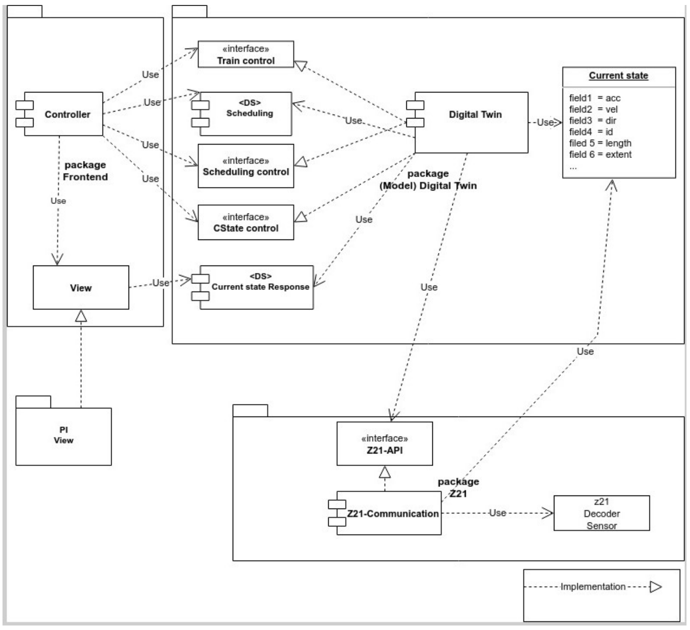
Golang: 1 Questen: Is the implementation of the observer pattern based on the following architecture in Golang correct? If not, what is the correct code.

If information is missing, it will be provided when asked.



The Current state Response data structure, which is controlled via the CState control interface, is an observer of the Current state data structure.

If the Z21 communication then reports a change, Current state Response is notified and updates its values accordingly.

Der Code:

```go
package digital_twin


import (
    "CStateControl"
    "Z21-Communication"
)


func (csr *currentStateResponse) notify() (){
    (csr.id, csr.currentSpeed) = z21Com.refresh()


}
```

```go
import (
    "CStateControl"
    "Z21-Communication"
    "time"
)


type currentState struct{
    lanSerialNumber uint
    lanVersion uint
    lanStatus bool
    locId byte
    direction byte
    trackLength uint
    maxDriveLevel byte
    driveLevel byte
    currentVelocity float32
    maxVelocity float32
    locMaxBrakeAcc float32
    locMaxAcc float32
    OberserverList [] Observer
}

var cs currentState

type Oberserver interface {
    notify()
}

func (csr *currentState) addObserver(o Observer){
    csr.ObserverList = append(cs.ObserverList, o)
}


func (cs *currentState) removeObserver(o Observer) {
    var index int
    for i, observer := range cs.OberserverList {
        if observer == o {
            index = i
            break
    }

    cs.ObserverList = append(cs.OberserverList[:index],
    cs.ObserverList[index+1:])
}


func (cs *currentState) notifyOberservers() {
        for _, observer := range cs.OberserverList {
            observer.notify()
        }
}
```

2nd Question: The Z21-Communication should update the Current state data structure every 250 ms with regard to the current speed of the locomotive.

Is the observer pattern suitable under these circumstances?

If so, does an additional data structure such as a ring need to be implemented in the Current State data structure?