Franklin Colton Parry

CS 3450 Software Patterns

9/24/2014

Home Work 3

The variation that makes the most sense to me is to introduce another object, a handler, which controls all of the update functions from multiple subjects. That object would implement all of the observer interfaces and then distribute the updates to each concrete observer. The concrete observers would then get the update notifications that they need form that handler object and then implement only the methods they need form the subjects they are observing.

Positives: This would still keep loose coupling, because the subjects can change their implementations as needed and still to push notifications to the handler. I think this still favors composition over inheritance because we are not making each observer implement all subject interfaces, only the ones that they want.

Negatives: The first negative is that it requires more code and introduces more bugs. While coupling is small it still exists. The concrete observer objects need to know how to go and get the data from the subjects.

n

1

1

1

n

n

<<Interface>>

Subject

add()

remove()

update()

Concrete

Subject

add()

remove()

update()

…. More function()

Concrete

Subject

update()

…. More function()

<<Interface>>

Subject

add()

remove()

update()

Handler

update()