

Domain-Driven Rails Redux

Pat Maddox
RailsConf 2010

A Short Story

**Software is a learning
process**

Design matters

**Complexity inherent in
the system**

**All complexity not
created equal**

“I need to do

because the code needs

”

“Coding into submission”

“I need to do



because the biz needs

”



5 Whys

Abstraction and modularity

**SOLID, DRY,
connascence**

What's next?

**Understand the
business context**

Core domain

Ubiquitous Language

Conway's Law

**How does your
business work?**

**Answer Yes/No
questions with
“What if?”**

**Don't solve business
problems with technical
solutions**

Organizing complexity

```
class User < ActiveRecord::Base
  named_scope :sorted_by_name,
              :order => 'last_name, first_name ASC'

  named_scope :sorted_by_member_points,
              :order => 'member_points DESC'
end
```

```
class WillcallController
  def index
    @users = @event.users.sorted_by_name
  end
end
```

```
class WillcallMailer
  def willcall_email
    users = @event.users.sorted_by_name
  end
end
```

```
class HorribleLegacySystemIntegrator
  def send_attendee_list
    users = @event.users.sorted_by_name
  end
end
```

```
class WillcallController
  def index
    @users = @event.users.sorted_by_name
  end
end
```

```
class WillcallMailer
  def willcall_email
    users = @event.users.sorted_by_name
  end
end
```

```
class HorribleLegacySystemIntegrator
  def send_attendee_list
    users = @event.users.sorted_by_name
  end
end
```

Incidental Coupling

```
class WillcallController
  def index
    @users = @event.users.sorted_by_name
  end
end
```

```
class WillcallMailer
  def willcall_email
    users = @event.users.sorted_by_name
  end
end
```

```
class HorribleLegacySystemIntegrator
  def send_attendee_list
    users = @event.users.sorted_by_name
  end
end
```

```
class User < ActiveRecord::Base
  named_scope :sorted_by_name,
              :order => 'last_name, first_name ASC'


  named_scope :sorted_by_member_points,
              :order => 'member_points DESC'

  def self.sorted_by_mcname(*args)
    all(*args).sort_by { |u|
      McName.new(u.last_name, u.first_name)
    }
  end
end
```

```
class User < ActiveRecord::Base
  named_scope :sorted_by_name,
    :order => 'last_name, first_name ASC'

  named_scope :sorted_by_member_pos,
    :order => 'member_pos DESC'

  def self.sorted_by_mcname(*args)
    all(*args).sort_by { |u|
      McName.new(u.last_name, u.first_name)
    }
  end
end
```



```
class User < ActiveRecord::Base
  named_scope :sorted_by_name,
              :order => 'last_name, first_name ASC'

  named_scope :sorted_by_member_points,
              :order => 'member_points DESC'

  def self.sorted_by_name_for_legacy_system(*args)
    all(*args).sort_by { |u|
      McName.new(u.last_name, u.first_name)
    }
  end
end
```

```
class WillcallController
  def index
    @users = @event.users.sorted_by_name
  end
end
```

```
class WillcallMailer
  def willcall_email
    users = @event.users.sorted_by_name
  end
end
```

```
class EventShowtime
  has_many :users

  def willcall_list
    users.sorted_by_name
  end
end
```

```
class WillcallController
  def index
    @users = @event.willcall_list
  end
end
```

```
class WillcallMailer
  def willcall_email
    users = @event.willcall_list
  end
end
```

