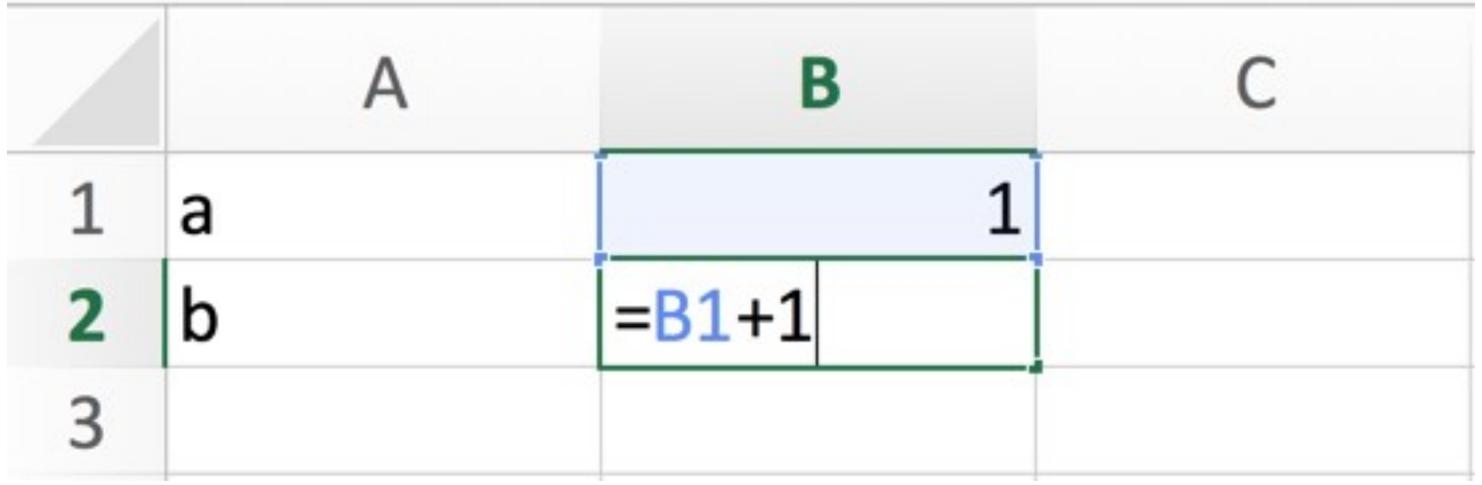
REACTIVE PROGRAMMING

Concept



	A	B	C
1	a	1	
2	b	=B1+1	
3			

Observable: B1

Observer: B2

Event: B1 value

Function: B2's Formula

How does it differ from the delegate Pattern?

Delegate pattern: 1-1

Observer: 1-n

Proactive/passive versus Reactive

Scenario: CoreData & Network... <> UI

Proactive

You need to maintain states (UI state, Network data, DB data) to coordinate the different flows of information

