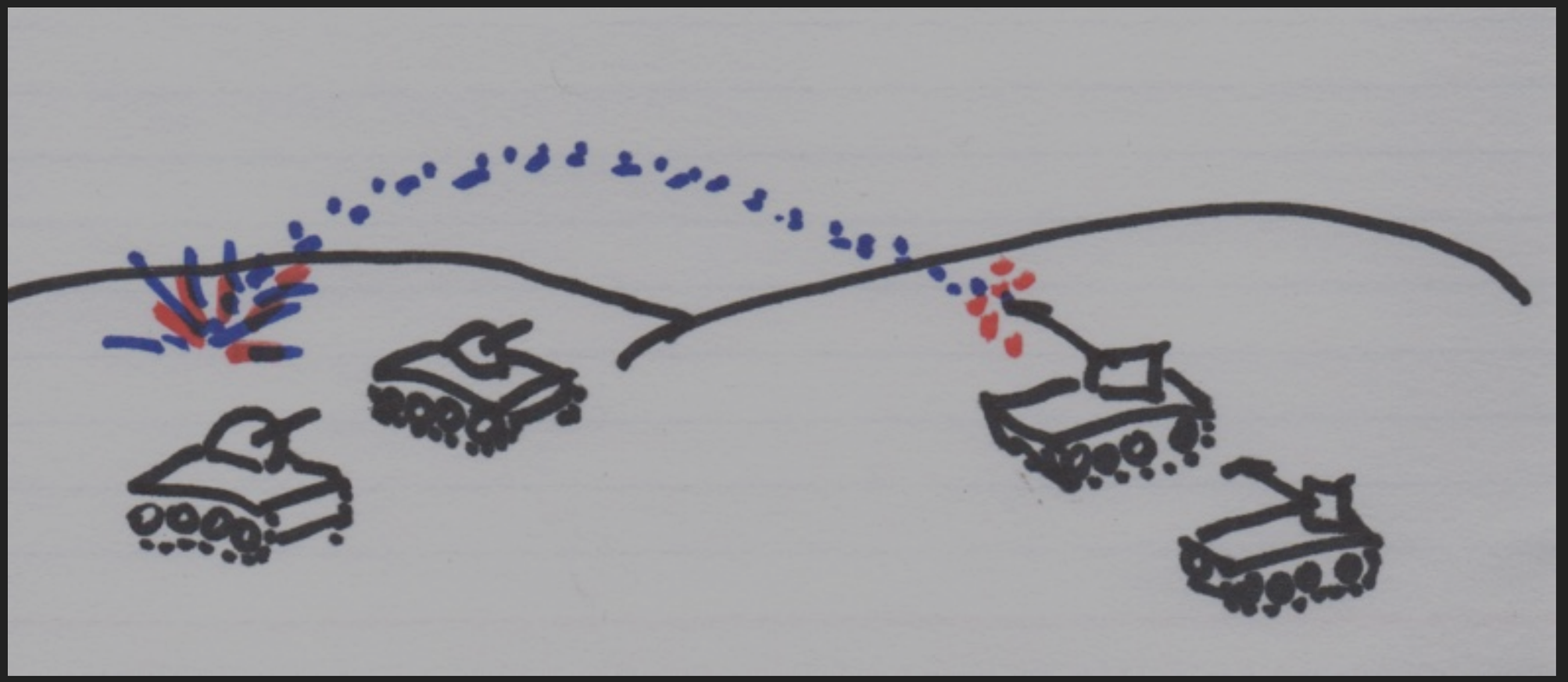


** no actual soldiers were harmed in the making of this presentation

MARK WINDHOLTZ - @WINDHOLTZ

WAR BETWEEN TOOLS & DESIGN

WHO DOES RAILS?

OVERVIEW

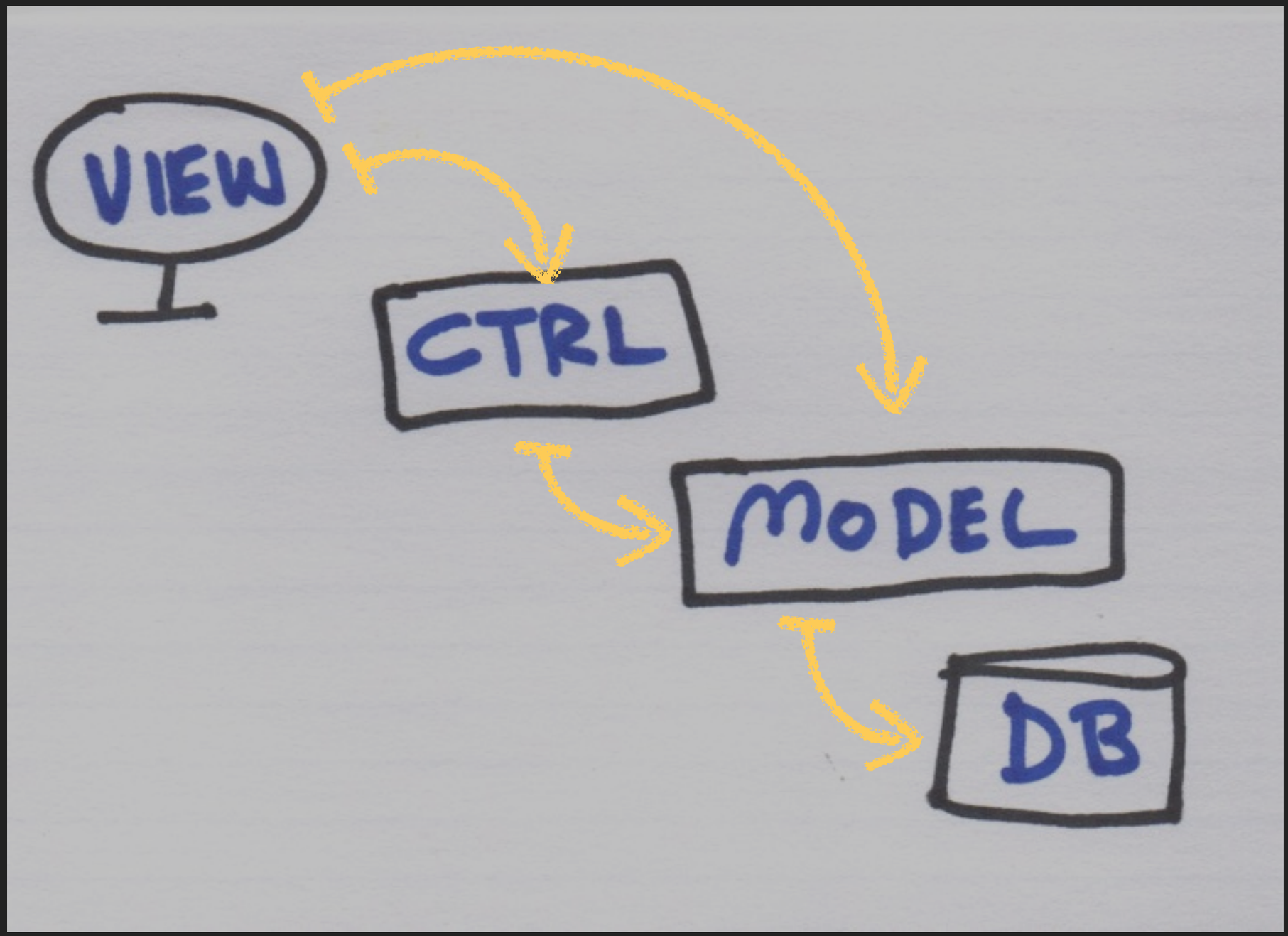
- Problem
  - No perfect solution
- Patterns
- Code Example
- A Potential Future Framework

REWRITE – RESEARCH

TOOLS ADD COUPLING

- Coupling and Cohesion
- Rails advocates a “Golden Path”
  - Standards for quick development
  - Connects view to DB.
- Coupling is not unique to Rails

RAILS DEPENDENCIES



TOOLS VS DESIGN

- Tools
  - Use their “way” to build
  - Tool lock-in and Coupling
- OO Design
  - Separate the Buisness Domain
  - May write more code
  - Risk: Extra coding not paying off

COUPLING PROBLEMS

- Hard to Test
- Slow to Test
- Different Rates of Change
- Check in / Merge issues

RANGE OF APPROACHES

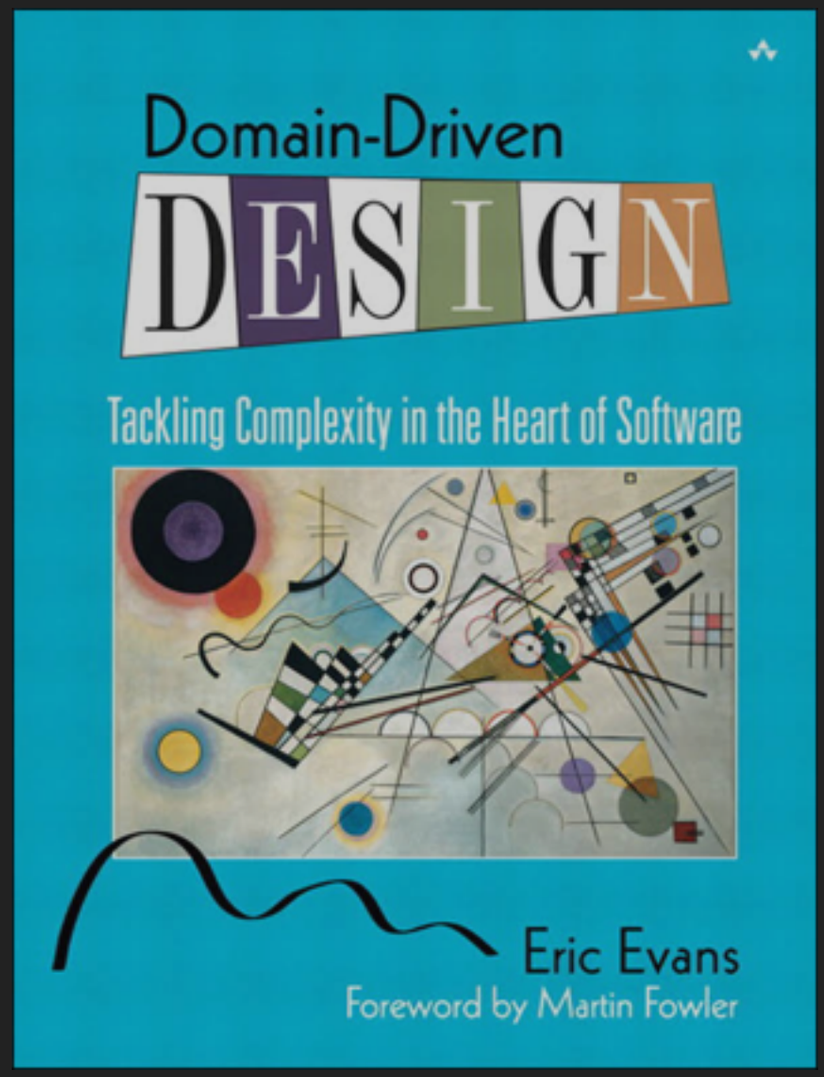
- Rails Scaffolding <—–> Hexagonal Architecture
- Short-Term Goal:
  - Something in between
  - Enable: Better OO, specifically DDD
- Long-Term Goal – DDD