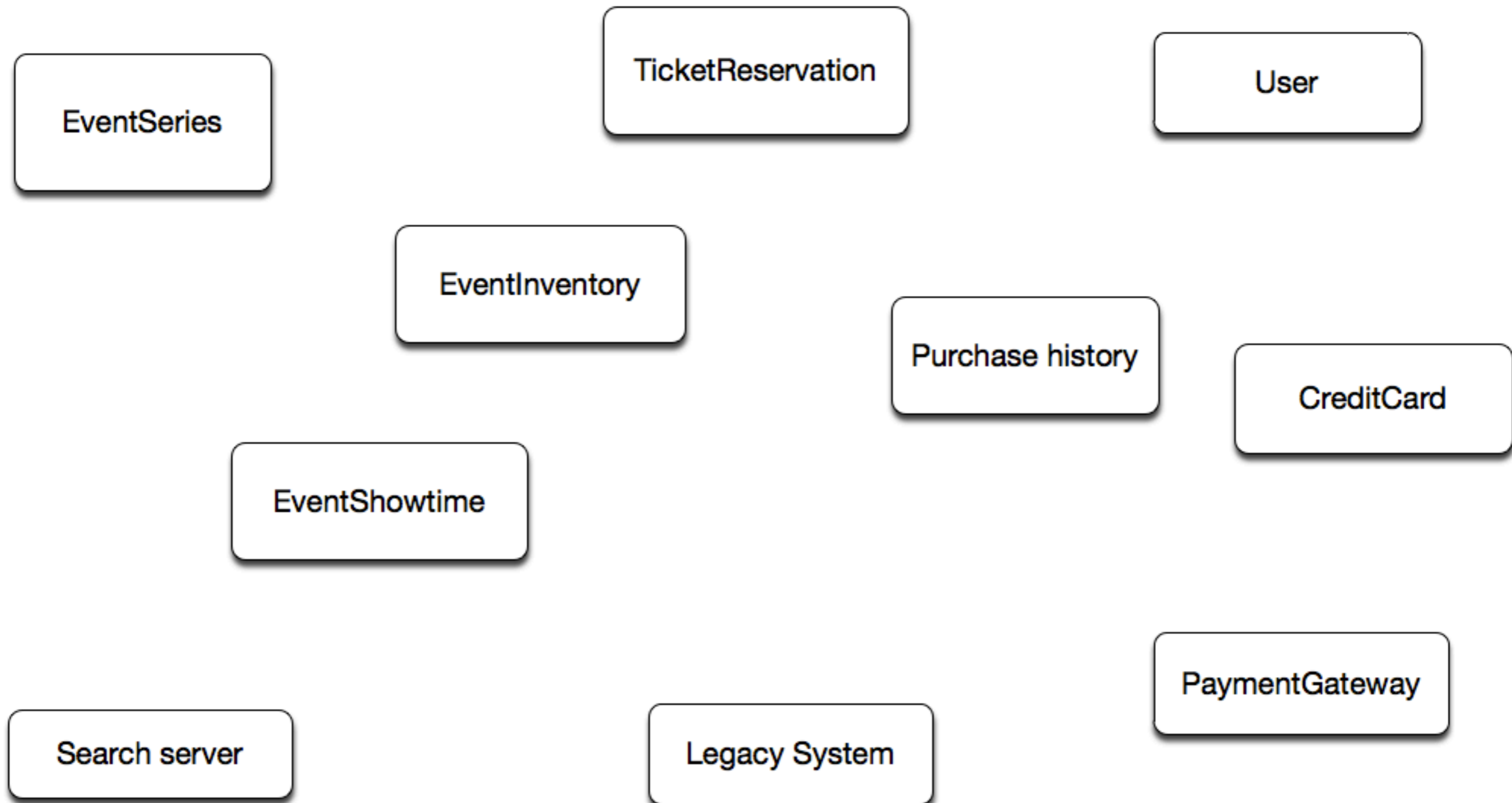
Connascence of Meaning

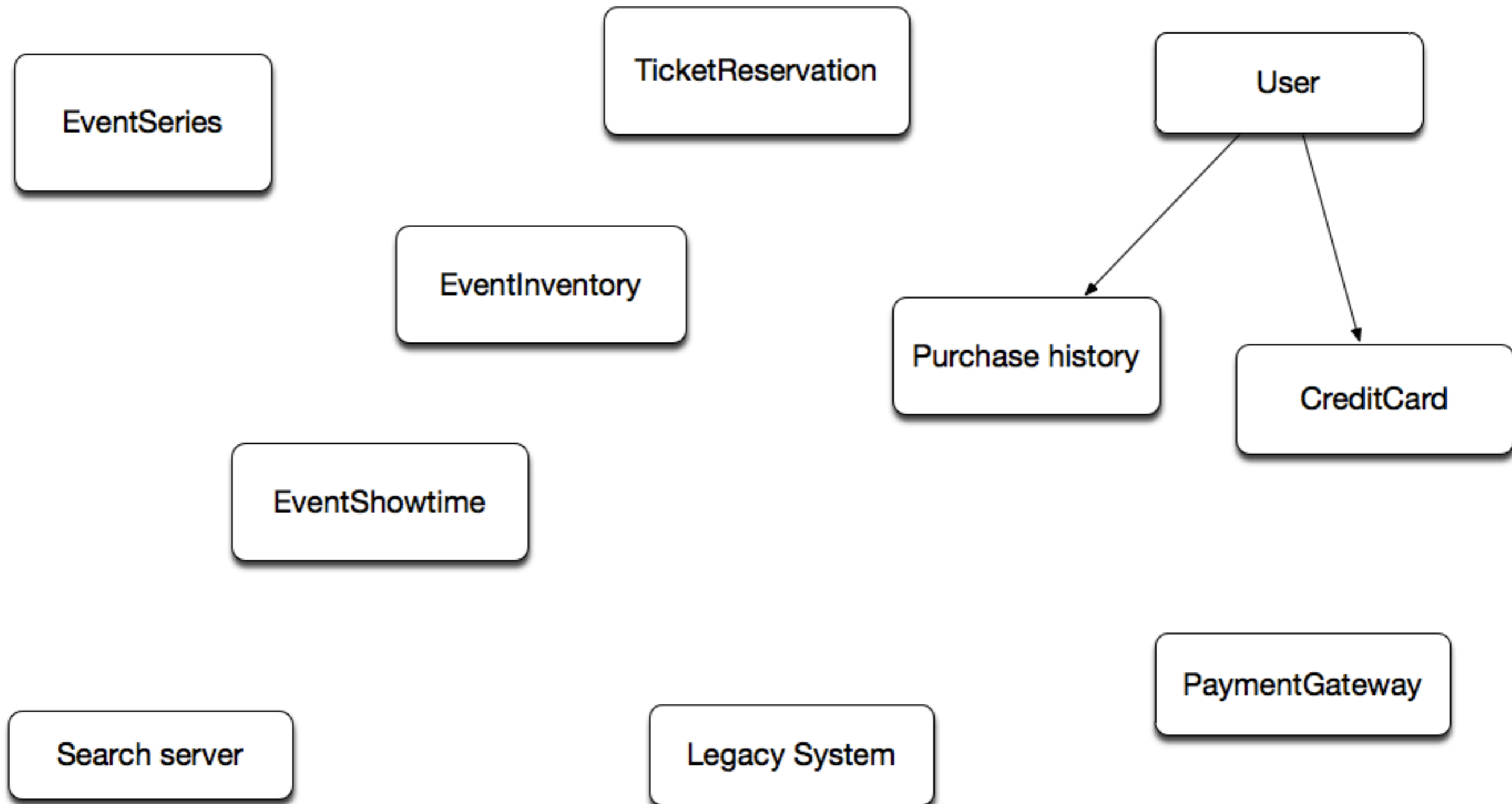
```
class WillcallController
  def index
    @users = @event.willcall_list
  end
end
```

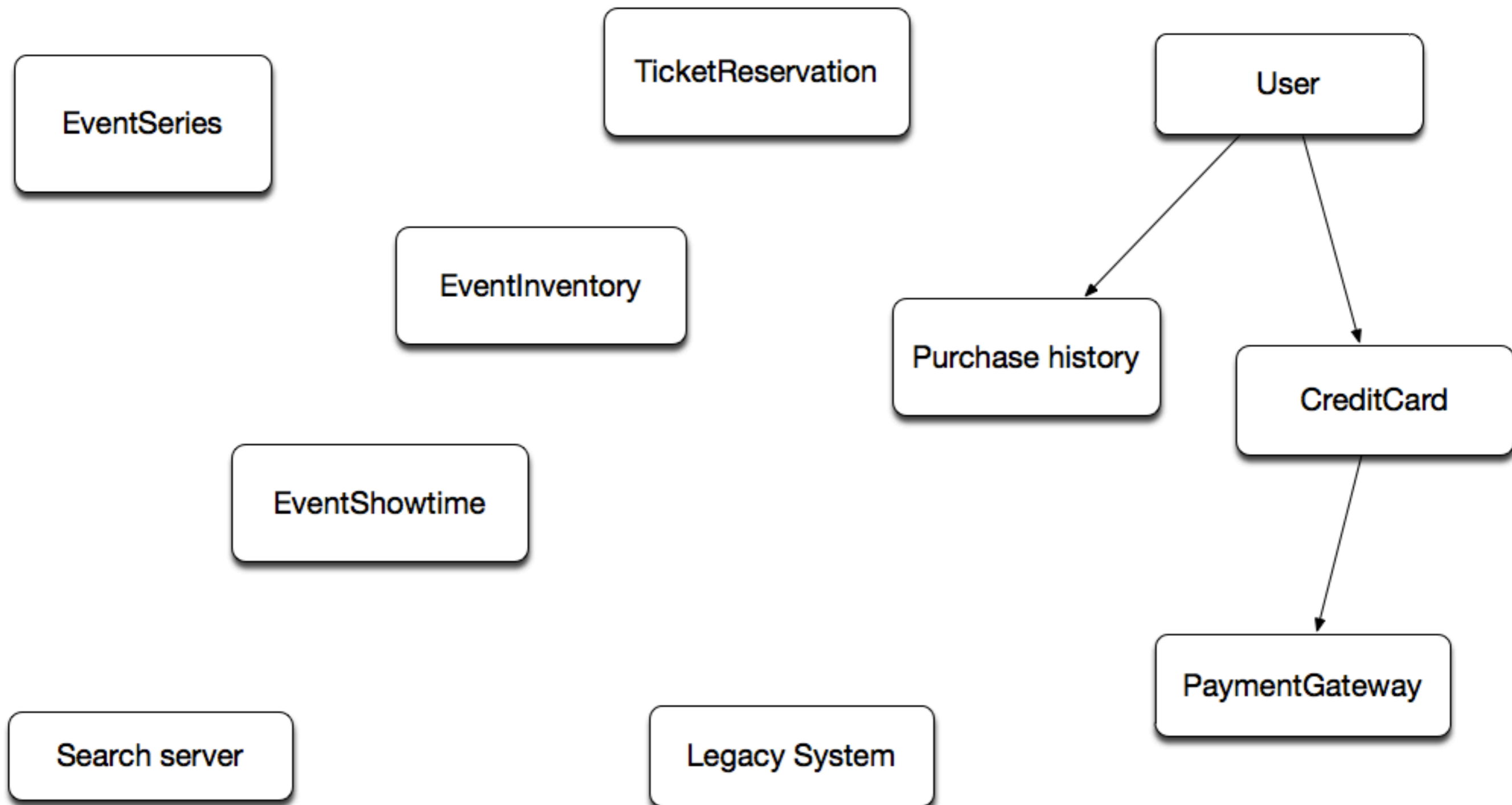
```
class WillcallMailer
  def willcall_email
    users = @event.willcall_list
  end
end
```

