Austin Olbrych

CSC 491

In software engineering, there are many types of design patterns that can be used to structure projects. The two patterns that will be compared are chain of responsibility and composite. First, background information will be given to provide basic knowledge of the two designs. Next, commonalities between the two patterns will be explored to see if they are truly similar patterns. Last, differences of the patterns will be enumerated to search for the uniqueness that may better fit a project in the future.

To begin, the chain of responsibility pattern at its core is just like a chain used on a bicycle, in the sense that each link is connected to the one before it and so on. However, each link in the chain has a special characteristic that binds it to one specific task. This action is known as a processing object, and the process contains specific information that defines the type of command objects that it can handle. A command object is just a task that needs to be completed by the software. To explain the process, imagine you have colored marbles in your hand with colored cups lined up in a row on a table. As you go from left to right you place the corresponding marble into the same color cup. If the first cup is green, the individual only places the green marbles into the green cup. Next, we encounter a red cup where only red marbles are placed. This pattern is continued until all the marbles in your hand are in their correct cup. This example is used to show how one cup has only one processing object or special information, that is color. The processing objects can only handle the information they are meant to handle and that is why you don't have red marbles in the green cup.

This pattern may seem very basic, but given more complicated command objects, you must create more processing objects that can handle more involved computing actions. Other styles of the chain of responsibility pattern can change the previously-used linear path into something more diverse,

which brings us to a tree of responsibility. A tree style would split the incoming command objects based on more characteristics than simply color until it finally flowed to the right processing object that is designed to fulfill the final task. Continuing the previous example, imagine you now have cubes and triangle pieces in your hands along with the marbles. Again, all pieces have different colors but our original chain cannot separate the different shapes. The tree style can break apart the command objects into subgroups of triangles, cubes, and marbles before it sorts for colors. Once they are dispatched to the proper path, they can continue in the pattern to be sorted and organized by colors and shapes. In some cases, you can even have higher-up processing objects that complete a task before it reaches the end of the tree. You can even take care of smaller forms of data in advance and creating recursion to ensure that all commands are processed completely. This will create checks along the way to guarantee that the command objects are fully broken down so that the tasks can be finished.

The second design to be examined in this paper is the composite pattern, where grouped information can be treated like single forms of information, composing small amounts of information into a tree structure to create a partial hierarchy of information. This pattern allows users to treat composite information the same as individual objects. To explain the process, look back to the example of the different colored shapes in our hand. This time the design pattern will group the objects be color, or any specific characteristic, and place those into a group where they can be processed together. It would then pass through a tree structure that can process both individual data and group data the same way. This design pattern can also include recursion to ensure completeness, and have higher-up processing objects that will complete smaller tasks before reaching the end processing object.

These two design patterns have many similarities when analyzing how they process data. Each pattern can consist of tree structures or hierarchies to better organize and complete tasks that the user requires. The organization of data is also done in a similar way because each piece of data is passed through a test and dispersed properly based on the results of the test. They may not always follow the

tree structure and move along a linear path, but the information is being processed and pushed along based on the outcome of the previous processing unit. Each process is almost identical due to recursion and higher-up processing that is possible with each design pattern. Each system has the same checks to make sure commands are done fully and has ways to complete smaller objects ahead of time before reaching the end of the system. In all, the encompassing styles of the patterns are very similar on how they organize, test, and complete command objects that are tested by each design pattern.

Differences between these two patterns include the type of data they can process, and how the data carries on after it passes through the system. The chain of responsibility pattern can complete small individual objects in a linear fashion, while composite design patterns can complete larger composite groups of information in a hierarchy style. The chain pattern is simply for smaller forms of information that can be passed along once each process completes its specific testing. This may take longer and use more resources but it allows the user to clearly specify where and when things are completed. The composite pattern is meant to uniformly handle individual and group data in the same manner to allow for only one tree structure to be used. This greatly increases time for a project because you can run all data through the same system without using any extra resources for specific forms of data.

The composite pattern and chain of responsibility are both very helpful designs in completing projects but it is clear to see there are advantages of using one over the other in certain situations. Each can fit a very specific style of processing and allows for customization to ensure completion from any task that is passed through it. One must note the resource and time differences between the two styles which is a major factor in business. Overall, the two patterns seem very similar at a basic level but are unique in organization of data and types of data that can be handled by each.

References

"Design Patterns and Refactoring." *SourceMaking*,

sourcemaking.com/design_patterns/chain_of_responsibility. Accessed 25 Sept. 2017.

"Design Patterns and Refactoring." *SourceMaking*,

sourcemaking.com/design_patterns/composit. Accessed 25 Sept. 2017.