"JUST ENOUGH" APPROACH

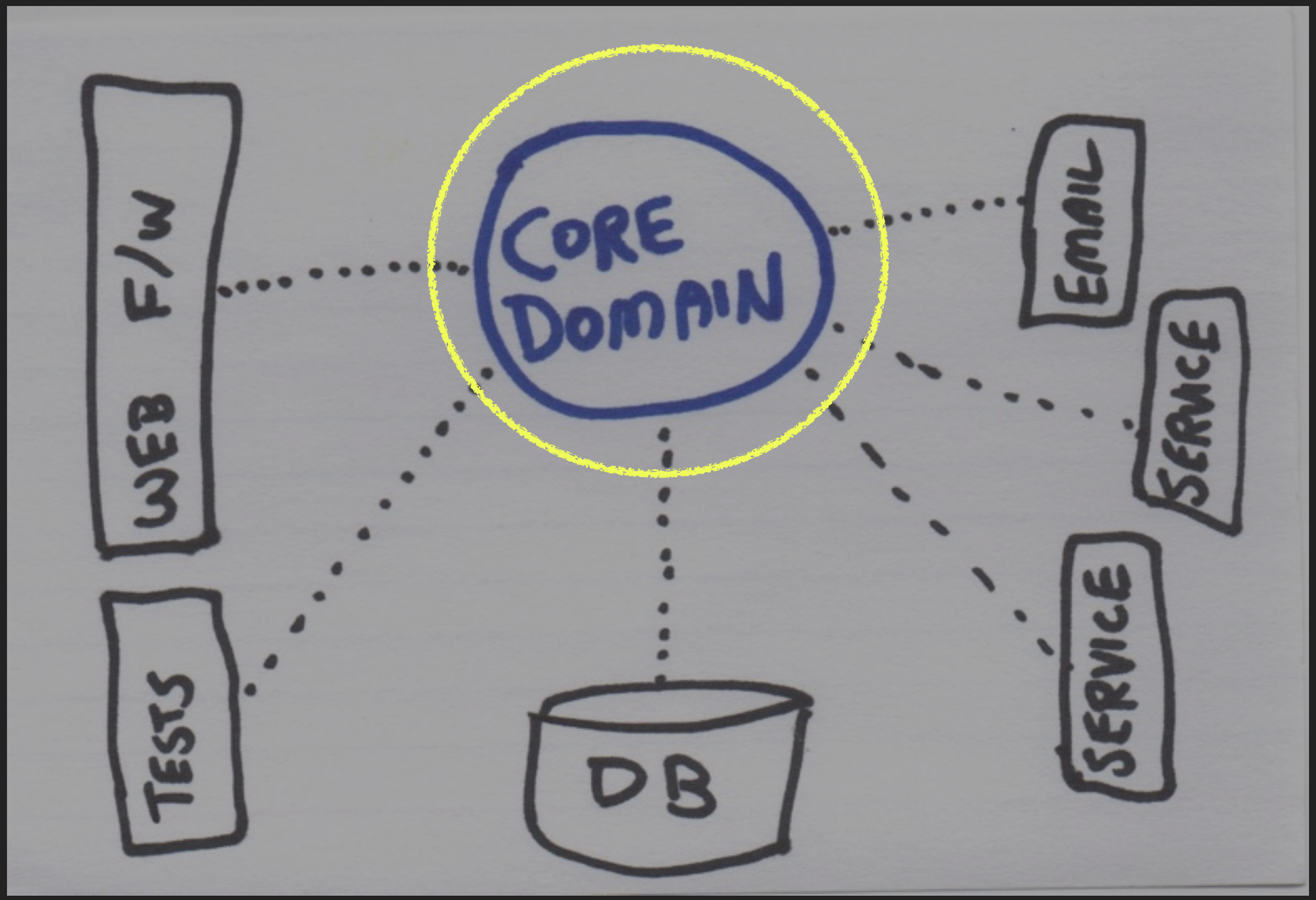
- Just Enough to Grow
  - but not too much to Slow you
- Nothing original
- Pattern (s)
- Large app ideas in a small demo

DOMAIN DRIVEN DESIGN

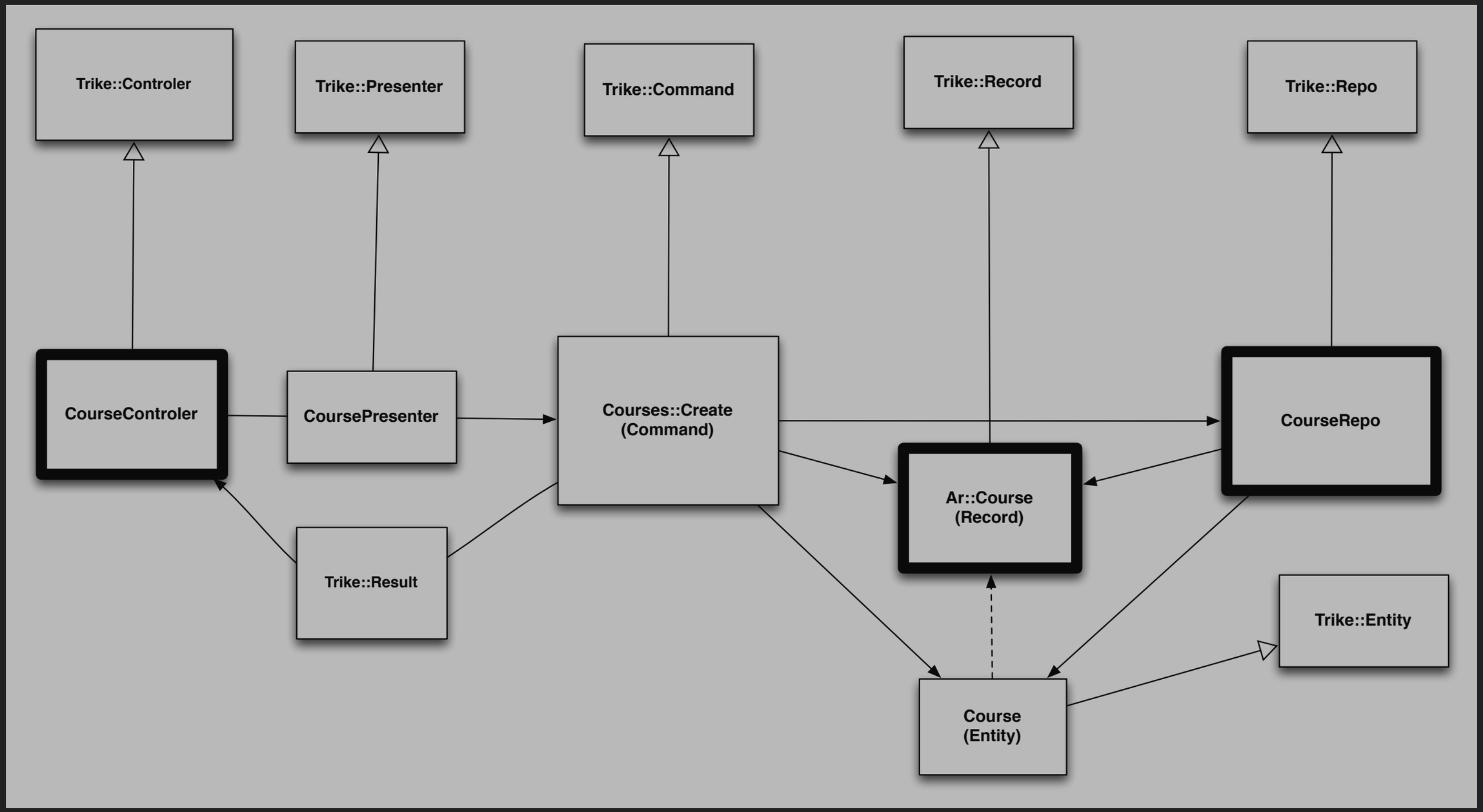
- Domain-driven design (DDD)
- 2004
- Patterns and Techniques
- Focus on the core domain