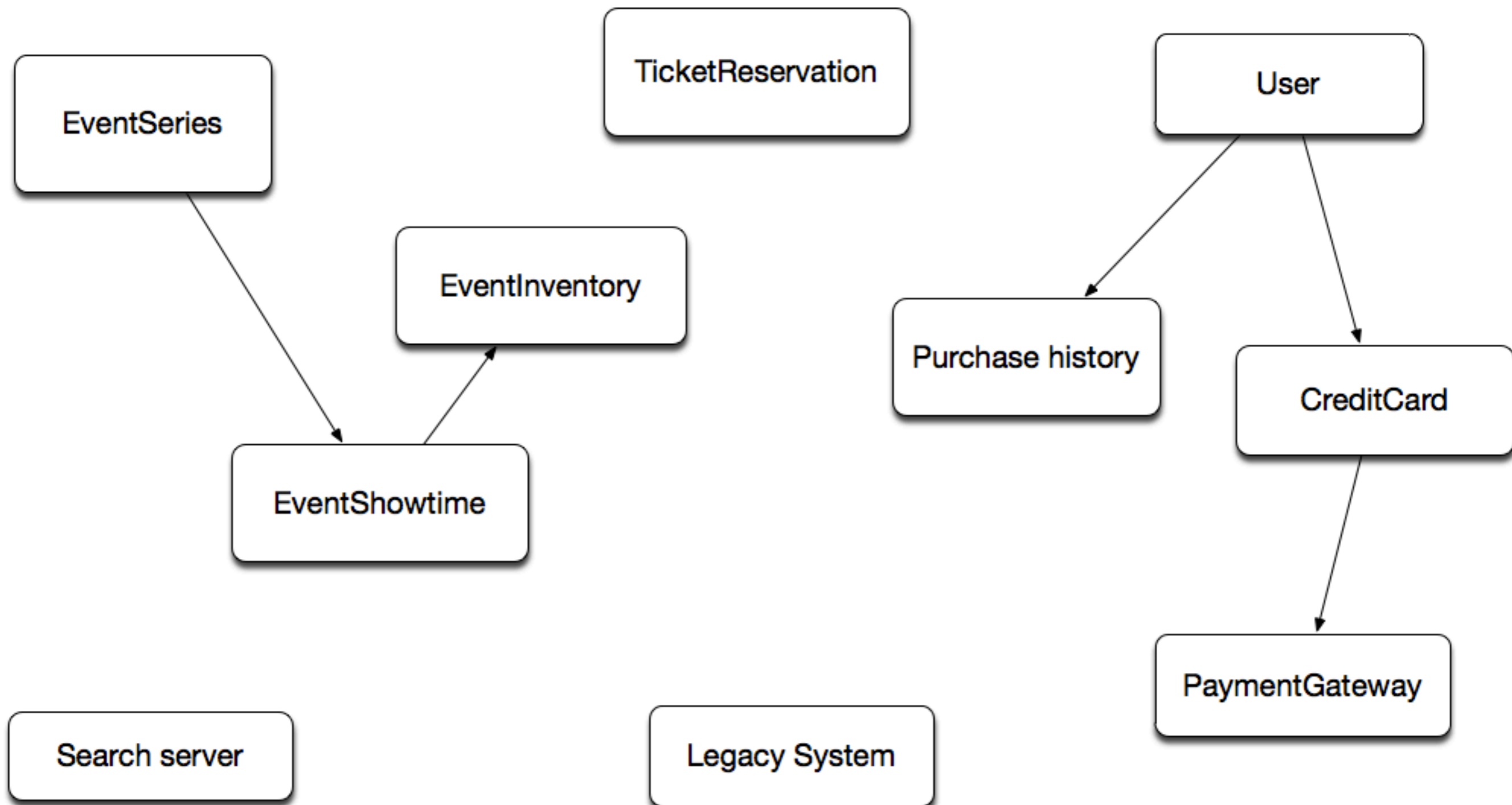
Bounded Contexts

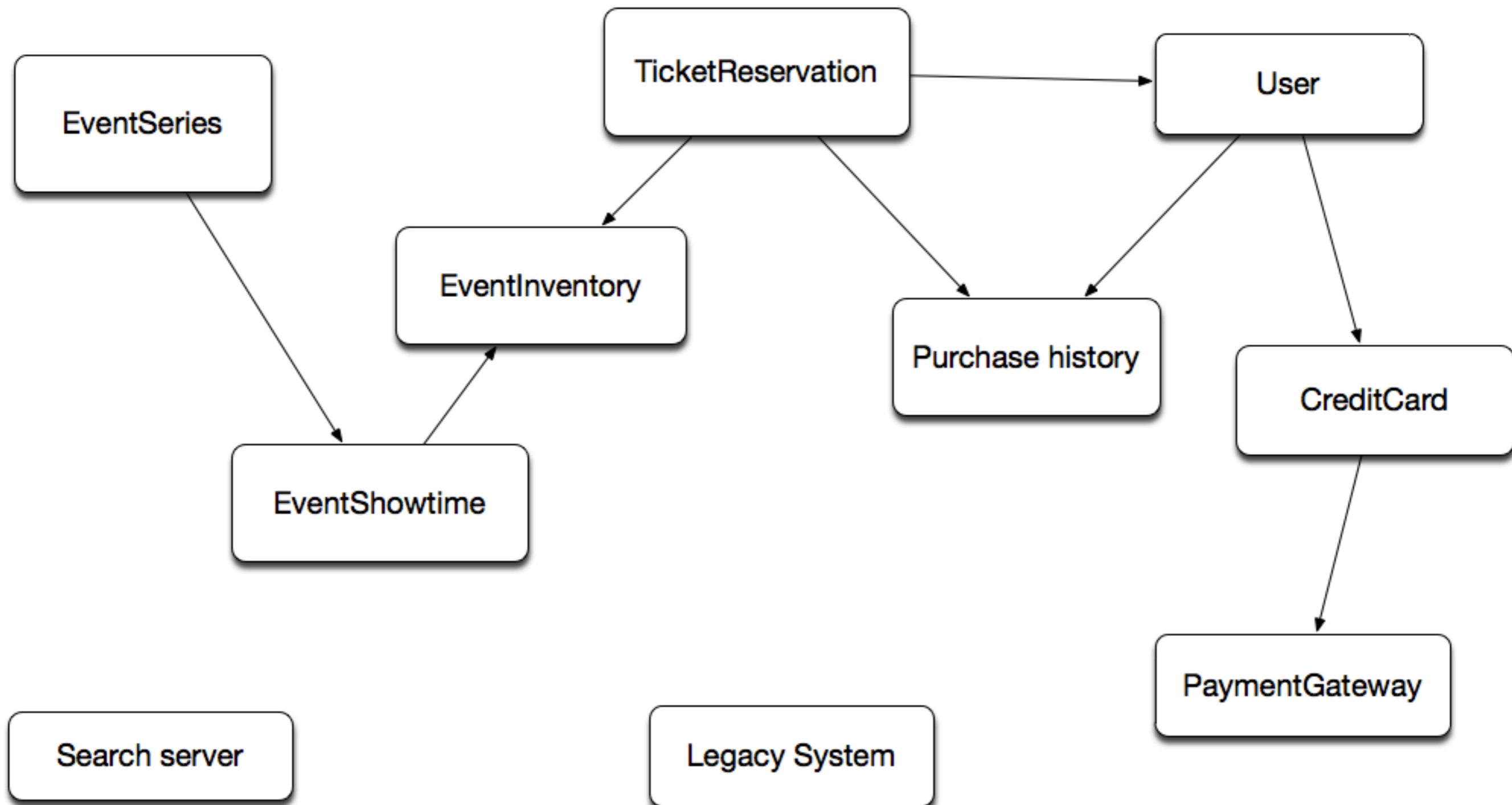
**Models exist within a
context**

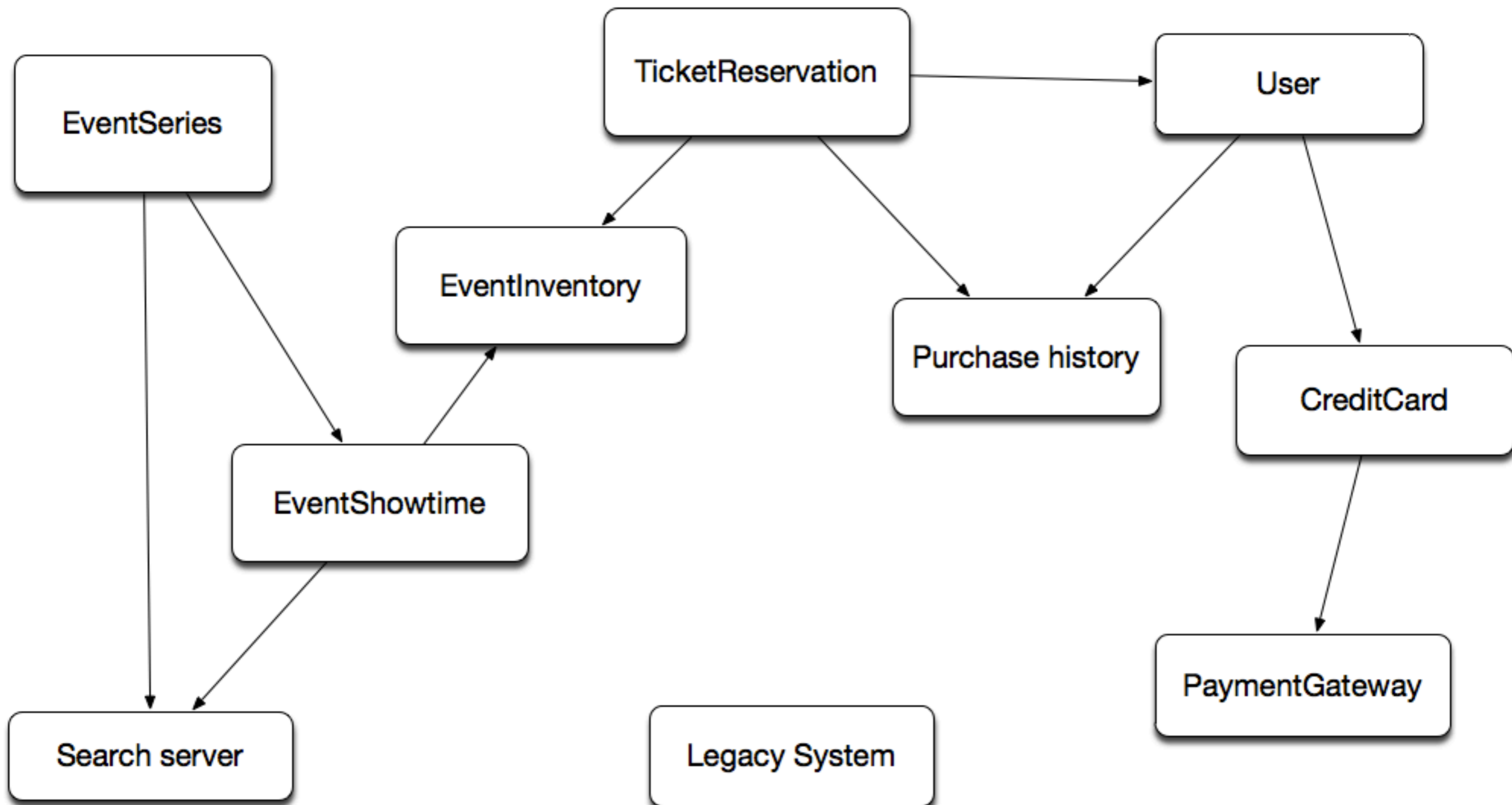


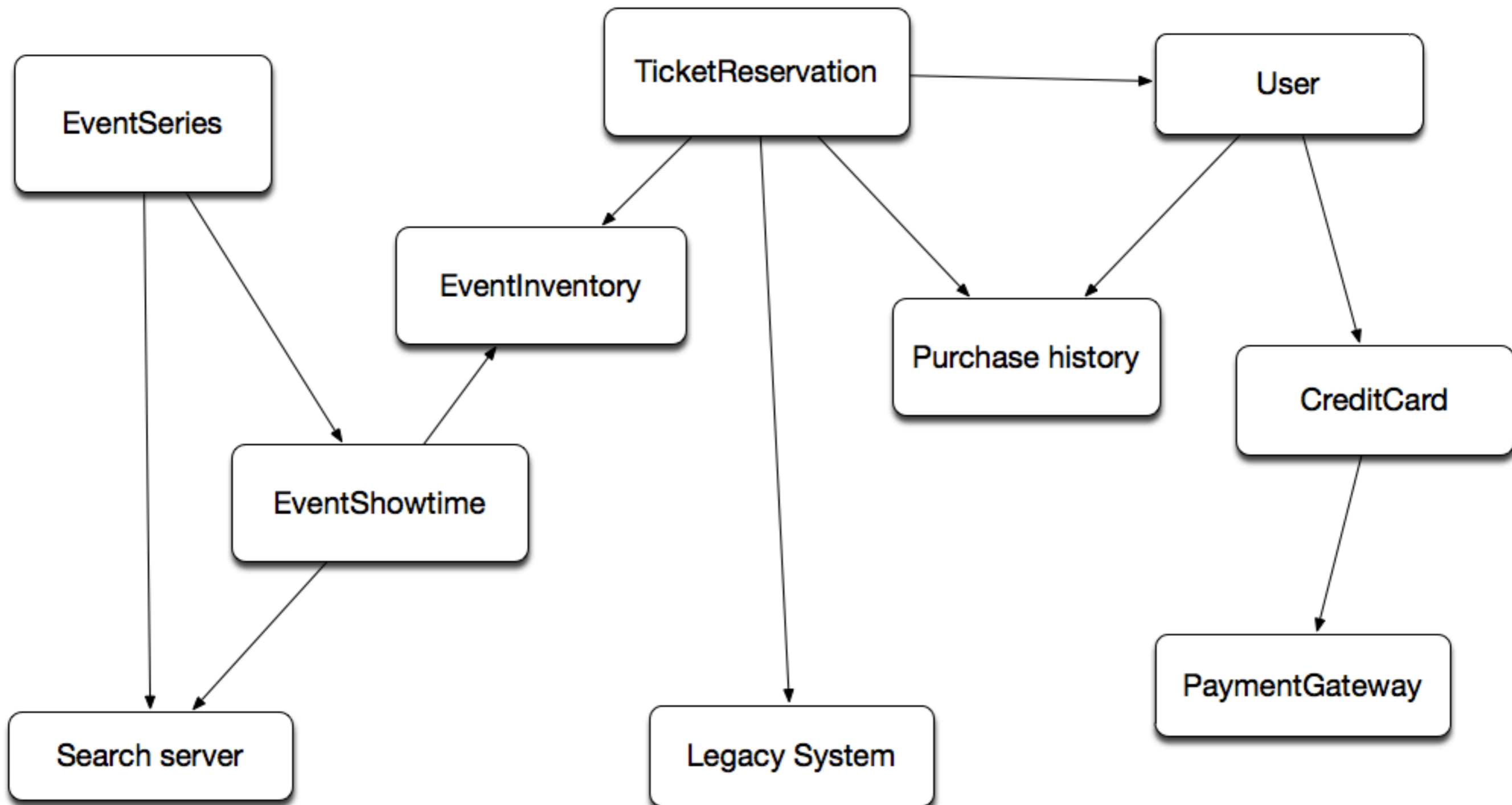


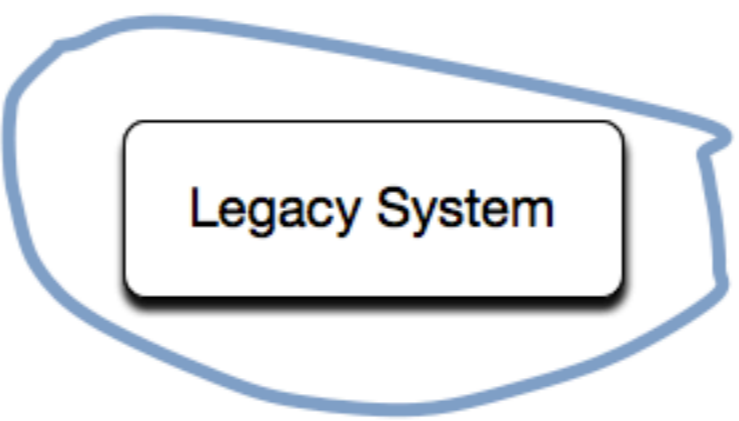
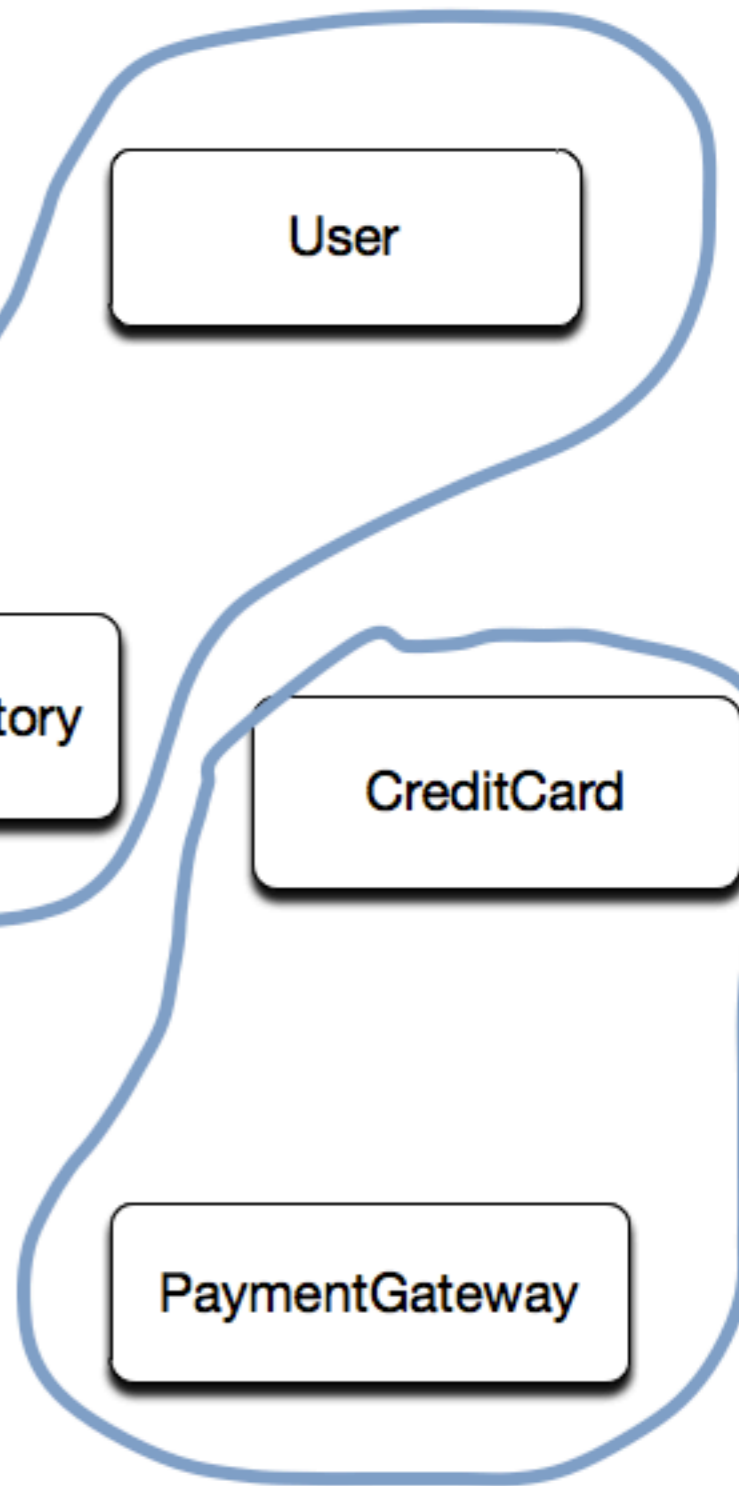
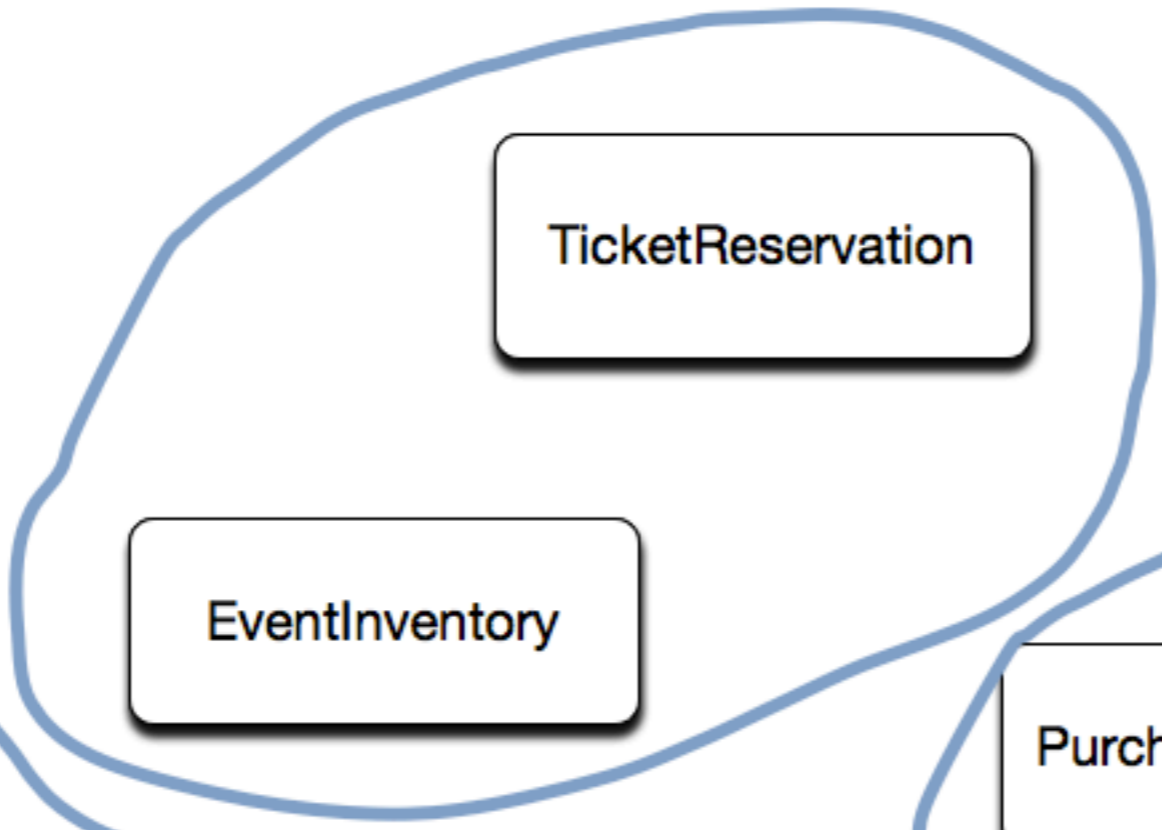