Mutable states = more difficult to test

1 Boolean to test = 2 pow(1) values to test

x Boolean to test = 2 pow(x) values to test

Pros

1. # Easy to understand

Cons

1. # Tight coupling between components
2. # Difficult to test

2

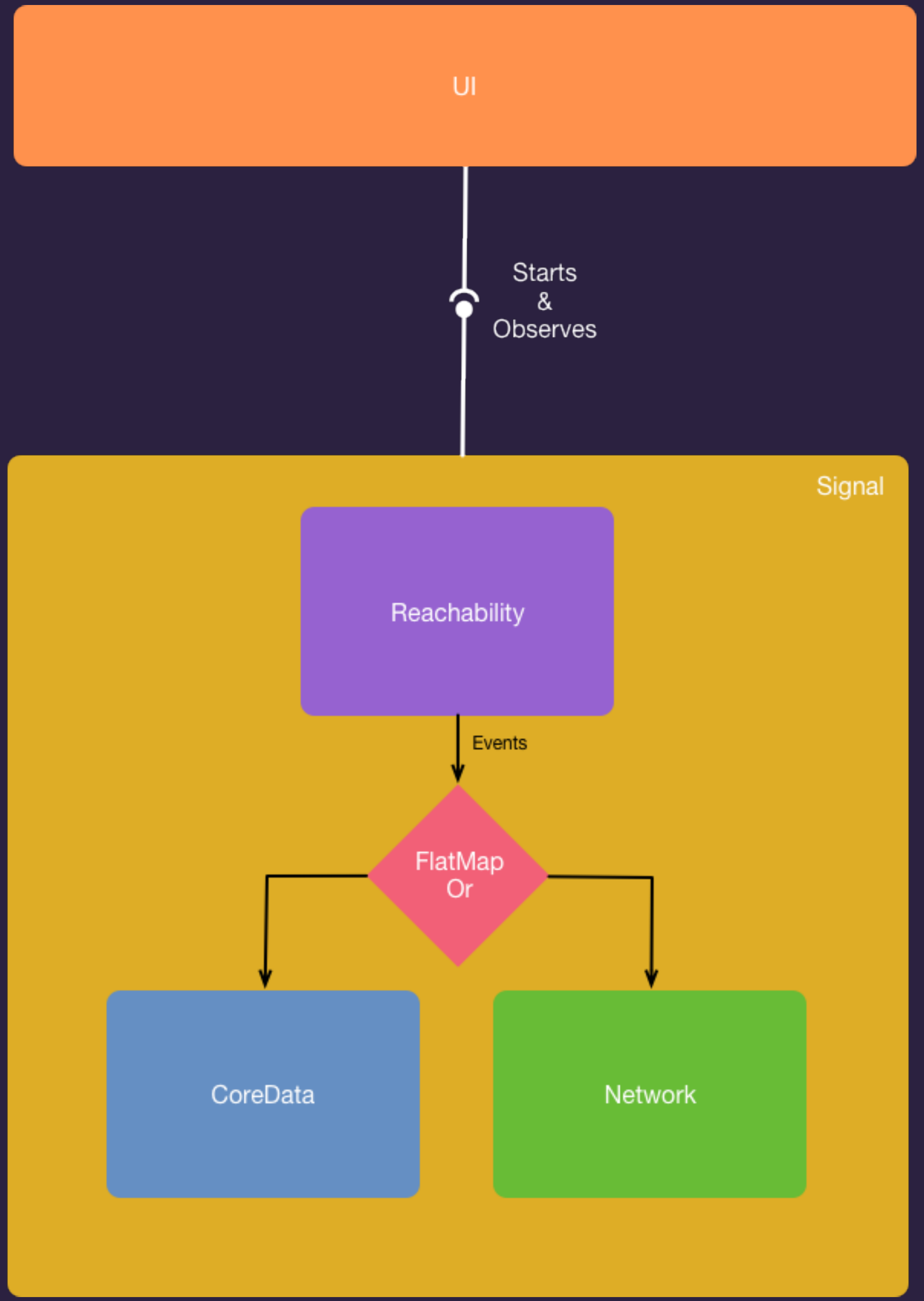
Reactive

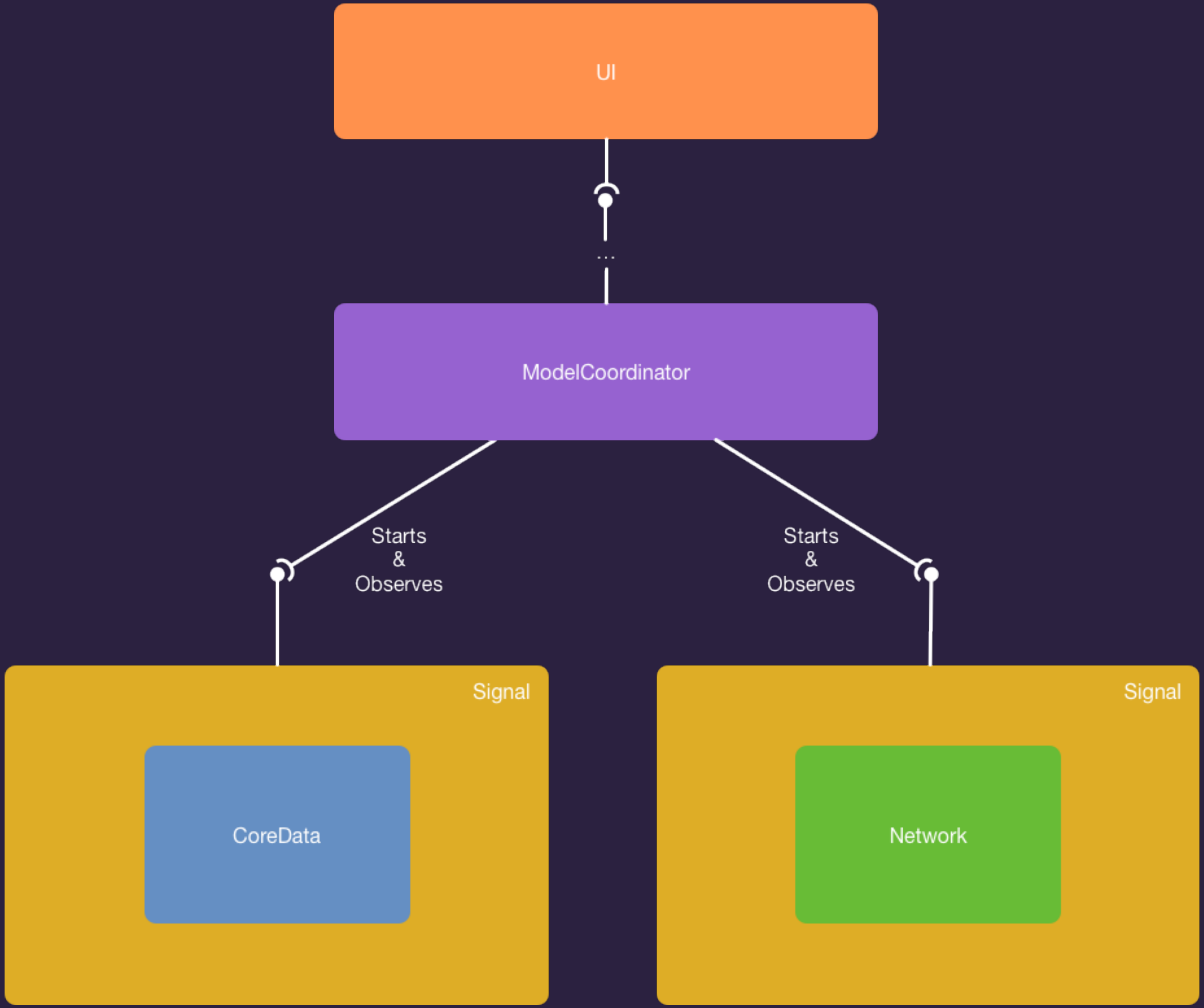
Pure functions ❤ Immutability

We don't maintain states

We react to events when they happen

WAIT WHAT?


UI
Starts
&
Observes
Signal
Reachability
Events
FlatMap
Or
CoreData
Network


UI
...
ModelCoordinator
Starts & Observes
Starts & Observes
Signal
CoreData
Signal
Network

Pros

1. # Less side effects
2. # Modular code & loosely coupling between mods
3. # Inherently asynchronous
4. # A lot of operators to work with

Cons -> Rx's learning curve

Some operators

defer

from

interval

just

range

repeat

timer

buffer

map

- groupBy
- scan
- reduce
- window
- debounce
- distinct
- elementAt
- filter
- first

- sample
- skip
- skipLast
- skipWhile
- take
- takeList
- takeUntil
- combineLatest
- zip

- amb
- startWith
- do
- observeOn
- subscribeOn
- delay
- publish
- throttle
- timestamp

Some links

Reactivex.io defines a common API for Rx implementation

RxMarbles.com to figure out what operator does what