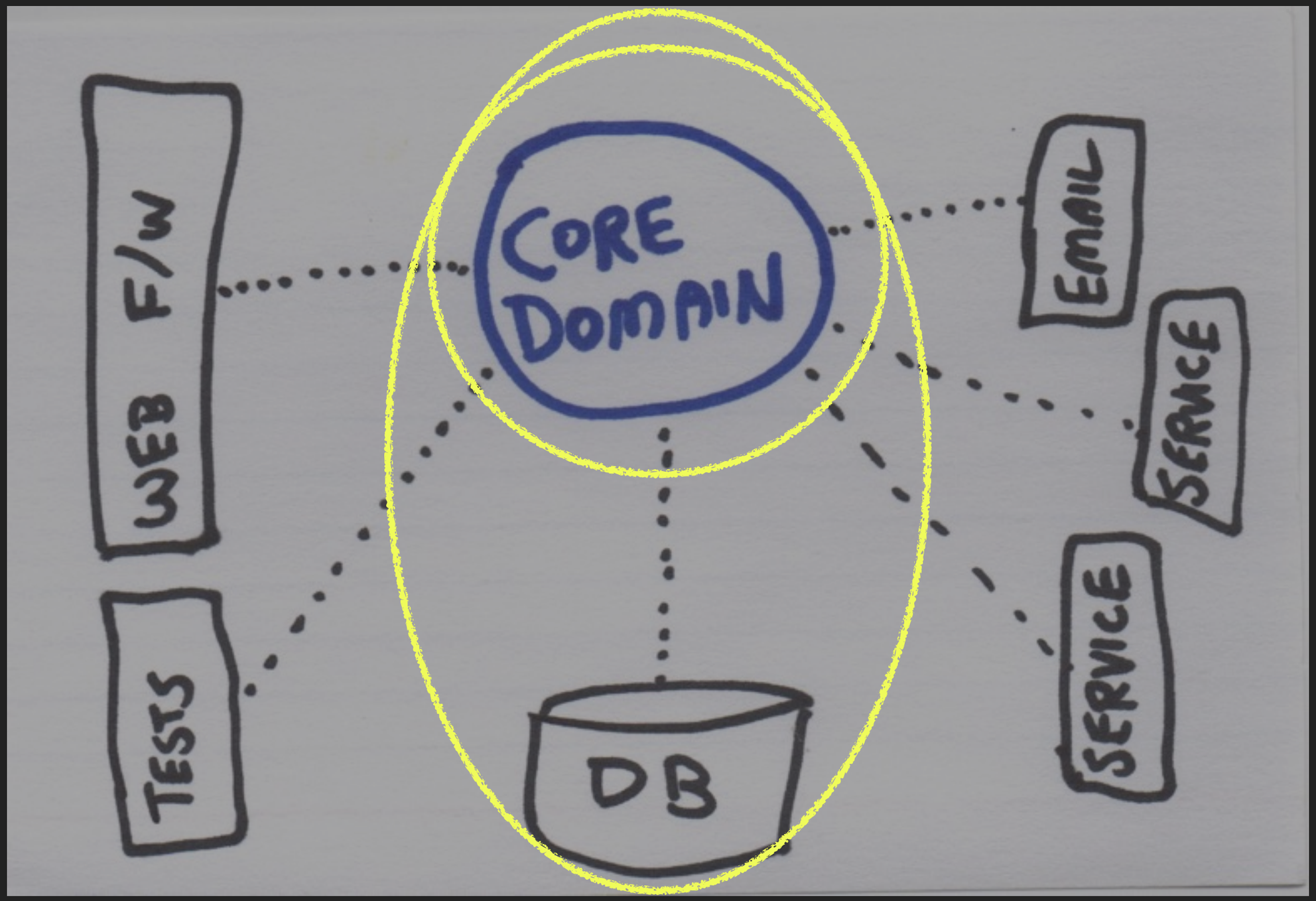
WHAT WE WANT



WHAT WE TRIED

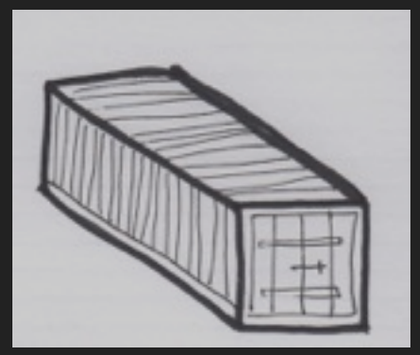


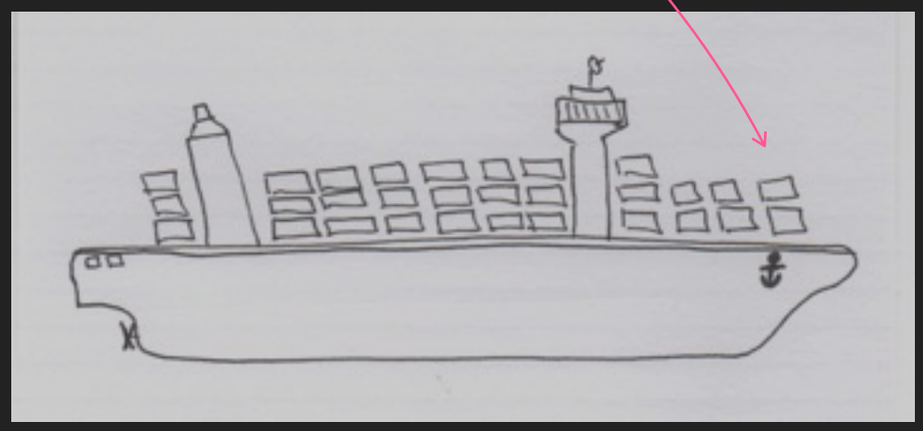
WHAT WE WANT (PART 2)



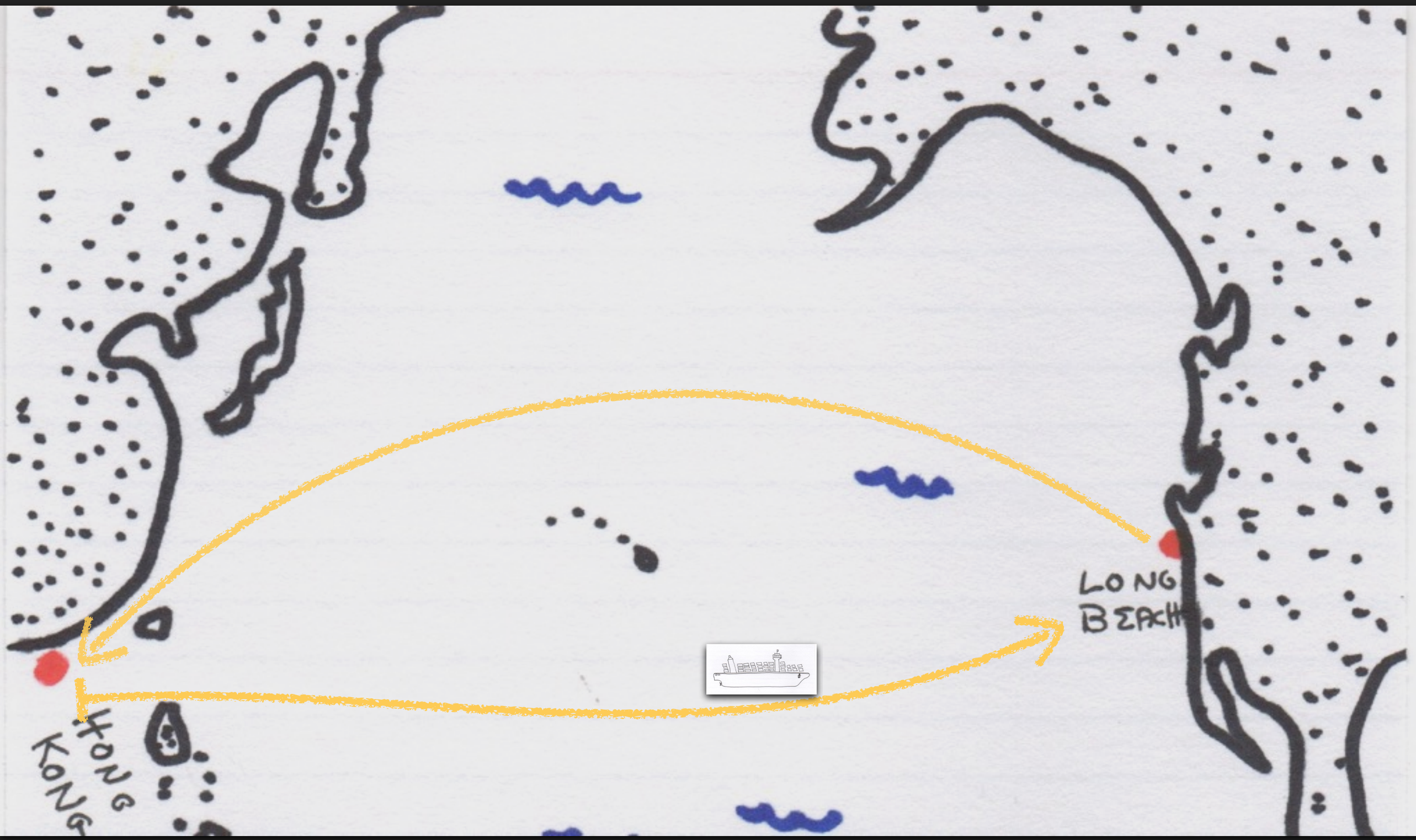
CARGO SHIPPING EXAMPLE

- Started with Rails scaffolding of Cargo
- Refactored to Patterns
- https://github.com/mwindholtz/ddd_cargo_on_rails