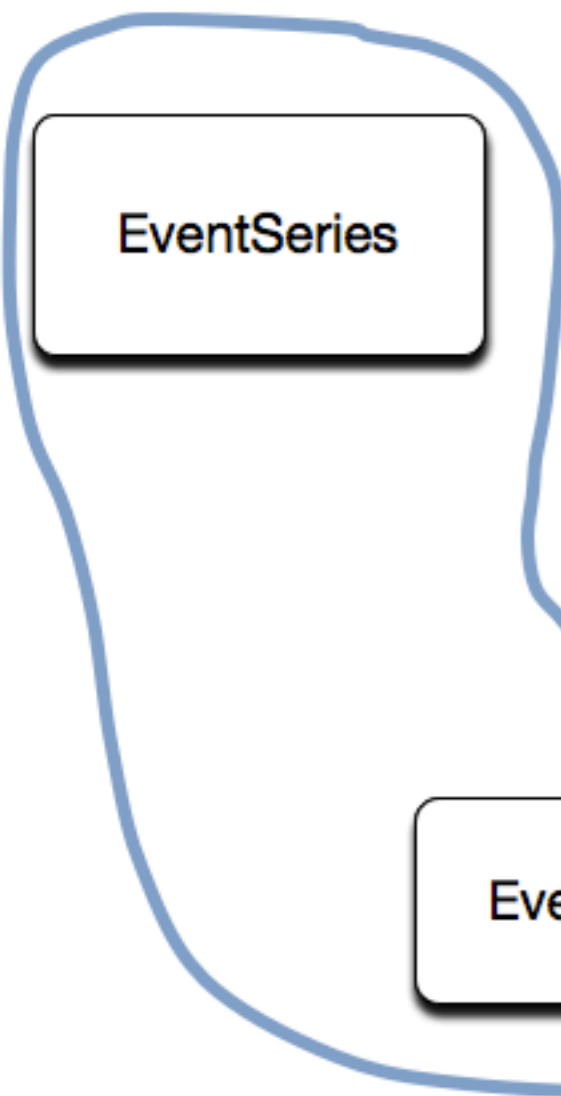












```
module Willcall
  class User < ::User
    default_scope :order => 'last_name, first_name ASC'
  end
end
```

```
module Promotions
  class User < ::User
    default_scope :order => 'member_points DESC'
  end
end
```

```
module HorribleLegacySystem
  class User < ::User
    def all(*args)
      super.sort_by { |u|
        McName.new(u.last_name, u.first_name) }
    end
  end
end
```

```
module Willcall # app/models/willcall/user.rb
  class User < ::User
    default_scope :order => 'last_name, first_name ASC'
  end
end
```

```
module Willcall # app/controllers/willcall/willcall_controller.rb
  class WillcallController
    def index
      @users = @event.users # Willcall::User
    end
  end
end
```

