

Exercise:

- **In telephone switching there is often a call object that keeps track of the state of a call(idle, offhook, dialing, busy, ...) and services (call-forward, call-wating, busy-call-return, ...) that want to activate when certain states are reached.**
- **It is desirable for services to have some context to refer to so they can determine when to activate. Also services should not interfere with, or be coupled to, one another**
- **Sketch out a design for service objects based on the Observer and Mediator patterns -review the intent of the patterns**

OLD Code Structure (Anti-Pattern)

Call

```
Event event;  
State state;  
handle_event(Event e){  
...  
if(three_way_call == running)  
    three_way_call(e);  
else if(call_waiting == running)  
    call_waiting(e);  
else if(message_answer == running)  
    message_answer(e);  
...  
}
```

Three Way Call Service

```
Event event;  
State state;  
three_way_call(Event e){  
...  
if(call_waiting == running)  
    deny_service();  
else if(call_forward == running){  
    call_forward(e);  
    make_3waycall(e);  
}  
else if(message_answer == running)  
    deny_service(e);  
...  
}
```

Given N Services:

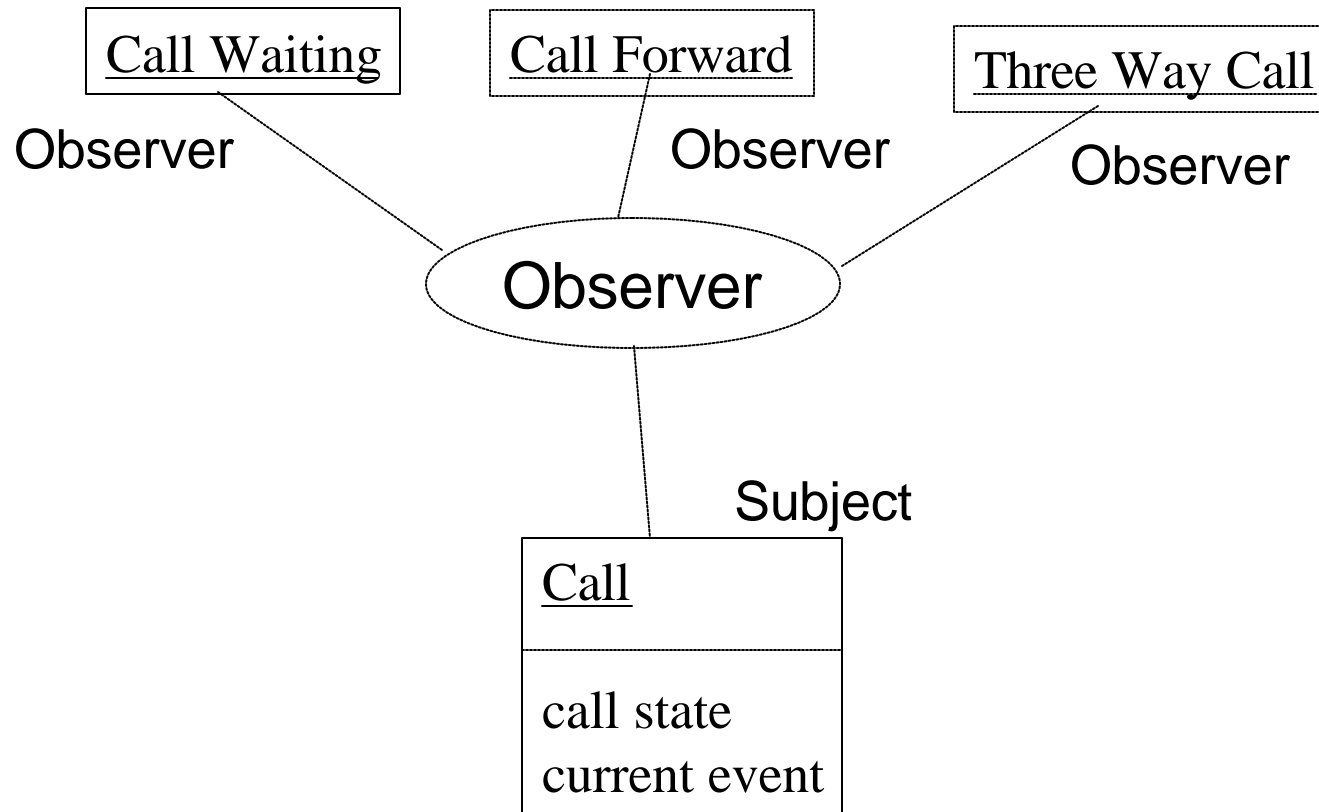
Call object is $O(N)$ code complexity

Each Service Object is $O(N)$ code complexity

Overall: $O(N*N)$ Code complexity or worse

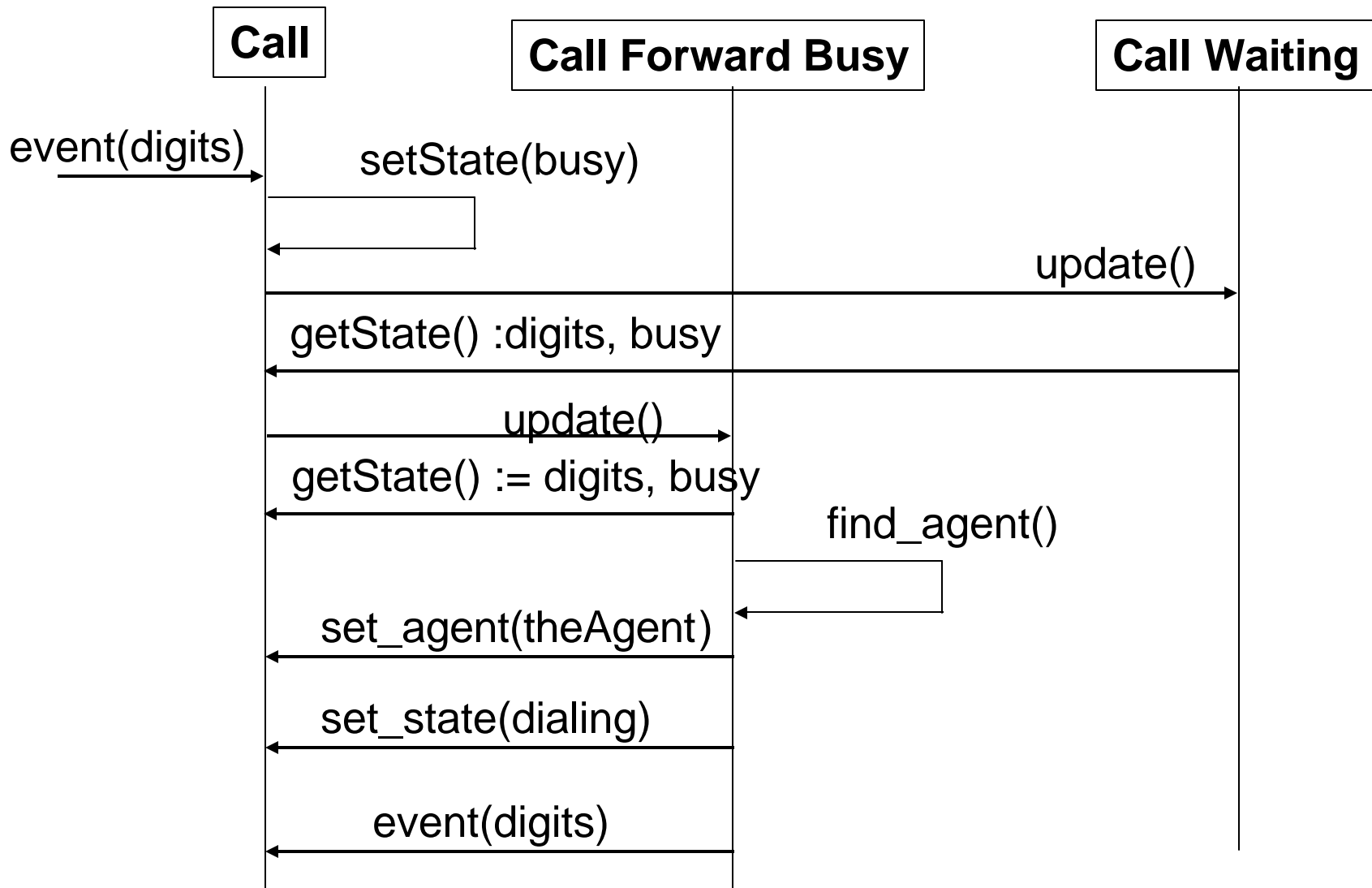
$O(N*N)$ Testing complexity or worse

Exercise part 1- services as observers

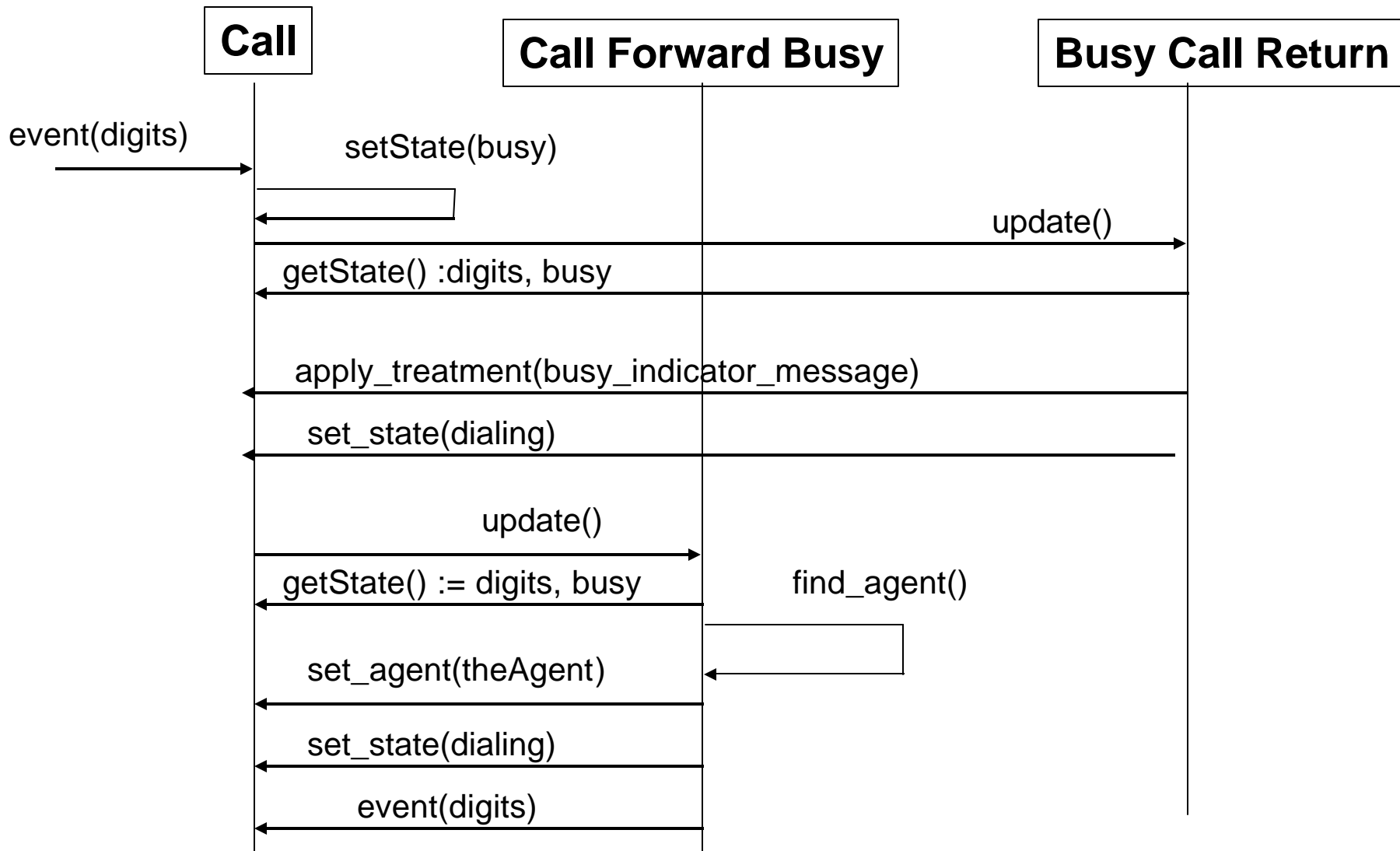


