Observer Design Pattern [Gamma et al]

Observer pattern allows us to implement publisher/subscriber scenario where any change to publisher automatically notify its subscribers. There are many different ways to implement observer pattern, such as using delegates and events or the Template Classes, the concepts are all the same. That is, the observers are registered to listen to the changes in the subject and are notified when the subject changes. It is also termed as “Distributor/Listener pattern”. Following are the variants of Observer design pattern:

Multiple Instance Observer [65]

A formal and unified architecture of software patterns has been introduced Real Time Process Algebra notations. Case studies on the Abstract Factory and Observer patterns have been chosen to illustrate the expressive power of Real Time Process Algebra in specifying design patterns. The formal approach has shown that architectures, static and dynamic behaviors of software patterns can be rigorously described by RTPA. Relationships between the formal and informal models of design patterns have been comparatively analyzed.

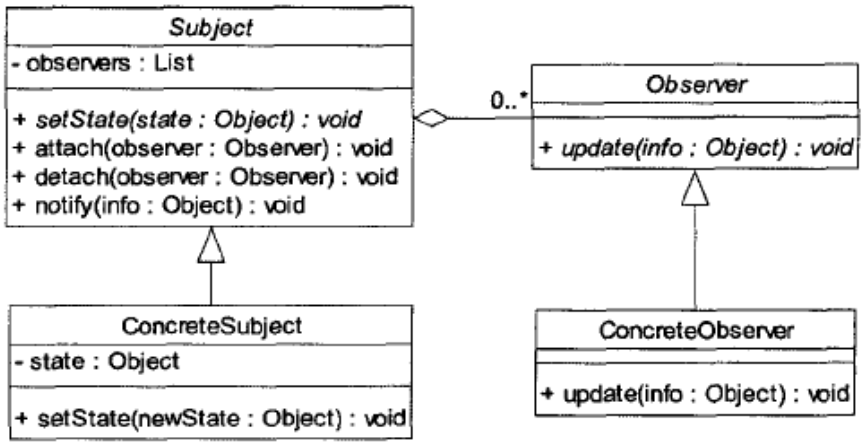

Figure 1.21 Variant of Observer

Compound Implementation [66]

In this variant observer is embedded into responsibility chain to form a composite pattern. It is not only to enhance the ability that responsibility chain pattern deals with user's request, but also because of each handler's observer being non-coupling with other handlers enables the responsibility chain to have a better flexibility, and each handler can dynamically add, delete their own observer. Following is the UML Diagram:

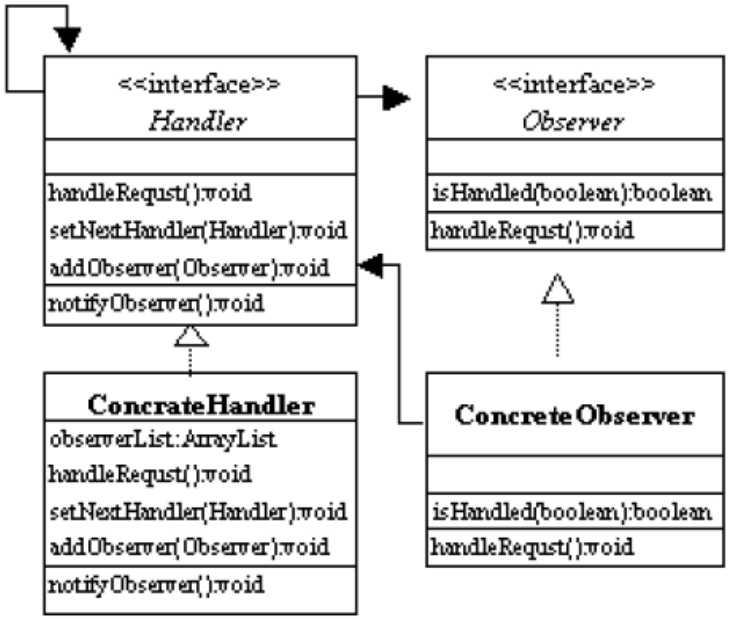

Figure 1.22 UML Diagram of composite implementation