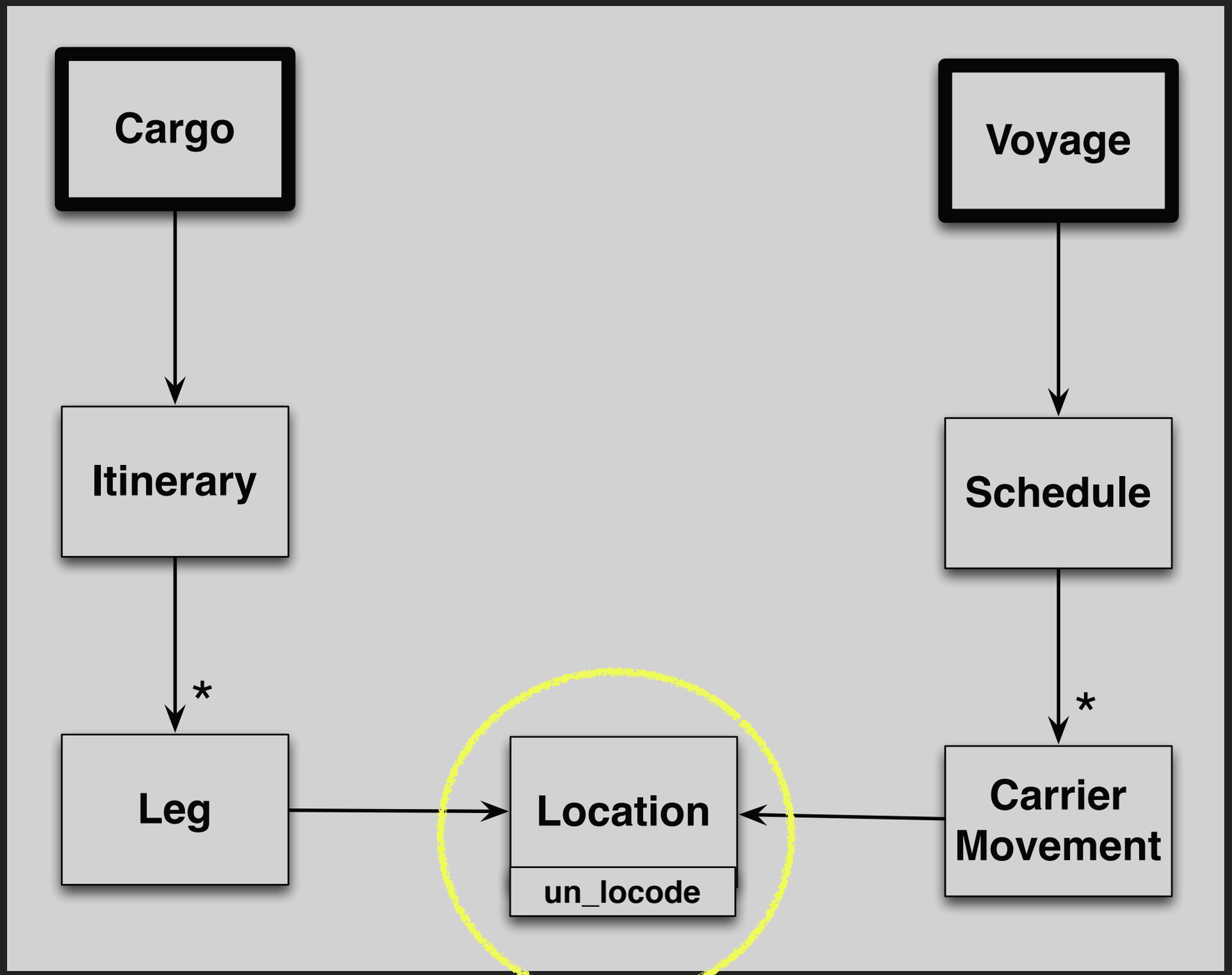
DEMO CODE



SCENARIO #1

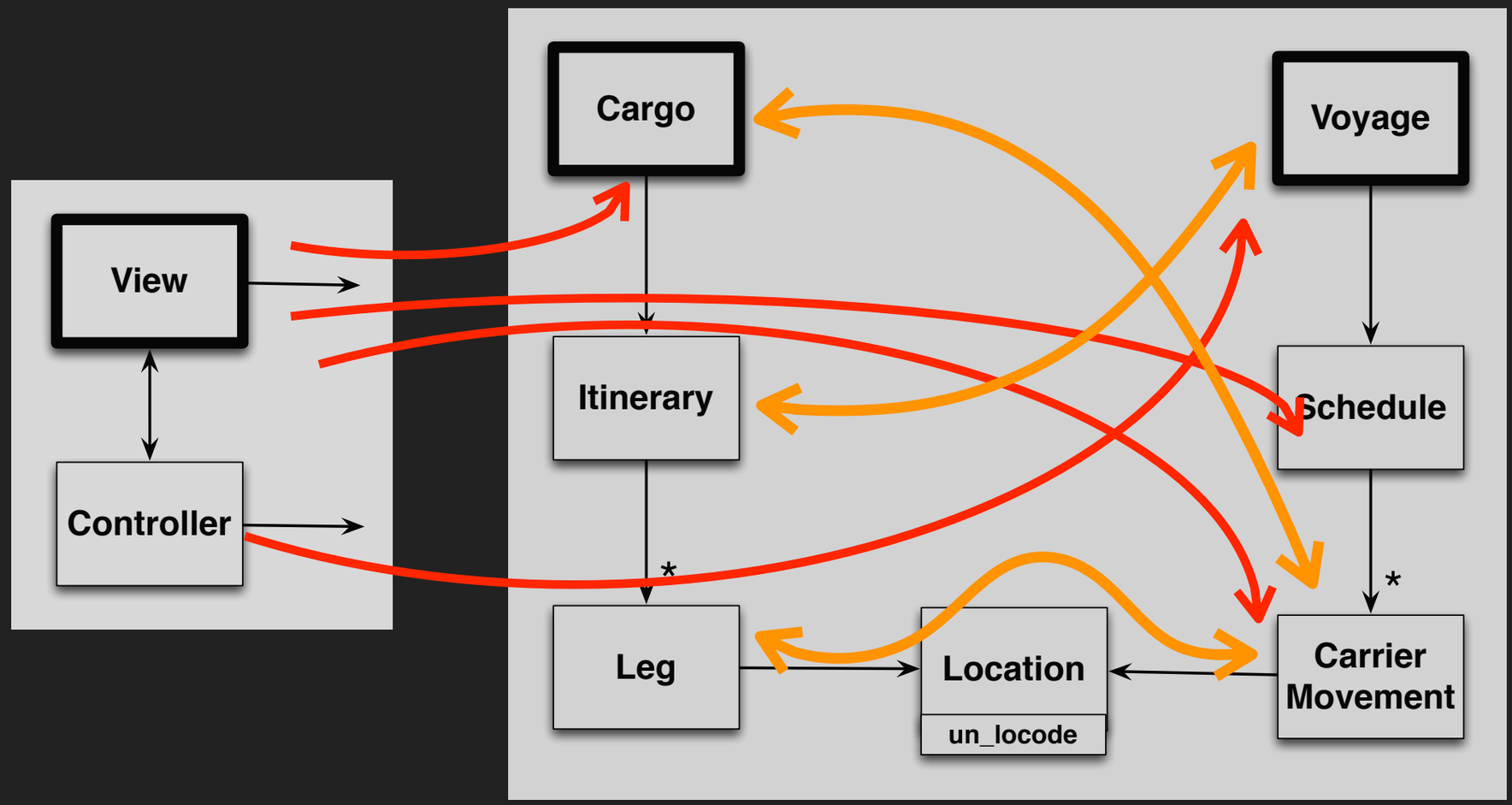
- Customer book a Cargo, from Hong Kong to Long Beach
  - Create a Cargo
  - Generate an Itinerary