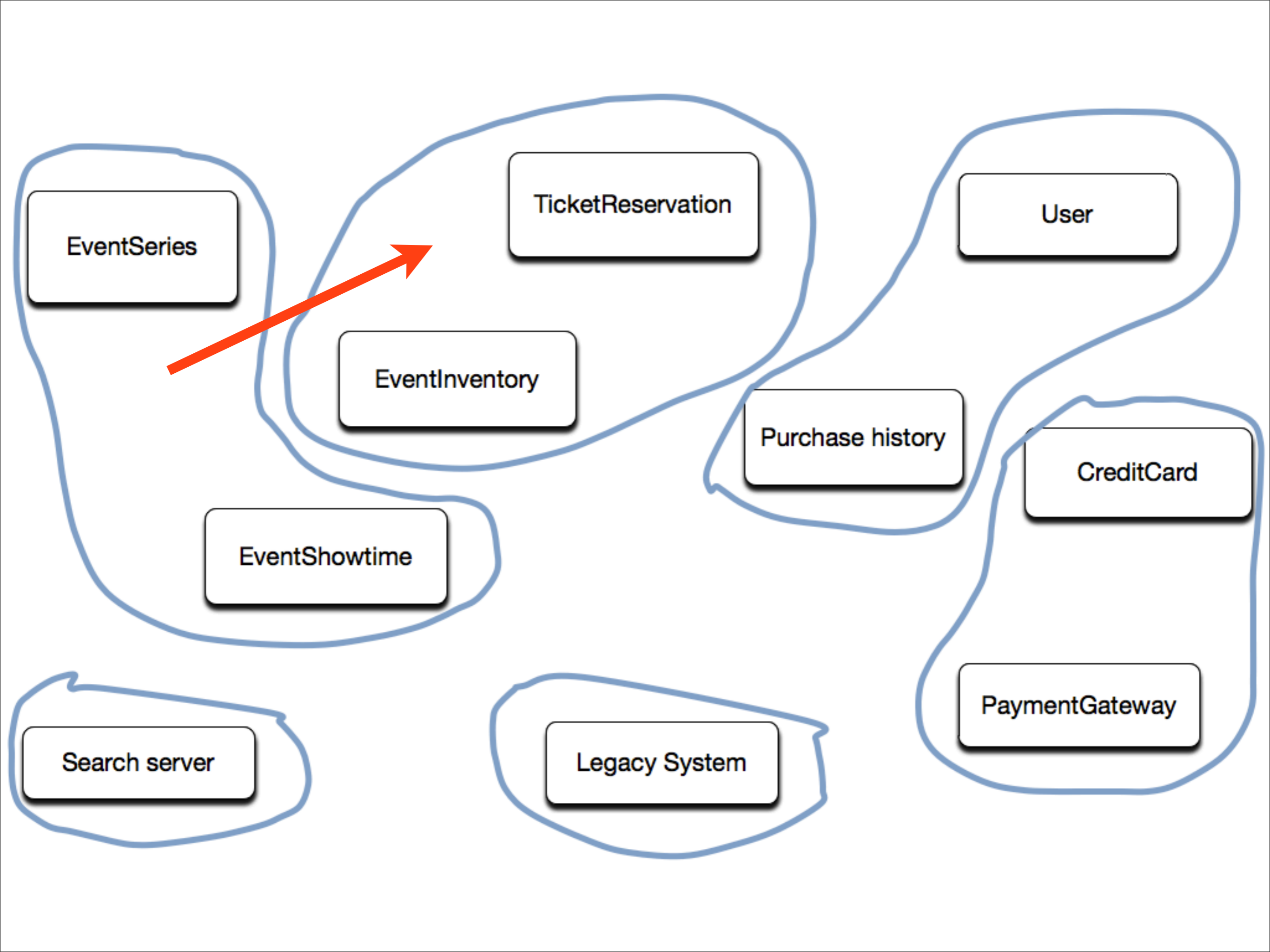
```
module Willcall # app/models/willcall/willcall_mailer.rb
  class WillcallMailer
    def willcall_email
      users = @event.users # Willcall::User
    end
  end
end
```

- validations
- attr_protected / attr_accessible
- named_scope
- associations
 - caveat - some duplication

