1.1 Composite Pattern

The Composite Pattern allows the objects in a tree or part-whole structure. Clients can differentiate between compositions of objects and individual objects . Therefore, the client can treat the node uniformly.

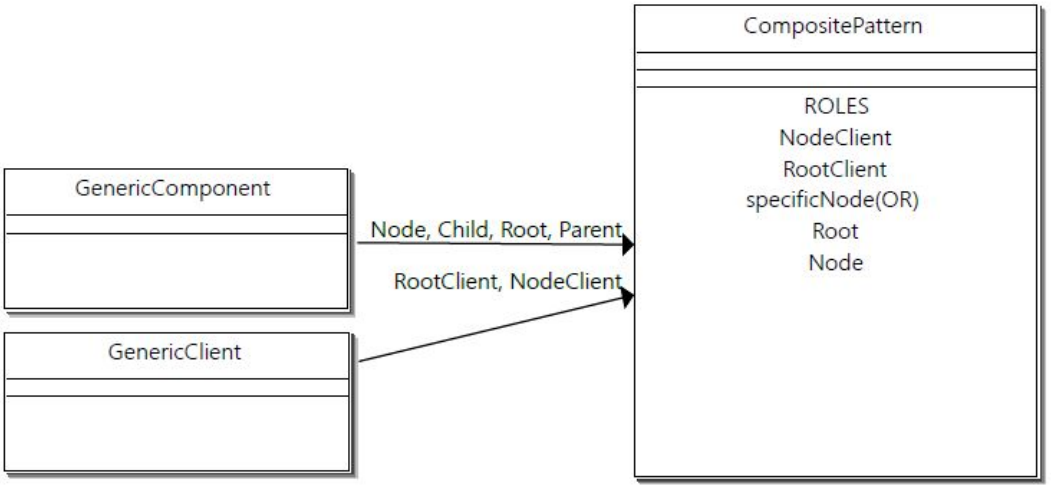

Figure 1.1 Composite Pattern Class Model

Documentation

The Figure 1.1 shows the class model of the composite pattern in the CROM representation. The Class GenericComponent is representative of classes that have a node in the tree represent. Thus, the classes Component, Composite and Leaf from the description of Gang of Four have summarized. Their different characteristics are in the presentation with the CROM through the roles that the GenericComponent class can play. There can be a class with the same characteristics as the class composite of the Gang of Four one Play root, parent and child nodes. A leaf class, can only do the role Child take over. Playing all these roles also shows that the role fulfills Node becomes.

Objects of class GenericClient can be client roles NodeClient and RootClient play. So you are accessing nodes and in the special case that they take over the second-named role, towards the root node. The CompositePattern class enables the design pattern to instantiate and includes the role model described below.

