Untitled

Why This Topic

What were Taught about OOP

- Inheritence
  - The road to reuse, inheritance hieracies
- Encapsulation
  - Hide those fields
- Polymorphism
  - Animals Speak!

What’s not Covered

- Abstractions
    - What’s the proper level of abstractions
- Composition
- The power of polymorphism glossed over
- What should really be encapsulated
- How to Reduce Coupling

Coupling & Cohesion

- Cohesion: components that are self-contained, independent and with a single, independent purpose
- Coupling: a measurement of the effect of changing a component in your system.
  - Changing on component of system requires changing the elements that utilize the component

Abstractions, Abstractions, Abstractions

- How we model the world we are creating
- Allows for Higher levels of expression
- Easier to maintain over Primitives
- Proper encapsulation

Common Themes Regarding Abstractions

- From Clean Code
  - “... Clean code never obscures the designer’s intent but rather is full of crisp abstractions and straightforward lines of control.” -- Grady Booch
  - “Reduced duplication, high expressiveness, and early building of simple abstractions. That’s what makes clean code for me” -- Ron Jeffries

Avoiding Primitive Obsession

- Use objects to represent concepts in your system
  - DateOfBirth, Money, TimePeriod, Address, ZipCode
- Encapsulate parameter lists to a parameter aggregation object

Encapsulation

- Not Data Hiding
- Encapsulate the correct state of an object

Law of Demeter

- Methods only talk to to members of it's object, parameters passed in, and objects it creates
- Do not expose the internal state of a parameter
    - order.LineItems.Count *probably* does not break encapsulation
    - customer.Wallet.Cash *probably* does

Dependency Inversion

- Abstractions should be defined by a contract, not by an implementation
  - Interfaces, Abstract Classes
  - Code Contracts can define invariants, pre-conditions & post-conditions
- Objects that use the abstraction, should only know about the contract, not the implementation
- Remove the “new” keyword

IOC Containers

- Inversion of Control is a Design concept
- Dependency Injection is a pattern to implement IoC
    - Constructor Injection
    - Setter Injection
- As an Application gets more complex, the number of dependencies gets larger and larger
- IOC Containers track dependencies and fill in the concrete implementations for you, based on configuration

Modern IOC Container features

- Allow Convention based configuration
- Configuration through a dsl or xml
- Allow for AOP techniques (Interception)
- Manage object life cycle
- “Profile” based configuration

Composition over Inheritance

- Deep Inheritance Models are not what they're cracked up to be:
  - Very Rigid
  - Only one point of variation
-

Polymorphism instead of Conditionals

- Avoid using switch statements (or if / else if) to change behavior
- Can do the same thing by applying polymorphism

Anemic Domain Model Anti-Pattern

- Domain Objects used as data containers
- Business Logic is done in “Service” objects
- Domain models cannot guarantee they are in a valid state
- Service Objects should should retrieve information an domain object needs to perform an operation and pass to the domain object altogther
- Or better yet, get rid or your Service classes altoghter

