Journey to DDD:

through the land of Dungeons, Dragons and Daemons

Piotr Brych @

"Campfire" by anndr on deviantart

Let's get to know each other

How many of you know what DDD means?

How many of you know what DDD means?

How many of you used DDD in a production app?

How many of you know what DDD means?

How many of you used DDD in a production app?

How many of you is constantly using it?

How I see myself

How DDD masters would see me

There!

"Magic: the Gathering - Arcane Endeavor" By JustynaGil on deviantart

Trappy- Attack Roll 18 + 4 = 22 : Hit
Trappy did 12 damage to Firkraag
PAUSED
Ogre- Auto-Paused: Round End

DDD

DDD

what does it stand for?

DDD
what does it stand for?

Dungeons & Dragons

fantasy tabletop role-playing game (RPG) first published in 1974

Daemon

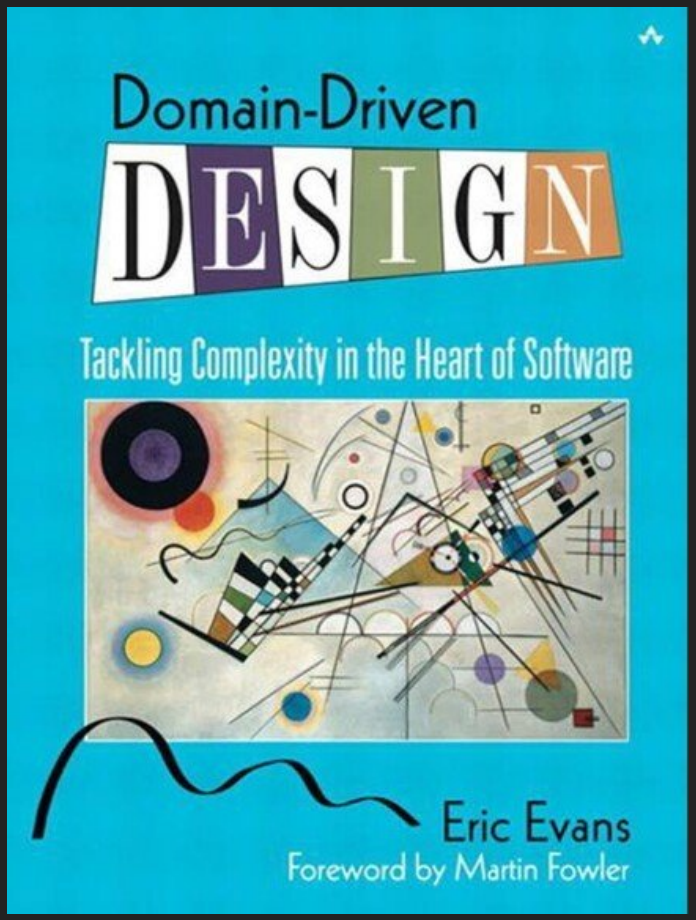
a computer program that runs as a background process


GOT YA!
memegenerator.net

Although technically it could be true...

Domain-Driven Design

term coined by Eric Evans, and used as a title of his book from 2003



Domain-Driven Design

way of designing software
systems with a special focus
on the **domain knowledge**
instead of technical aspects

exact implementation

~~exact implementation~~

~~exact implementation~~

building blocks

~~exact implementation~~

building blocks

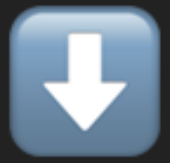
patterns to reuse

The Holy Grail

Source: https://onceuponatime.fandom.com/wiki/Holy_Grail

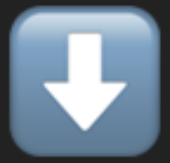
Domain Model

Domain Model

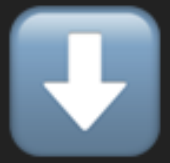


extra concepts

Domain Model



extra concepts



business breakthroughs

How to get to the Grail, you ask?

"Campfire" by anndr on deviantart

Wisdom

Global App Testing

domain

functional testing by using crowd of testers

We're going to build a new castle

Photo by Felix on Unsplash

Wait, but what's wrong with the old one?

Not scalable enough

New campaign: flexible testing

Act 1

Into the Darkness

"You must gather your party before venturing forth."


