Command Pattern [Gamma et al]

Command pattern is intended to provide a framework within which data is modified via controlled mechanism. This pattern is used to manage algorithms, relationships and responsibilities between objects. This pattern is often a matter of discussion in menu systems and applications where client request an operation and the object that perform it. Being a versatile in nature Command Pattern can be used in multiple ways, following are the variants of Command Pattern:

Façade Command [56]

The Facade design pattern was implemented as one class, which remotely exposed methods corresponding to use cases of the application. The Command design pattern was implemented as a set of classes that represented use cases and corresponded directly to the methods of the Facade pattern. The command classes were executed remotely in a command executor. The UML Diagram is given below:

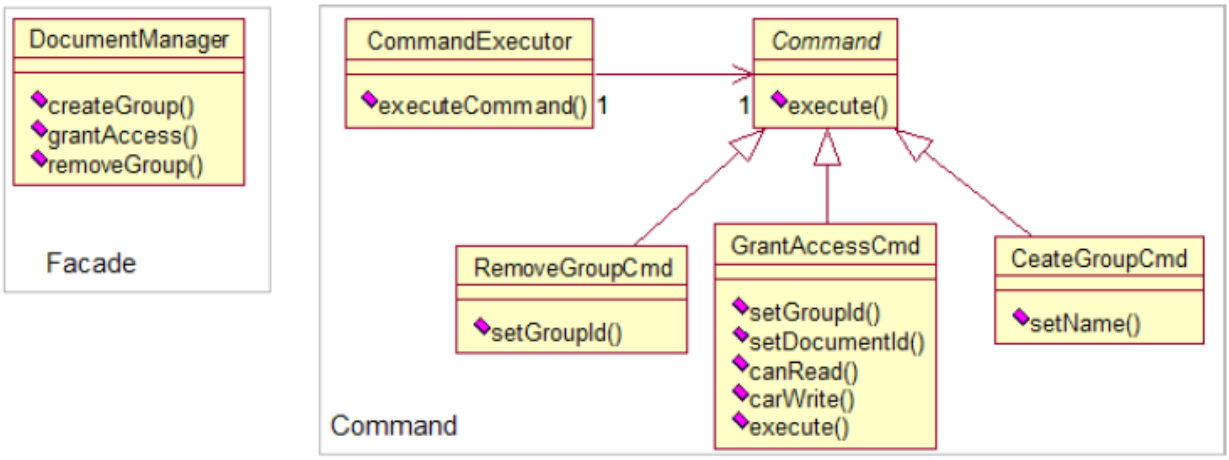

Figure 1.2 UML Diagram of Façade Command

Combined Command [56]

Command Pattern is modified to use multiple commands for one remote call. Following is the UML Diagram:

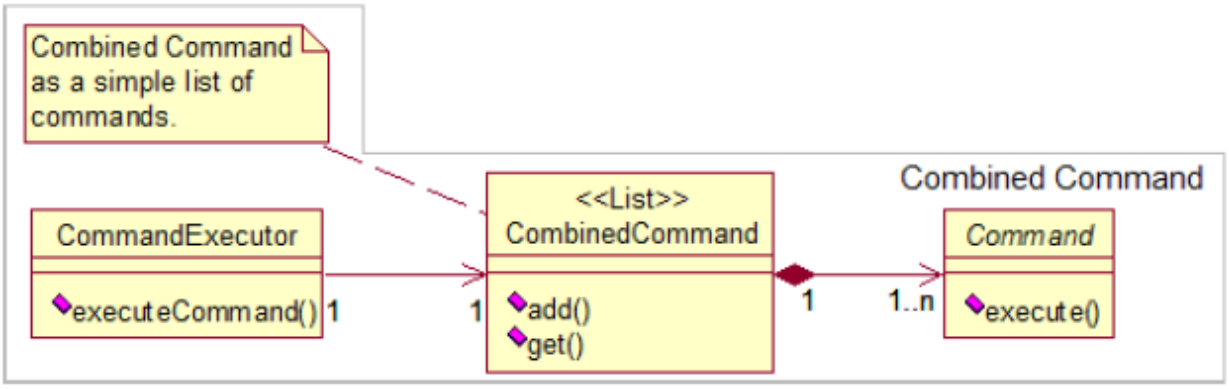

Figure 1.2 UML Diagram of Combined Command

Distributed Command Pattern [57]

Distributed command pattern solves the problems by introducing a mechanism where the same command is invoked automatically on connected systems when any one system executes a command. This pattern offers a way to invoke a command in-process on a connected system from another connected system. There is no need to send a custom message containing required data to the server to notify about the command that is executed, and there is also no need for clients to read and parse a message in order to extract necessary data and then act according to the message. The every command that is executed on a system is serialized and transmitted to other systems and then executed locally in order to simulate the same situation that occurred on the originating system. Following is the UML Diagram:

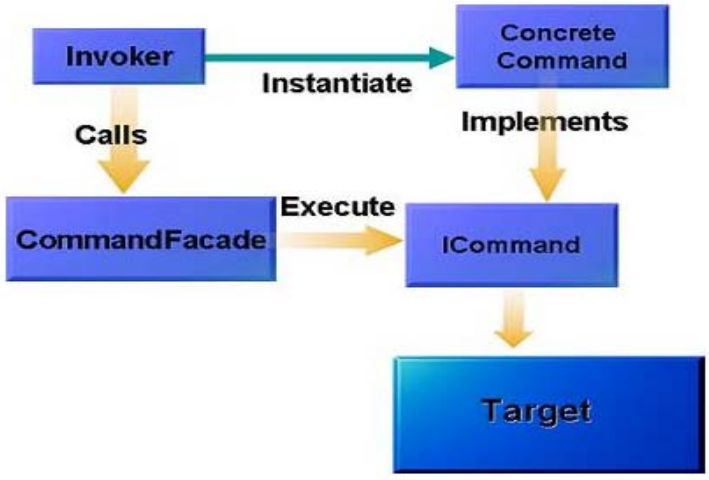

Figure 1.3 UML Diagram of Distributed Command

Command Revisited [58]

This variant integrates a piece of functionality as well as its parameterization in an object in order to reuse in other context. The basic intent of this variant was to address context independent execution, parameterization and decoupling. The UML Diagram is given below:

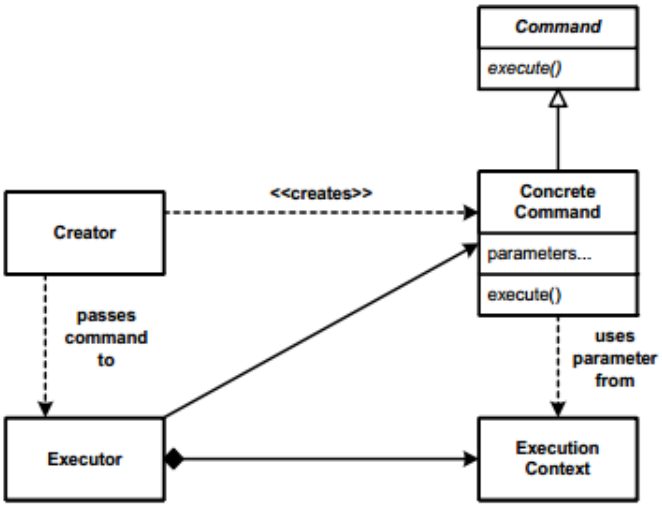

Figure 1.4 UML Diagram of Command Revisited Pattern