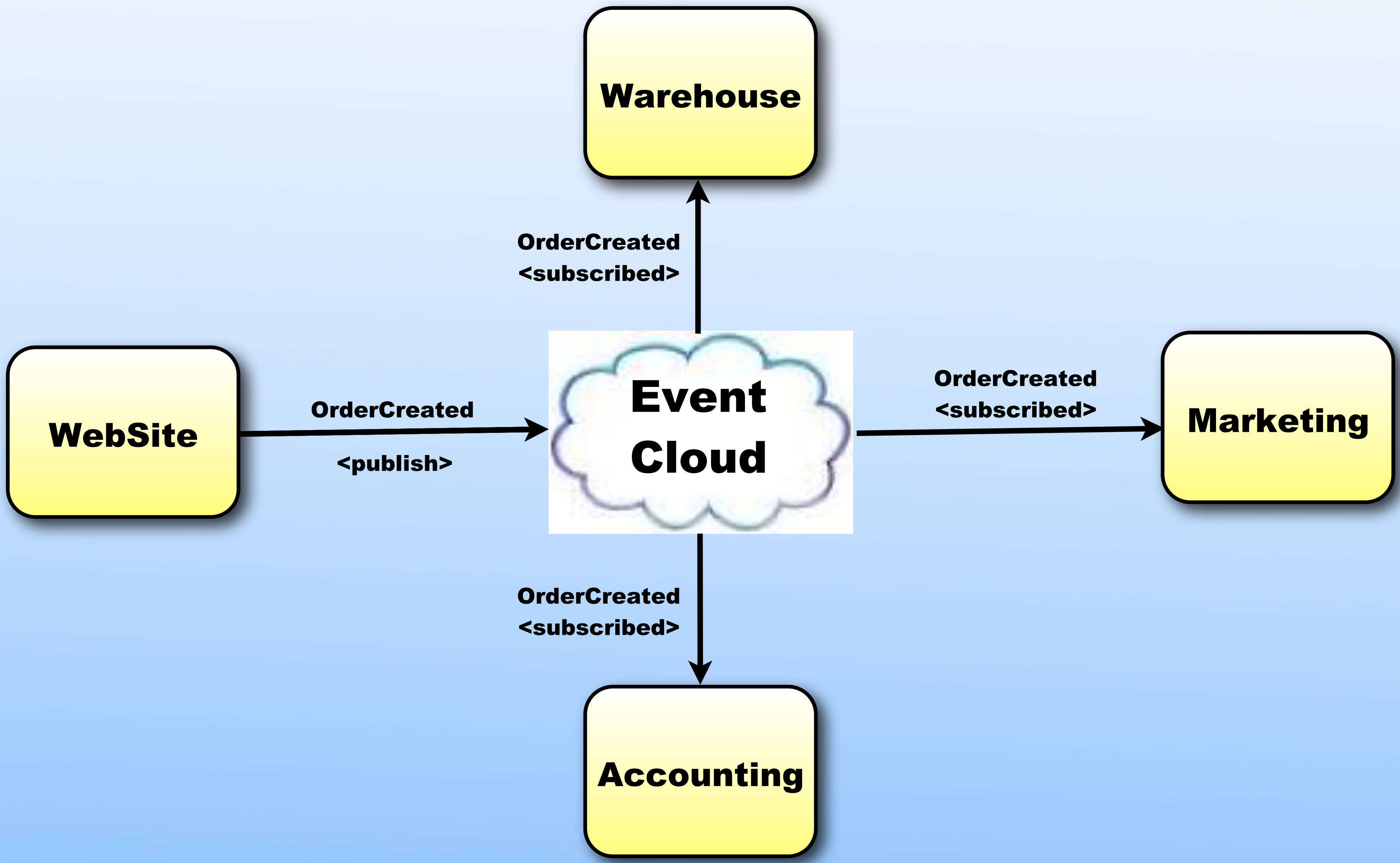
Command Query Separation

- Methods should either query for information or perform actions, but not both

Event Driven Architecture

- Style of Application Design where the application responds to events that occur in the system
    - OrderRecieved, OrderCancelled
- Allows for separation between the caller and the responder
- Allows for increased scalability
    - Responder could be on another machine
    - Could be more the one responder

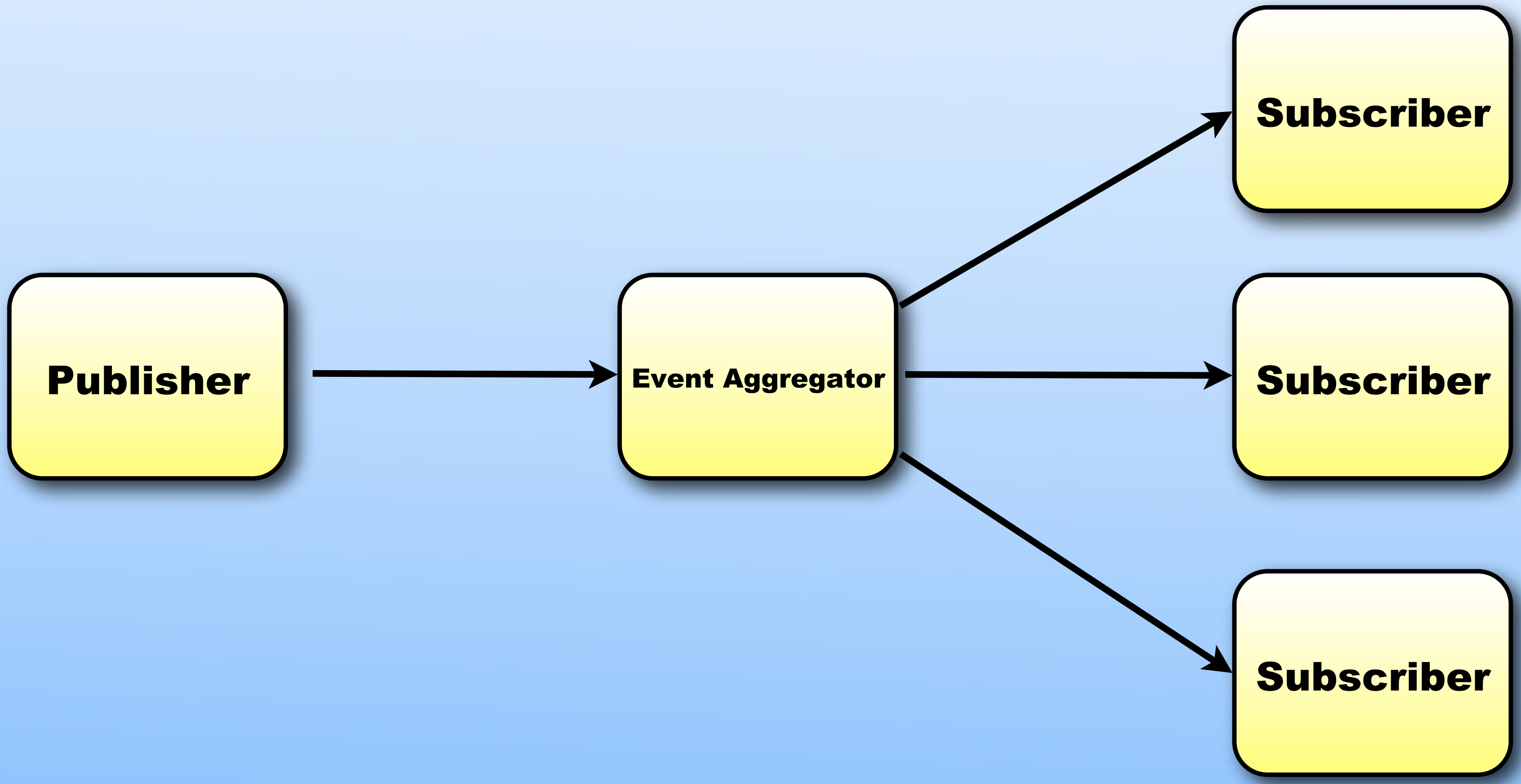
Connected Through Events



Warehouse
OrderCreated <subscribed>
WebSite
OrderCreated
<publish>
Event Cloud
OrderCreated <subscribed>
Marketing
OrderCreated <subscribed>
Accounting

Publish / Subscribe

- One Way Messaging
- 1 Publisher: N Subscribers
- Publisher is Not Coupled To Subscribers


Publisher
Event Aggregator
Subscriber
Subscriber
Subscriber