LET'S GET THIS
PARTY
ROLLING.
www.GRUMPYGEEKS.eu

Foundations

Foundations

- common understanding

Foundations

- common understanding

- divide the project into subparts

Foundations

- common understanding

- divide the project into subparts

- boundaries of MVP

First quest:

Big Picture Event Storming

Event Storming

- emerged around 2015, created by Alberto Brandolini
- workshop format for collaborative exploration of complex business domains
- book is not finished :)

"mines of Moria" by Gellihana-art on deviantart

Our weapons

The darkness of unknowns

Domain experts might also not know

Domain experts might also not know

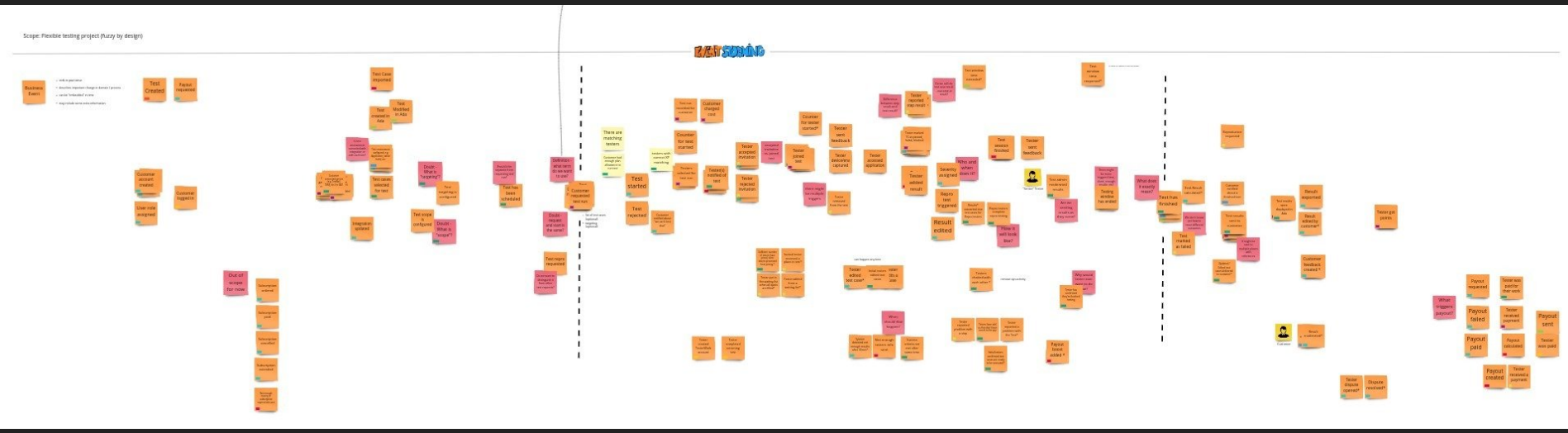
and that's ok

Be explicit about the unknowns - mark it with a hot spot note

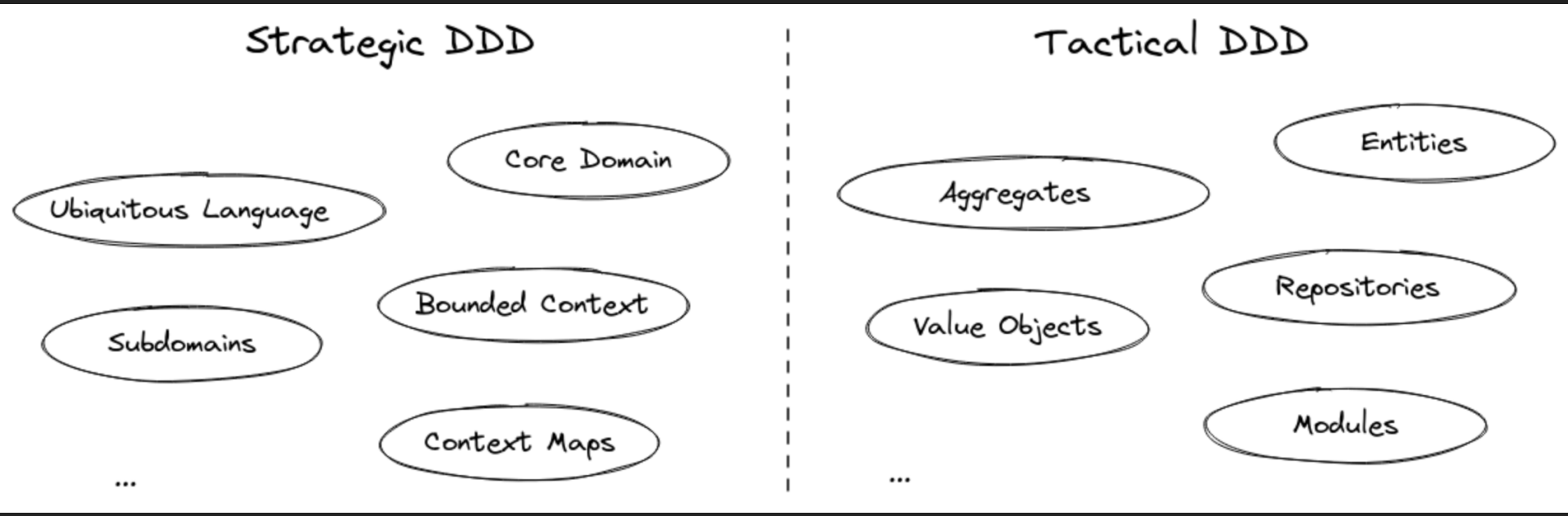
Be explicit about the unknowns - mark it with a hot spot note