CORE MODEL

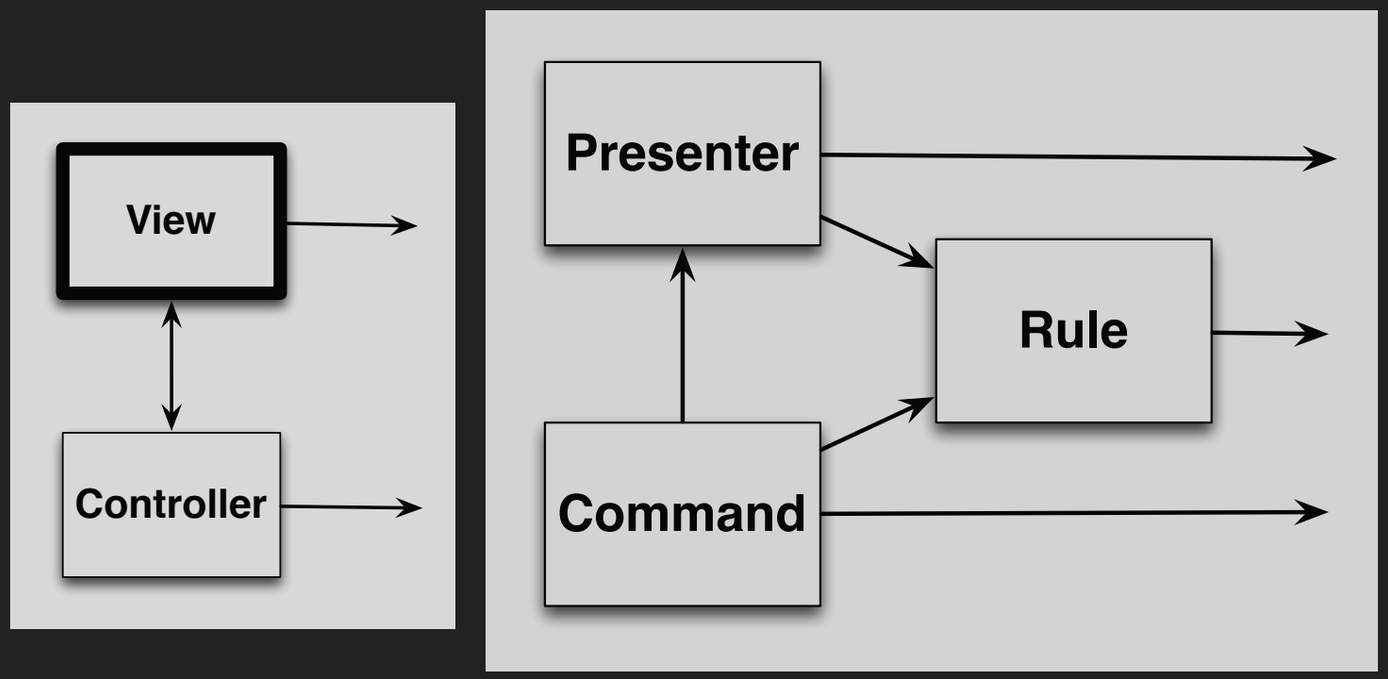

Cargo
Itinerary
*
Leg
Location
un_locode
Voyage
Schedule
*
Carrier Movement

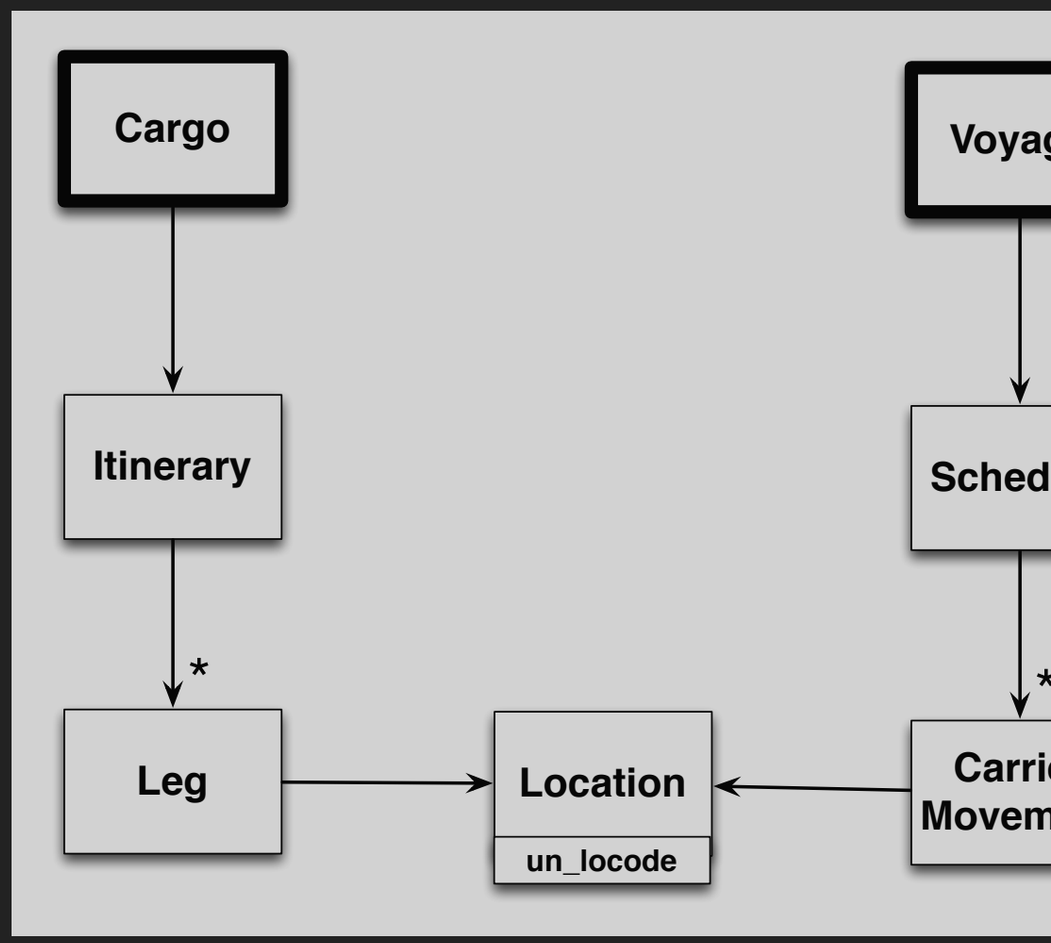
A LOOK AT EXAMPLE APP

GOLDEN PATH APPROACH



APPLICATION LAYER



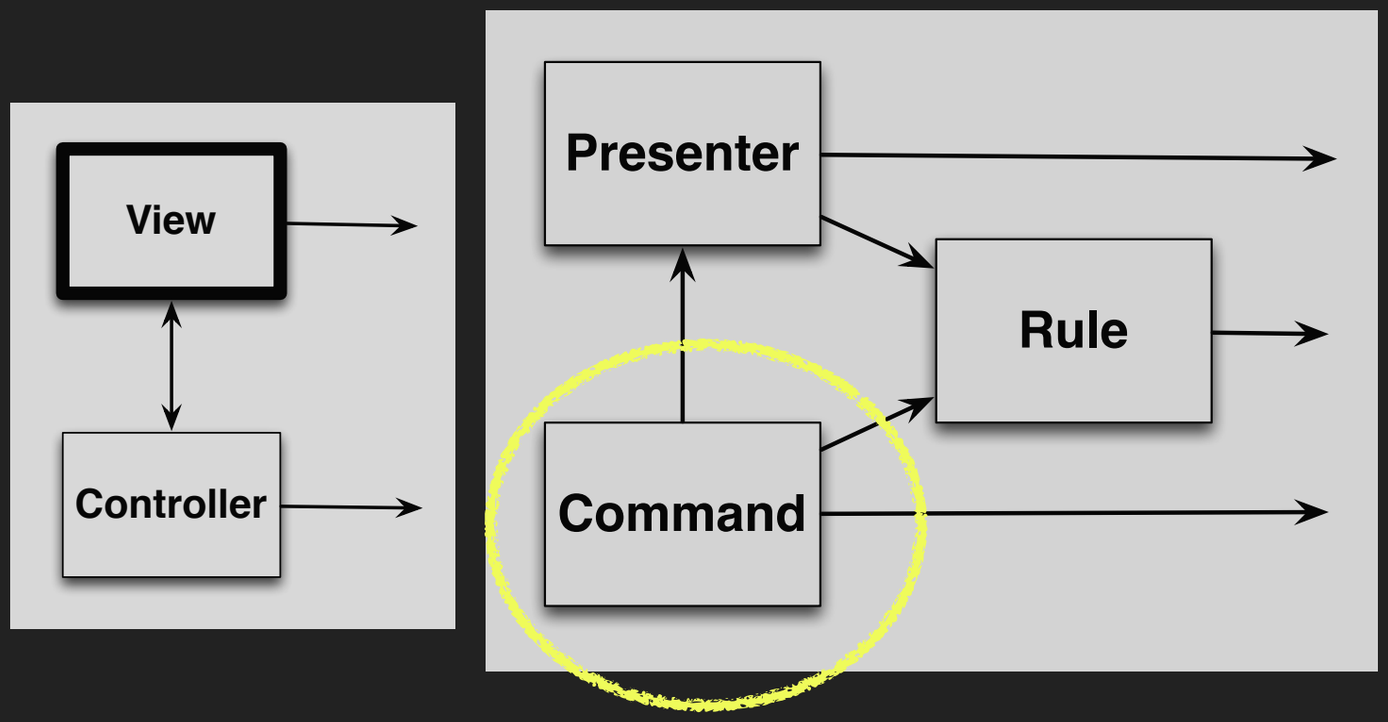


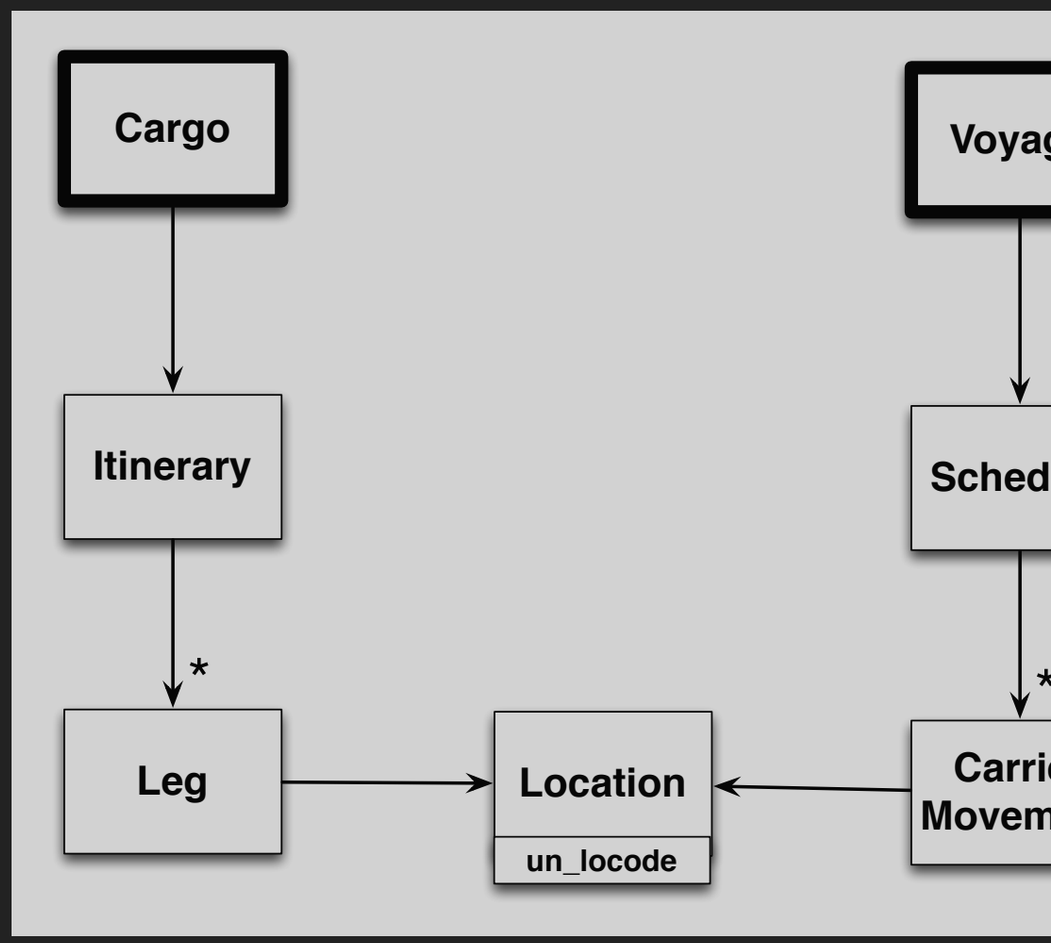
"All problems in computer science can be solved by another level of indirection except … the problem of too many indirections." — David Wheeler