**Call object: O(1) complexity, Each Service O(1) complexity.
Overall O(N) code and Test complexity**

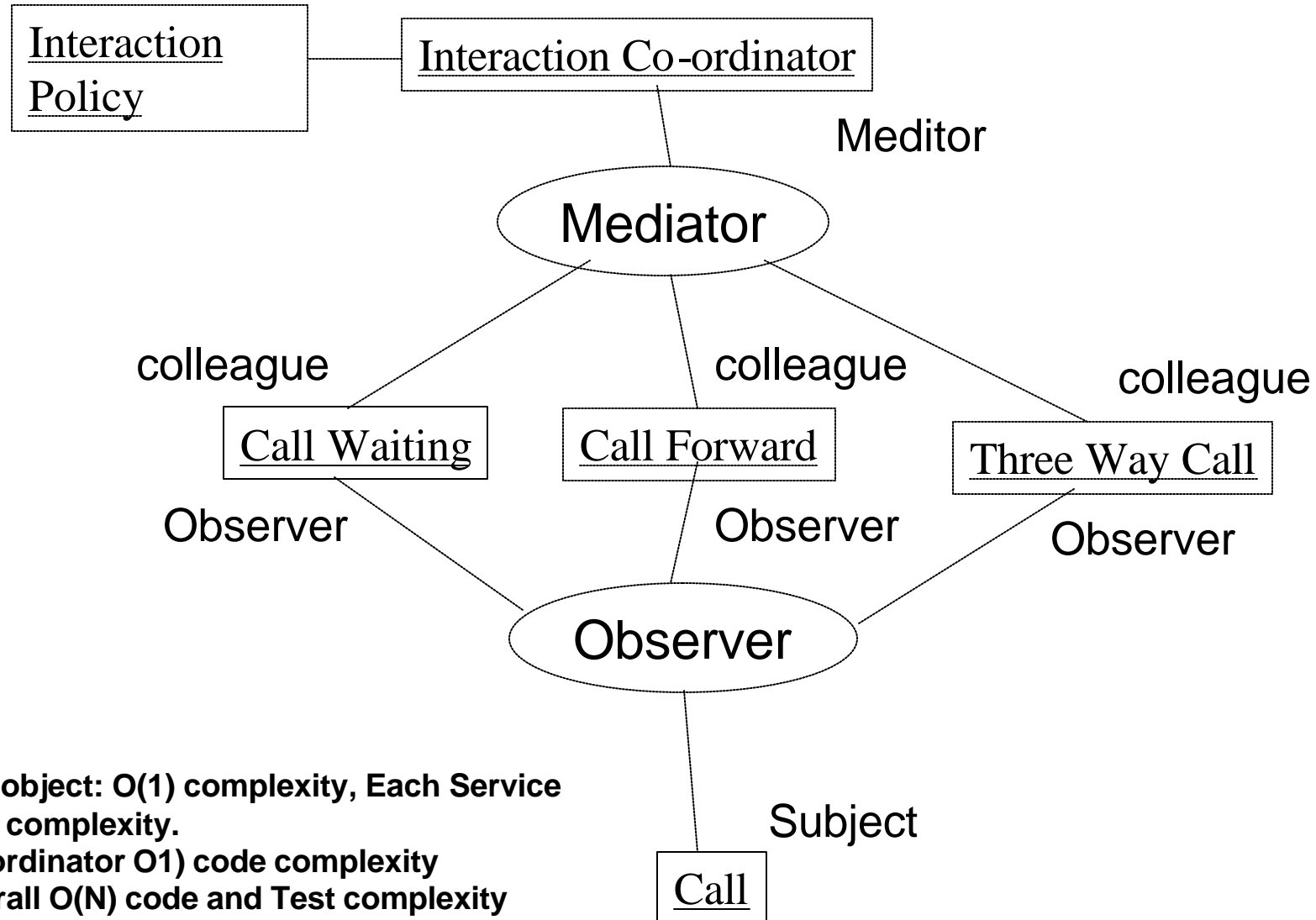
Services as Observers



Services as Observers - Interaction Problem



Exercise part 2- mediating service interactions



Call object: $O(1)$ complexity, Each Service $O(1)$ complexity.
Co-ordinator $O(1)$ code complexity
Overall $O(N)$ code and Test complexity
Any $N*N$ complexity is represented in the Policy Data

Mediating service interactions

