangle, Line, Text, ...)
  - represents leaf objects in the composition. A leaf has no children
  - defines behavior for primitive objects in the composition

Composite Pattern

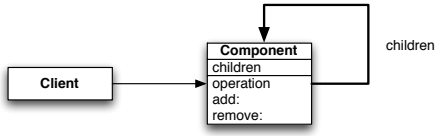
- Composite (Picture)
  - defines behaviour for components having children
  - stores child components
  - implements child-related operations in the Component interface
- Client
  - manipulates objects in the composition through the Component interface

Composite Pattern Collaborations

- Clients use the Component class interface to interact with objects in the composite structure.
- Leaves handle requests directly.
- Composites forward requests to its child components
- Consequences
  - defines class hierarchies consisting of primitive and composite objects.
  - Makes the client simple. Composite and primitive objects are treated uniformly. (no cases)
  - Eases the creation of new kinds of components
  - Can make your design overly general

An Alternate Structure

- Again structure is not intent!



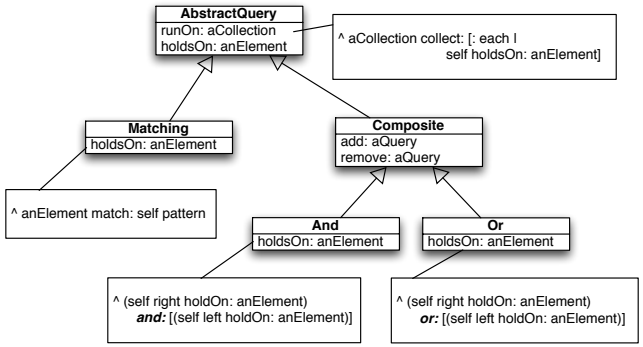
Queries...

- To be able to specify different queries over a repository

```
q1 := PropertyQuery property: #HNL with: #< value: 4.
q2 := PropertyQuery property: #NOM with: #> value: 10.
q3 := MatchName match: '*figure*'
```

- Compose these queries and treat composite queries as one query
- (e1 e2 e3 e4 ... en)((q1 and q2 and q4) or q3) -> (e2 e5)
- composer := AndComposeQuery with: (Array with: q1 with: q2 with: q3)

A Possible Solution



In Smalltalk

- Composite not only groups leaves but can also contain composites
- In Smalltalk add:, remove: do not need to be declared into Component but only on Composite. This way we avoid to have to define dummy behavior for Leaf

Composite Variations

- Use a Component superclass to define the interface and factor code there.
- Consider implementing abstract Composite and Leaf (in case of complex hierarchy)
- Only Composite delegates to children
- Composites can be nested
- Composite sets the parent back-pointer (add:/remove:)

Composite Variations

- Can Composite contain any type of child? (domain issues)
- Is the Composite's number of children limited?
- Forward
  - Simple forward. Send the message to all the children and merge the results without performing any other behavior
  - Selective forward. Conditionally forward to some children
  - Extended forward. Extra behavior
  - Override. Instead of delegating

Other Patterns

- Composite and Visitors
  - Visitors walks on structured objects
- Composite and Factories
  - Factories can create composite elements