APPLICATION LAYER PATTERNS

- Command
- Presenter
- Rule

APPLICATION LAYER – COMMAND





COMMAND

- Problem
  - Some Logic needs multiple models
  - Some Logic is specific to one request.
- Solution
  - Move the responsibility in a single class
  - Standard REST commands can be easily abstracted
- AKA: Transaction Script, PEAA

TWO MAIN COMMAND TYPES

- *Domain Facade*
  - Simple pass thru to the Core Domain
  - Standard REST
- *Operation Script*
  - Implements Application Logic
  - Uses the Core Domain

COMMAND INTERFACE

- # create
- # call
- return a Presenter

FIND LOCATION COMMAND

```ruby
class LocationsController < ApplicationController
  def show
    @prez = Cmds::FindLocation.new(params[:id]).call
  end
end
```

```ruby
class Cmds::FindLocation
  attr_reader :object_id

  def initialize(object_id)
    @object_id = object_id
  end

  def call
    object = Location.find(object_id)
    LocationPresenter.new(object)
  end
end
```

GENERIC FIND COMMAND

```ruby
class RestCmdsController < ApplicationController
  def show
    @prez = Cmds::Find.new(controlled_klass, params[:id]).call
  end
end
```

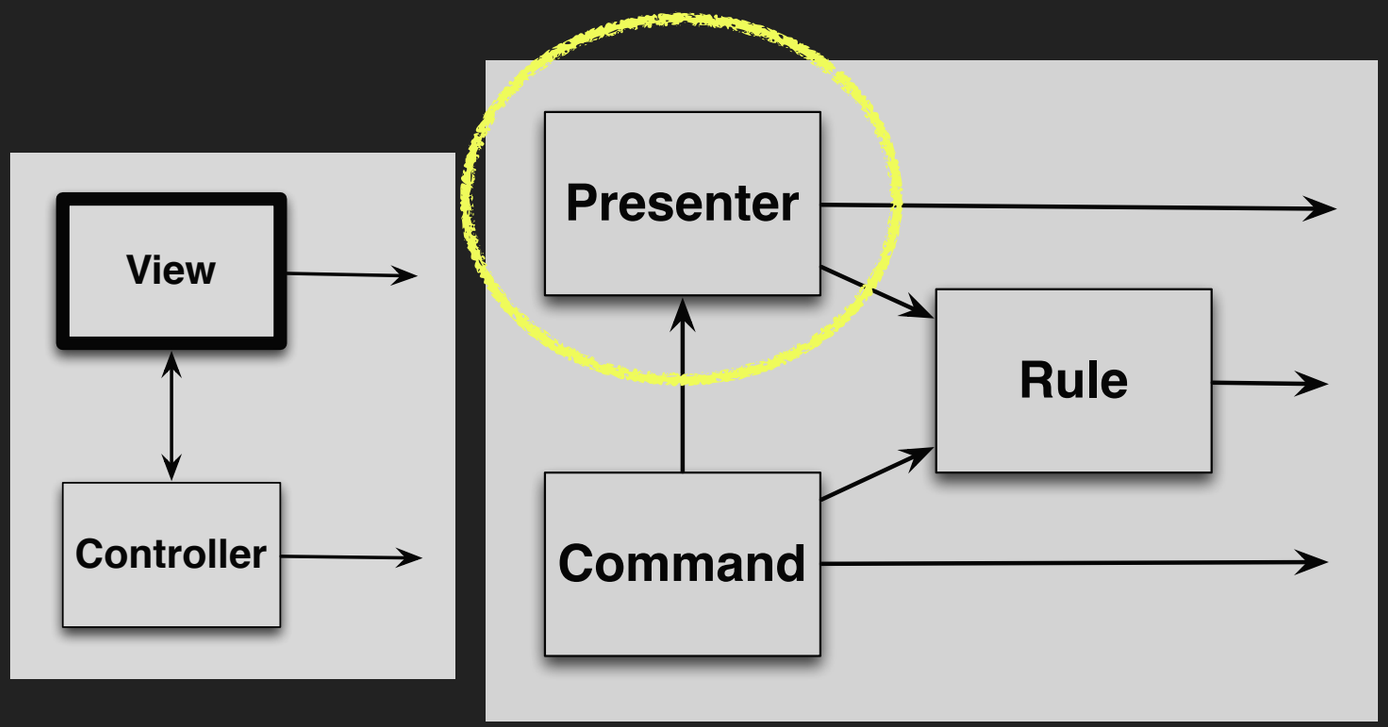
```ruby
class Cmds::Find < Cmds::GenericBase
  attr_reader :object_id

  def initialize(klass, object_id)
    @object_id = object_id ; super(klass)
  end

  def call
    object = klass.find(object_id)
    presenter_klass.new(object)
  end
end
```

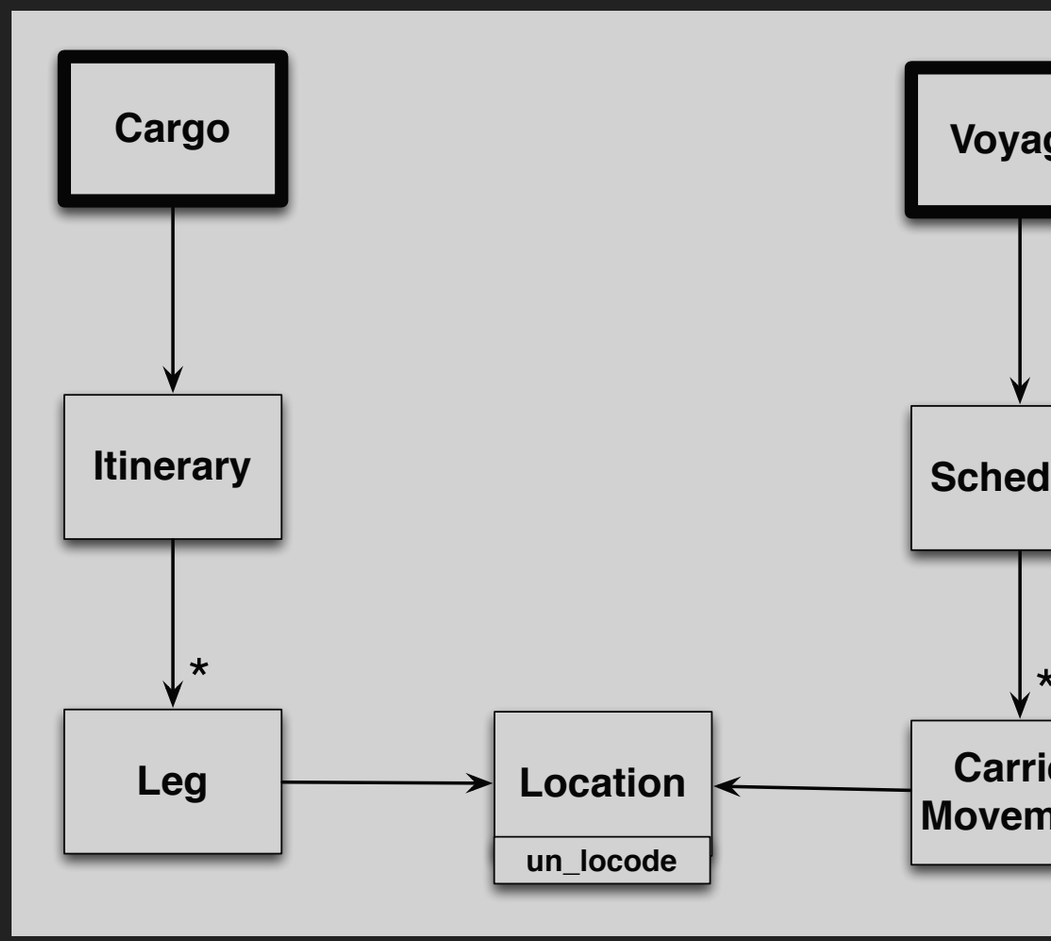
LOCATION CONTROLLER

OPERATION SCRIPT - GENERATE ITINERARY

```
module Cmds
  class GenerateItinerary < Cmds::Base

    def initialize(cargo_id)
      @cargo_id = cargo_id
      super()
    end

    def call
      routing_service = RoutingGateway.service.new
      cargo = Cargo.find(cargo_id)
      routing_service.itinerary( cargo )
    end

    protected
      attr_reader :cargo_id
  end
end
```

APPLICATION LAYER





PRESENTER

- Problem
  - Many variables in View add Coupling
  - Often Needs links and paths from Rails
- Solution
  - Only return 1- object to the view
  - Encapsulates the data needed by the view
  - Pass in a view_context for link and path generation
  - AKA, Two-Step View, PEAA

LOCATION PRESENTER – EXAMPLE – DEFINITION (1 / 2)

```ruby
class LocationPresenter

  ATTRS =  [:code, :name]
  attr_reader *ATTRS

  def initialize(object=nil)
    @object = object || Location.new
    unless @object.new_record?
      @code =  @object.code
      @name =  @object.name
    end
  end

  def errors; @object.errors
  end

  def values
    ATTRS.map{|each| self.public_send(each) }
  end
```