- **Modules**
- **Gems**
- **Rails 3 engines / mountable apps**

Communicating across BCs

Loose Coupling



EventSeries

TicketReservation

User

EventInventory

Purchase history

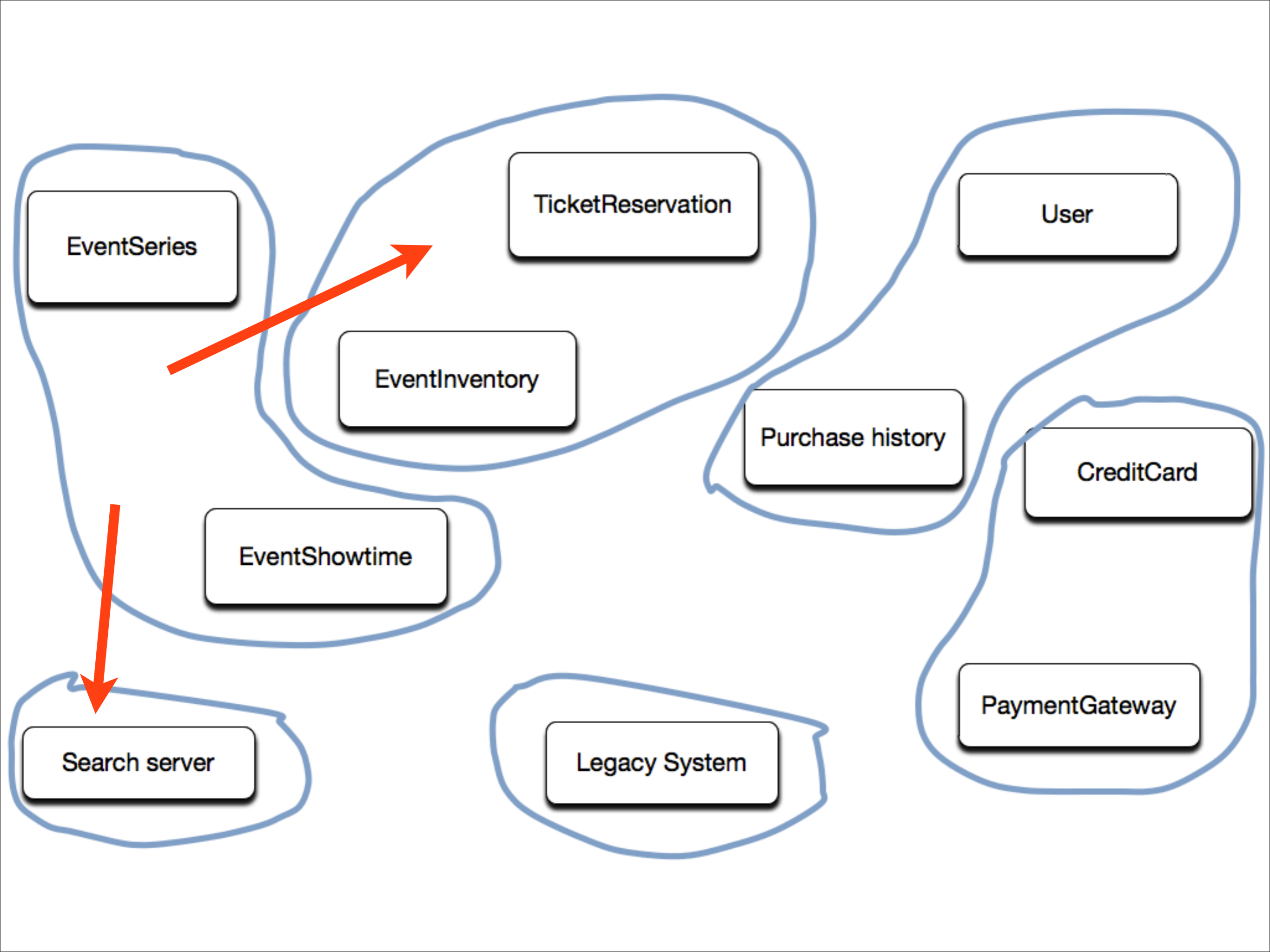
CreditCard

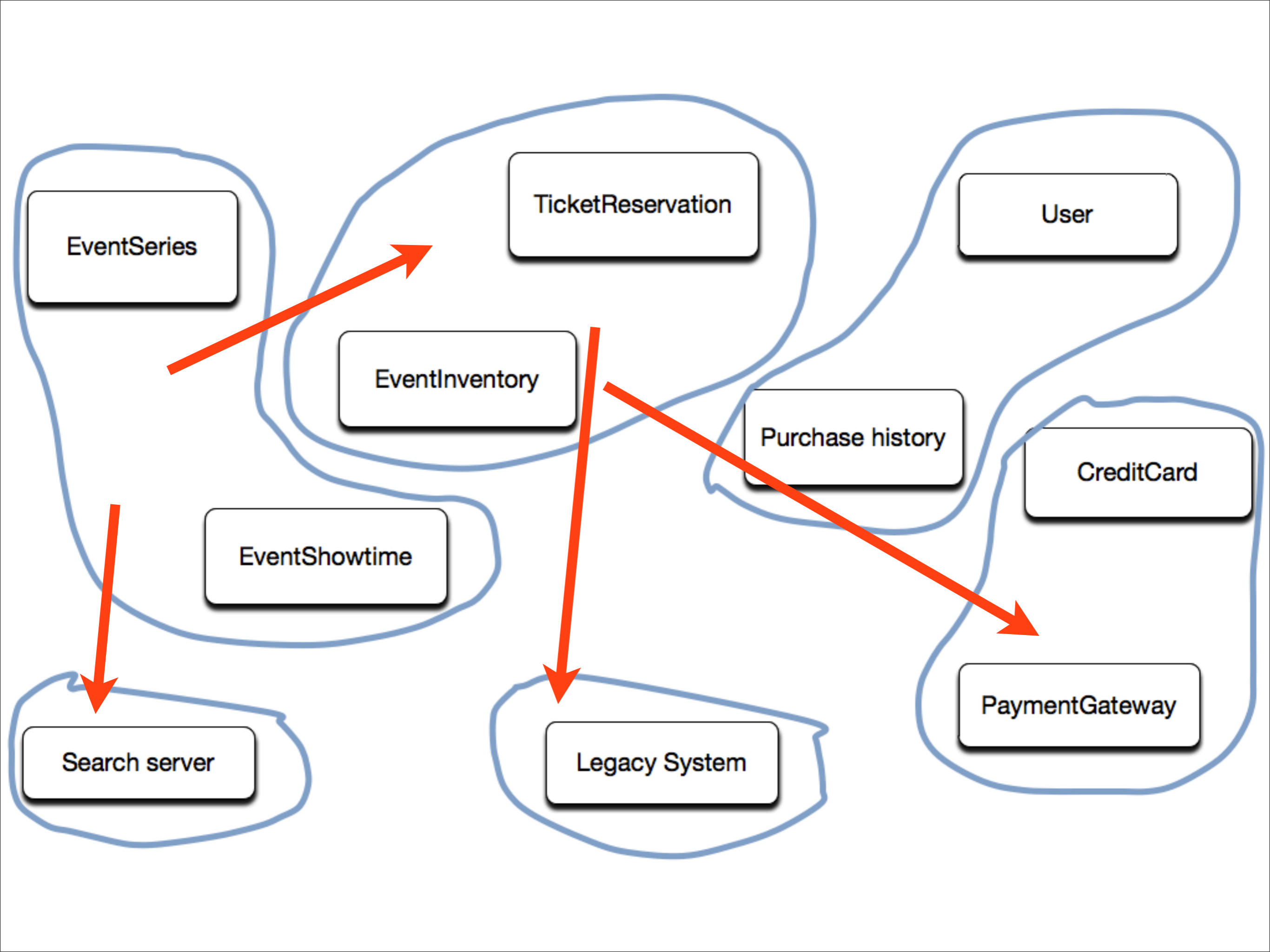
EventShowtime

Search server

Legacy System

PaymentGateway





Well-defined interfaces

Events

```
module ContentManagement
  class EventShowtime < ActiveRecord::Base
    after_create :publish_created_event

    def publish_created_event
      Events.publish :event_showtime_created, attributes
    end
  end
end
```

```
module Purchasing
  class Order < ActiveRecord::Base
    after_create :publish_order_placed_event

    def created_event
      Events.publish :order_placed,
        :event_showtime_id => event.id,
        :quantity => quantity,
        :credit_card => credit_card,
        :total_amount => total_amount
    end
  end
end
```

```
module Inventory
  class EventHandler
    def event_showtime_created(attrs)
      Inventory.create! :event_showtime_id => attrs[:id]
    end
  end
end
```

```
module Inventory
  class EventHandler
    def event_showtime_created(attrs)
      Inventory.create! :event_showtime_id => attrs[:id]
    end

    def order_placed(options)
      sid = options.fetch :event_showtime_id
      quantity = options.fetch :quantity
      inv = Inventory.find_by_event_showtime_id! sid

      unless inv(:available, quantity)
        Events.publish_errors
          :event_showtime_id => sid, :errors => inv
      end
    end
  end
end
end
end
```

```
module Payment
  class EventHandler
    def order_placed(options)
      credit_card = options.fetch :credit_card
      total = options.fetch :total_amount
      txnid = PaymentGateway.auth_capture(
        credit_card, total)
    end
  end
end
```

```
module Payment
  class EventHandler
    def order_placed(options)
      credit_card = options.fetch :credit_card
      total = options.fetch :total_amount
      @txn_id = PaymentGateway.auth_capture(