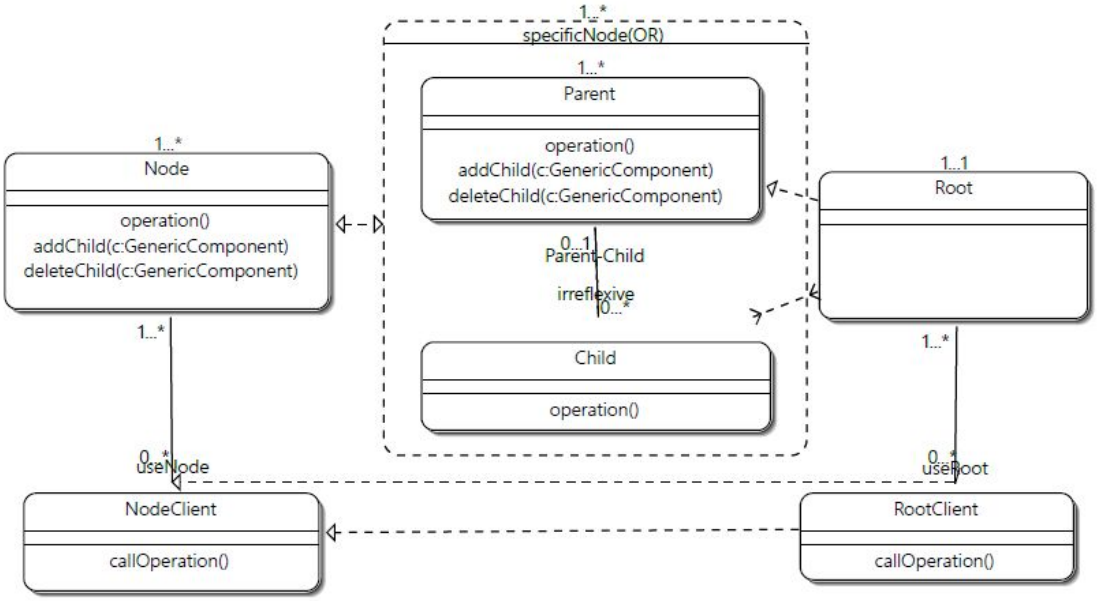

Figure 1.2 RoleModel

The Figure 1.2. The roles are primarily to those of Dirk Riehle classification. In the role model the methods and attributes are extended. The attributes are drawn over relations. The tree structure of the composite Pattern consists of 4 roles: **Node, Parent, Child and Root**. The role Node represents as general role of nodes in the standard tree implementation. Also for the methods of the child nodes, here referred to as operation, Here a standard implementation is defined. These child methods are the algorithm part of the design pattern. Methods to add and delete children become counted as administrative share. The parent role is played by a class that represents a grouping node. That's why If you use the role model the relation Parent-Child.

Evaluation

The **Gang of Four** documented the following features.

1. The design pattern creates a tree structure with leaf and grouping nodes.

2. Clients treat all nodes in the tree structure the same. Calls will not adapted to special nodes. This feature is in the parts intention and applicability described.

3. The classes in this pattern are Client, Leaf, Component and Composite.

4. The Component class is abstract.

5. The Component class provides the interface for methods used by all leaf and grouping nodes are shared.

6. The Component class provides a standard implementation for methods used by shared across all leaf and group nodes.

7. Grouping nodes call the methods of the same name in methods of the algorithm component Methods in their child nodes.

8. The Leaf and Composite classes inherit from the Component class.

9. The Composite class has an aggregation to the Component class to its Save child node.

10. The Leaf class has no child nodes.

11. The Leaf class implements the behavior of the methods of the algorithm part by overwriting the standard implementation from Component.

12. The Composite class stores its child nodes.

13. The Composite class implements the behavior of child node management methods by overwriting the standard implementation from Component.

14. The Composite class implements the behavior of the methods of the algorithmic part by overwriting the standard implementation from Component.

15. Clients interact with the common nodes, not with the specific node types.

16. Composite interacts with child composite nodes.

17. Composite interacts with child nodes of type Leaf.

18. The design pattern allows a simple client, as this on the interface can access from Component.

19. The pattern simplifies the addition of node types, as only these are must meet the interface of Component.

20. A check whether a particular child node is to a grouping node must be added at runtime. This is no more controllable by type queries.

21. If the parent node is to be stored explicitly, the invariant must apply, that the parent node also recorded the referencing node as a child.

22. The number of methods in the Component class should be maximized, given so the train? on nodes in the tree as possible as possible.

23. Saving child nodes can be done in any structure. This depends from the number of child nodes.

24. The order in which child nodes are stored can be important.

25. When caching data in the parent node involving child nodes, the cache Invalidate entry for changes in the child node.

26. If a grouping node is to be removed, it will delete all recursively beforehand his child nodes.

The following features are provided by **Dirk Riehle**.

27. There are the following roles in the design pattern: Parent, Child, Node, Root, NodeClient
and RootClient. This feature is additionally rendered graphically.

28. The root role represents a access point for the tree.

29. The role Node decrypts all client-invocable methods on nodes of the Tree.

30. The role Node corresponds in part to the Component of the Gang of Four.

31. The role Parent manages his child nodes.

32. The parent and child roles are partially equivalent to the component and component classes Composite of the Gang of Four.

33. The parent and child roles imply the role Node. This feature will too graphically documented.

34. The root role implies the parent role and prohibits the child role. this will additionally shown graphically.

35. The composition constraints of the Role Relationship Matrix [Rie09, p. 36] apply.

The Feature Table Explains all the properties as shown below.