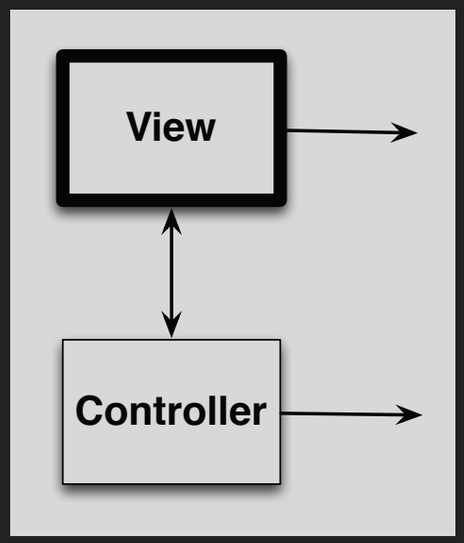
LOCATION PRESENTER – EXAMPLE – DEFINITION (2 / 2)

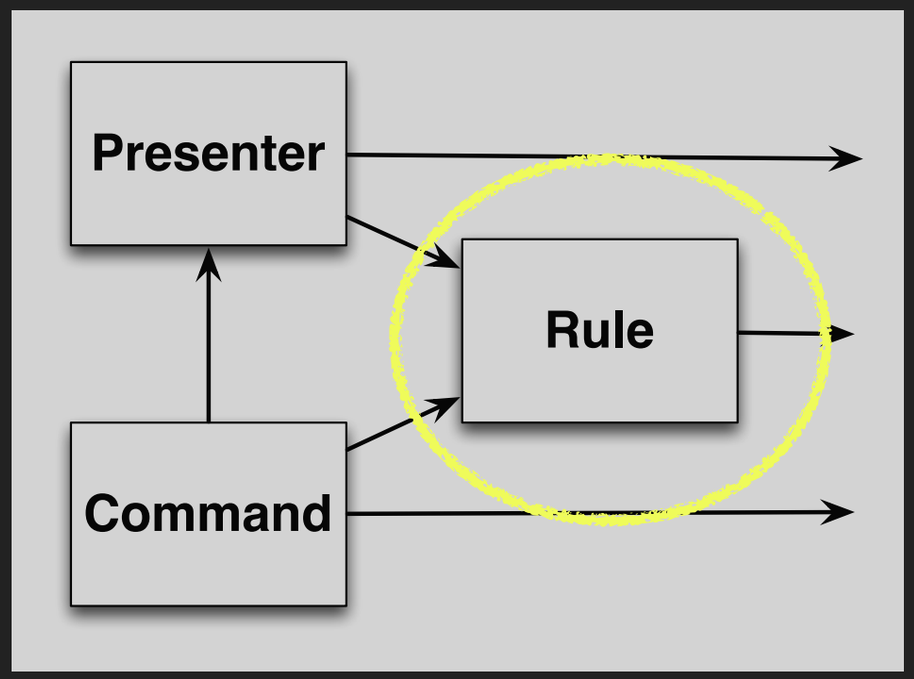
```ruby
  # Links ..........

  def link_to_show
    view_context.link_to('Show', @object)
  end

  def link_to_destroy
    view_context.link_to('Destroy', @object, method: :delete,
                                 data: { confirm: 'Are you sure?' })
  end

end
```

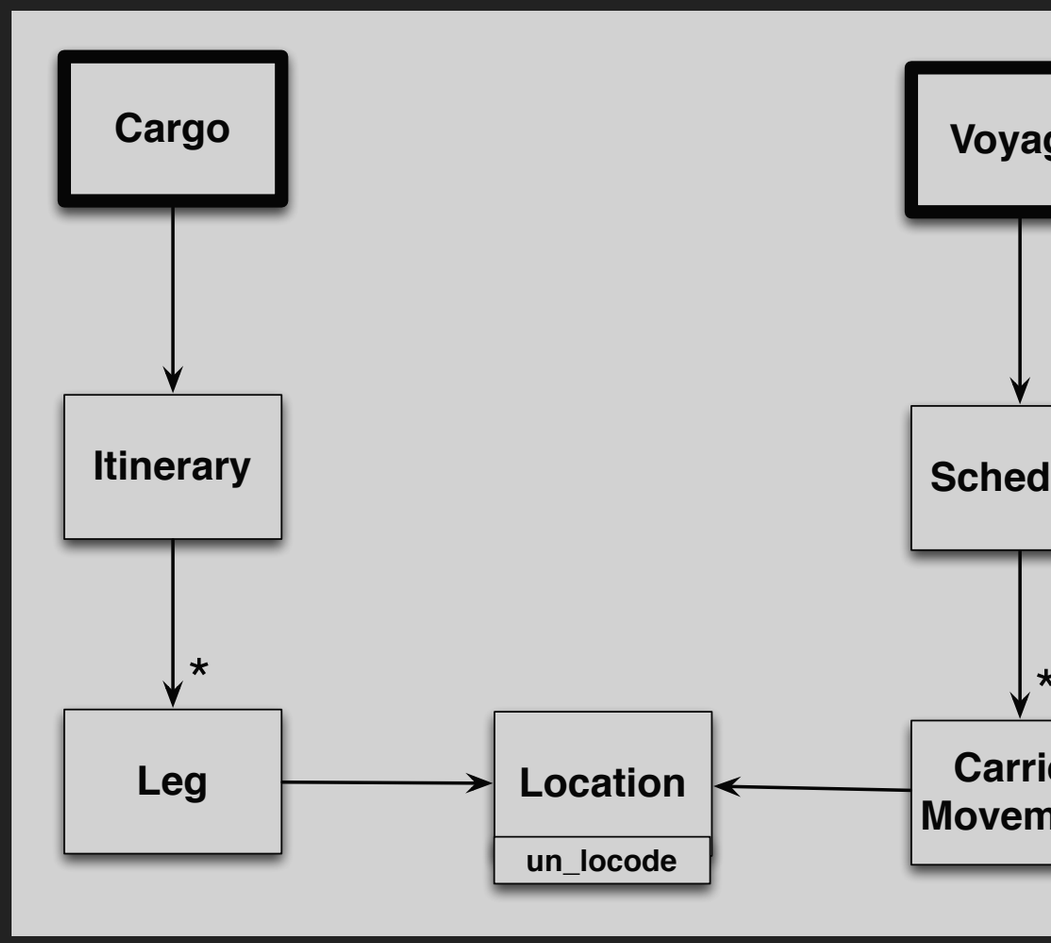
LOCATION PRESENTER - EXAMPLE - IN VIEW

```
<p id="notice"><%= notice %></p>

<div> <%= prez.name %> : <%= prez.code %> </div>

<br/>
<br/>
<%= prez.link_to_edit %> |
<%= prez.link_to_index('Back') %>
```

APPLICATION LAYER – RULE







RULE

- Problem
  - Code for business rules are getting spread among multiple classes
- Solution
  - Provide a class with the single responsibility to define and evaluate the business rule

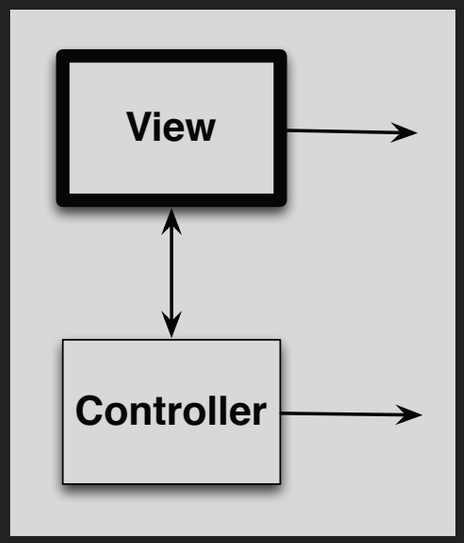
RULE - EXAMPLE

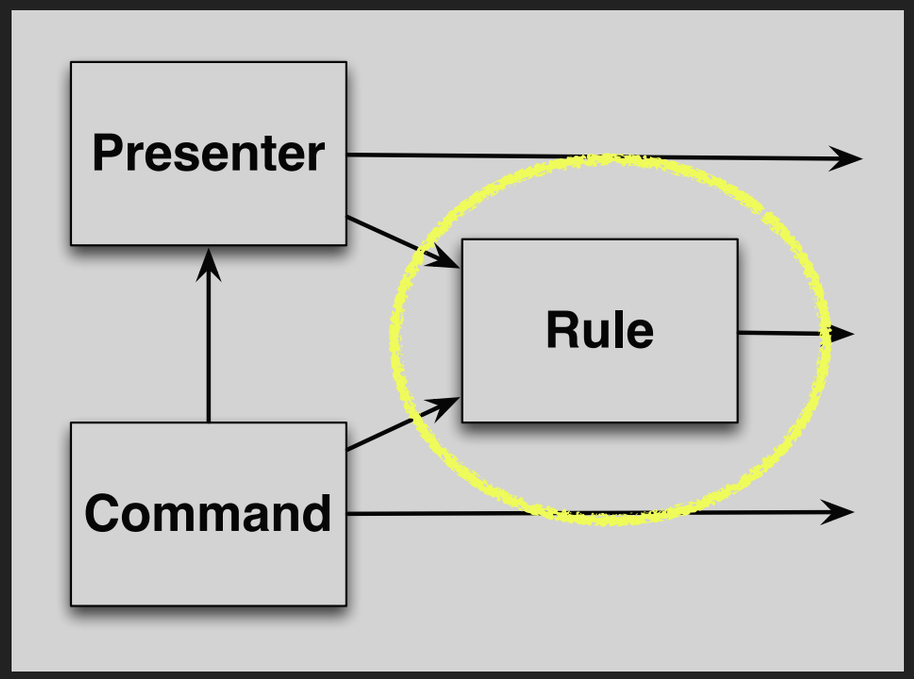
```
class ValidVoyageRule < RuleBase
  attr_reader :voyage

  def satisfied?
    is_a_loop?
  end

  private
    def is_a_loop?
      carrier_movements = voyage.schedule.carrier_movements
      home_port = carrier_movements.first.depart_location
      last_port = carrier_movements.last.arrival_location
      home_port == last_port
    end

end
```