        credit_card, total)
    end

    def rollback
      PaymentGateway.refund @txn_id
    end
  end
end
end
```

```
module Events
  def self.clear_handlers
    @listeners = Hash.new { |hash, missing_key| hash[missing_key] = [] }
  end

  def self.register(event_type, handler)
    @listeners[event_type] << handler
  end

  def self.publish(event_type, *args, &block)
    @listeners[event_type].each { |listener|
      listener.send(event_type, *args, &block) }
  end

  def self.rollback
    @listeners.values.each &:rollback
  end
end
```

```
class ApplicationController
  around_filter :dispatch_events

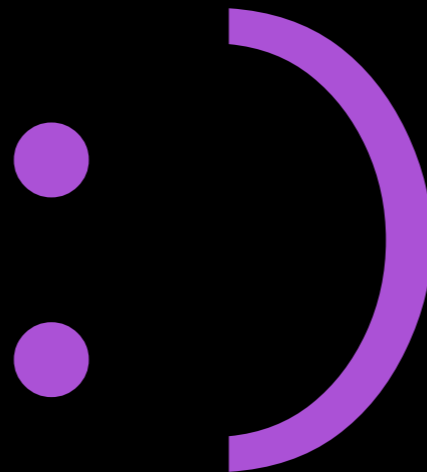
  def dispatch_events
    Events.clear_handlers
    Events.register :event_showtime_created, Inventory::EventHandler.new
    Events.register :order_placed, Inventory::EventHandler.new
    Events.register :order_placed, Payment::EventHandler.new

    begin
      yield
    rescue => e
      Events.rollback
      raise e
    end
  end
end
```

```
module Search
  class EventHandler
    def event_showtime_created(attributes)
      SearchServer.create_indexed_record attributes
    end
  end
end
```

```
module Search
  class EventHandler
    def event_showtime_created(attributes)
      begin
        SearchServer.create_indexed_record attributes
      rescue => e
        # Just log it and keep going. We'll be okay
        HoptoadLogger.log e
      end
    end
  end
end
end
```

```
module Search
  class EventHandler
    def event_showtime_created(attributes)
      begin
        SearchServer.create_indexed_record attributes
      rescue => e
        # Just log it and keep going. We'll be okay
        HoptoadLogger.log e
      end
    end
  end
end
end
end
```



- Single process is easy
- Database transactions
- Message queues when distributed

Wrapping up

Evolve the Ubiquitous Language

Design along business contours

Invest in the core domain

THANK YOU

patmaddox.com

twitter.com/patmaddox

pat.maddox@gmail.com