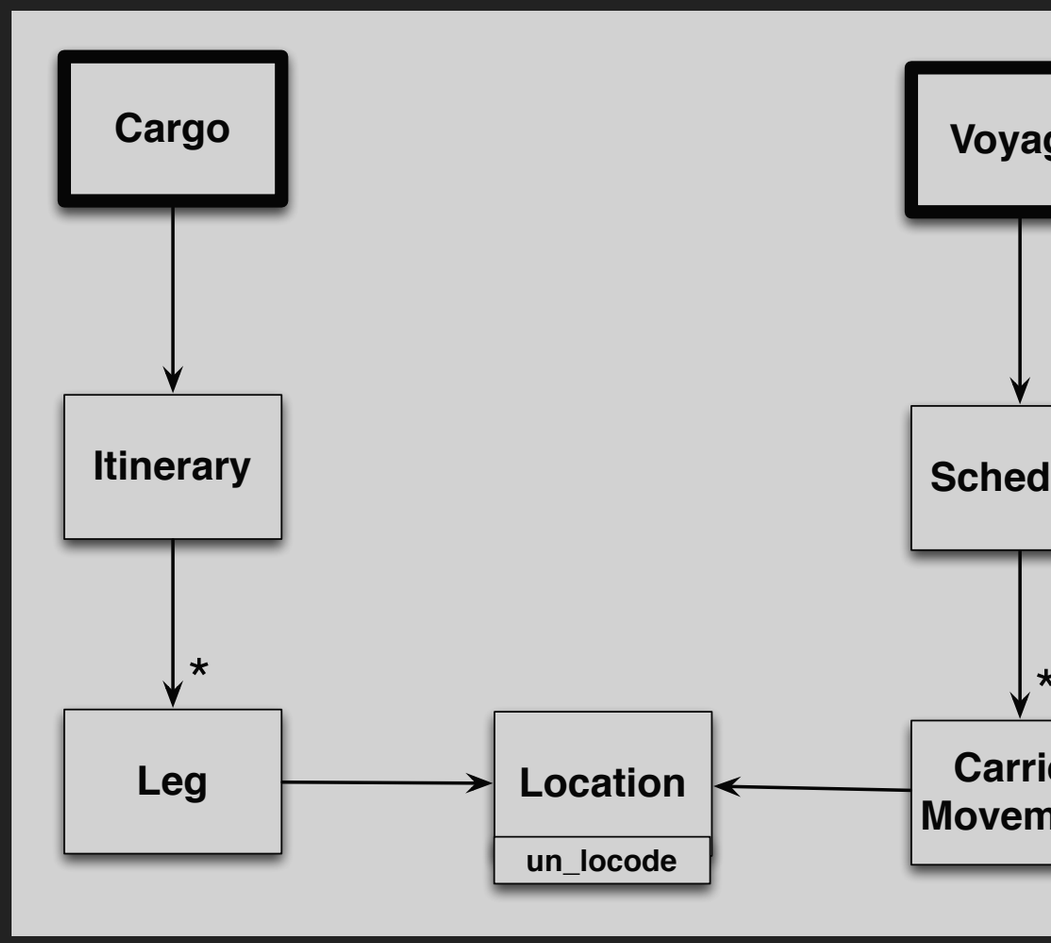
RULE - USAGE EXAMPLE

```ruby
def available?
  ValidVoyageRule.new(self).satisfied?
end
```

APPLICATION LAYER – RULE – AUDITOR







AUDITOR

- Problem
  - We want to know ***why*** a rule is satisfied (or not)
- Solution
  - Pass in an Auditor to record the reasons for the conditions in the rule
  - Pass a high performance NullAuditor when reasons are not recorded

AUDITOR - EXAMPLE

```
class ValidVoyageRule < RuleBase

  def satisfied?(auditor = Auditor::Null.new)

      if( ends_in_home_port? )
        auditor.add(self, 'Schedule is complete')
        return true
      else
        auditor.add(self, 'Does not end in home port')
        return false
      end
  end

end
```

RULE - USAGE EXAMPLE

```ruby
def why_available
  auditor = Auditor.new
  ValidVoyageRule.new(self).satisfied?(auditor)
  auditor.explain
end
```

A LOOK AT EXAMPLE APP

MORE PATTERNS

- Result
- Criteria

RESULT OBJECT

- Problem
  - Return values from Gateways varied and complex
  - Various result conditions, :ok, :error, :not_found, :out_of_stock, etc
- Solution
  - Return a Result object which encapsulates return data and provides status

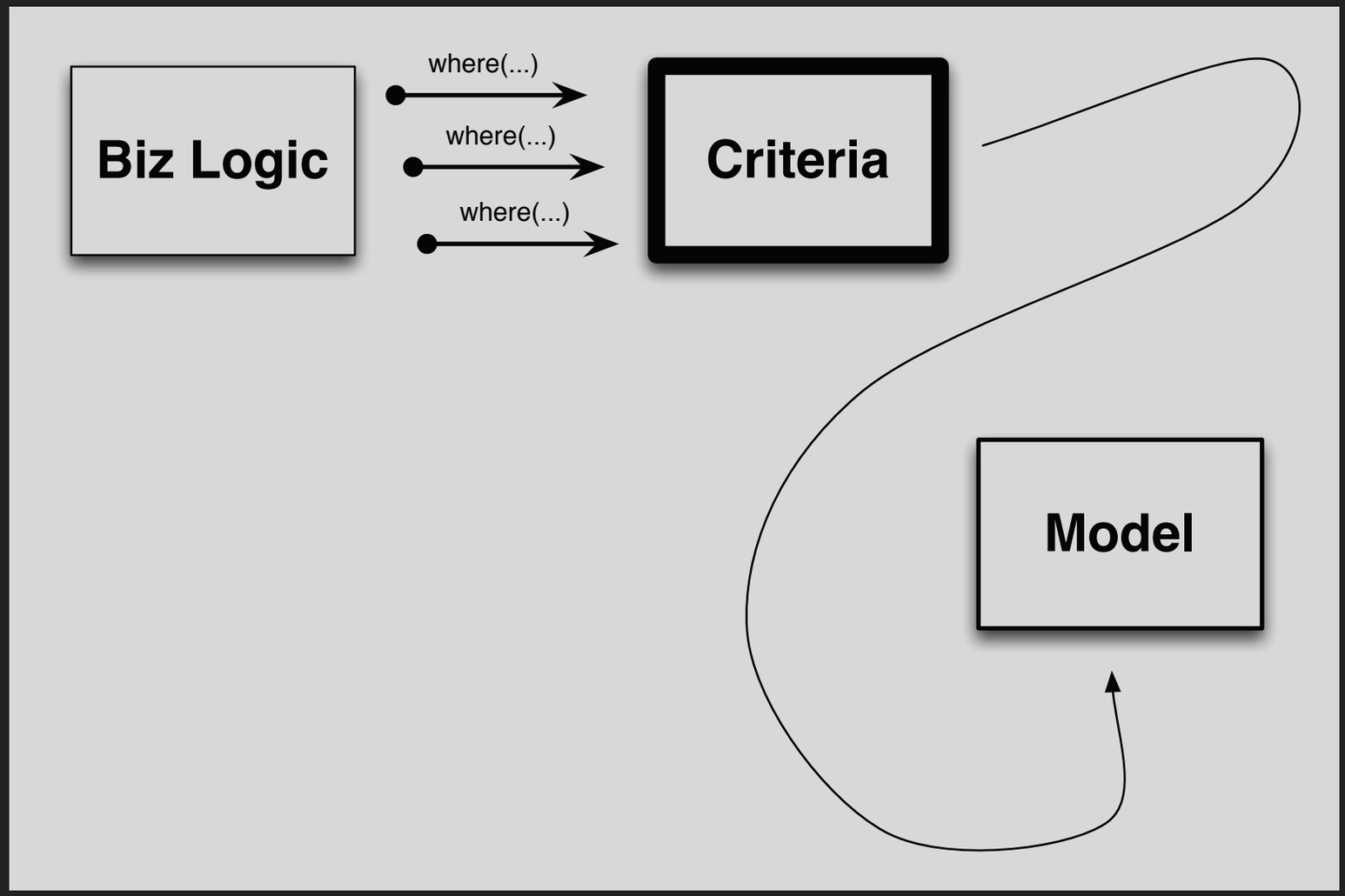
RESULT OBJECT – EXAMPLE

```
# ...
if success?
  Result.ok.add(itinerary: itinerary, message: "route found")
else
  Result.error.add(message: "no route could be found")
end
```

CRITERIA

- Problem
  - Rich (nearly Infinite) Active Record query interface
  - But we don’t want to couple A/R to queries, view or controller
- Solution
  - Cache the method sends in a Criteria
  - Send them to the actual A/R model later
  - Similar to *Specification* - DDD

CRITERIA



LOCATION CRITERIA - EXAMPLE

```ruby
class LocationsController < RestCmdsController

  # ...
  private

    def search_criteria
      criteria = Criteria.new
      criteria.where(code: params[:q])
      criteria
    end

end
```

OTHER TIPS

- Separate
  - User (for Domain)
  - from Credential (for Devise)

RODAKASE – ROM + RODA

- Ruby Object Mapper
  - Not coupled like ActiveRecord
- Rodakase is a lightweight web stack on top of Roda
- for web applications while decoupling your application code from the framework.
- https://github.com/solnic/rodakase

REFERENCES

- Object Design
  - SBPP - Smalltalk Best Practice Patterns, Kent Beck 1997
  - PEAA - Patterns of Enterprise Application Architecture, Martin Fowler, 2003
  - Domain-Driven Design: Tackling Complexity in the Heart of Software, Eric Evans. http://www.domainlanguage.com
  - http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod
  - DCI - http://www.artima.com/articles/dci_vision.html
- Getting Started with DDD when Surrounded by Legacy Systems
  - http://domainlanguage.com/ddd/strategy/GettingStartedWithDDDWhenSurroundedByLegacySystemsV1.pdf