and address it later, don't fall into the rabbit hole

First exploration

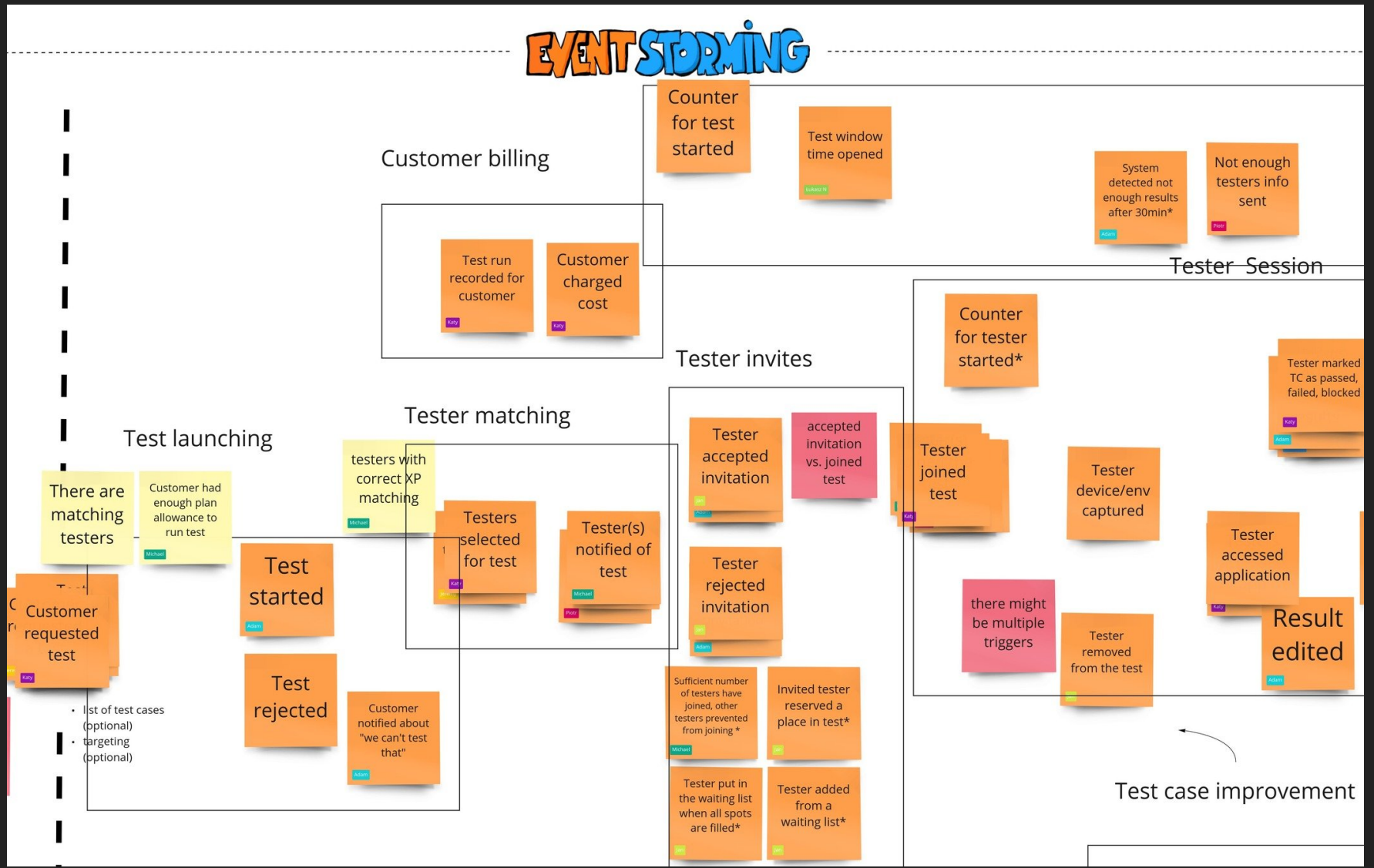


Strategic vs Tactical DDD

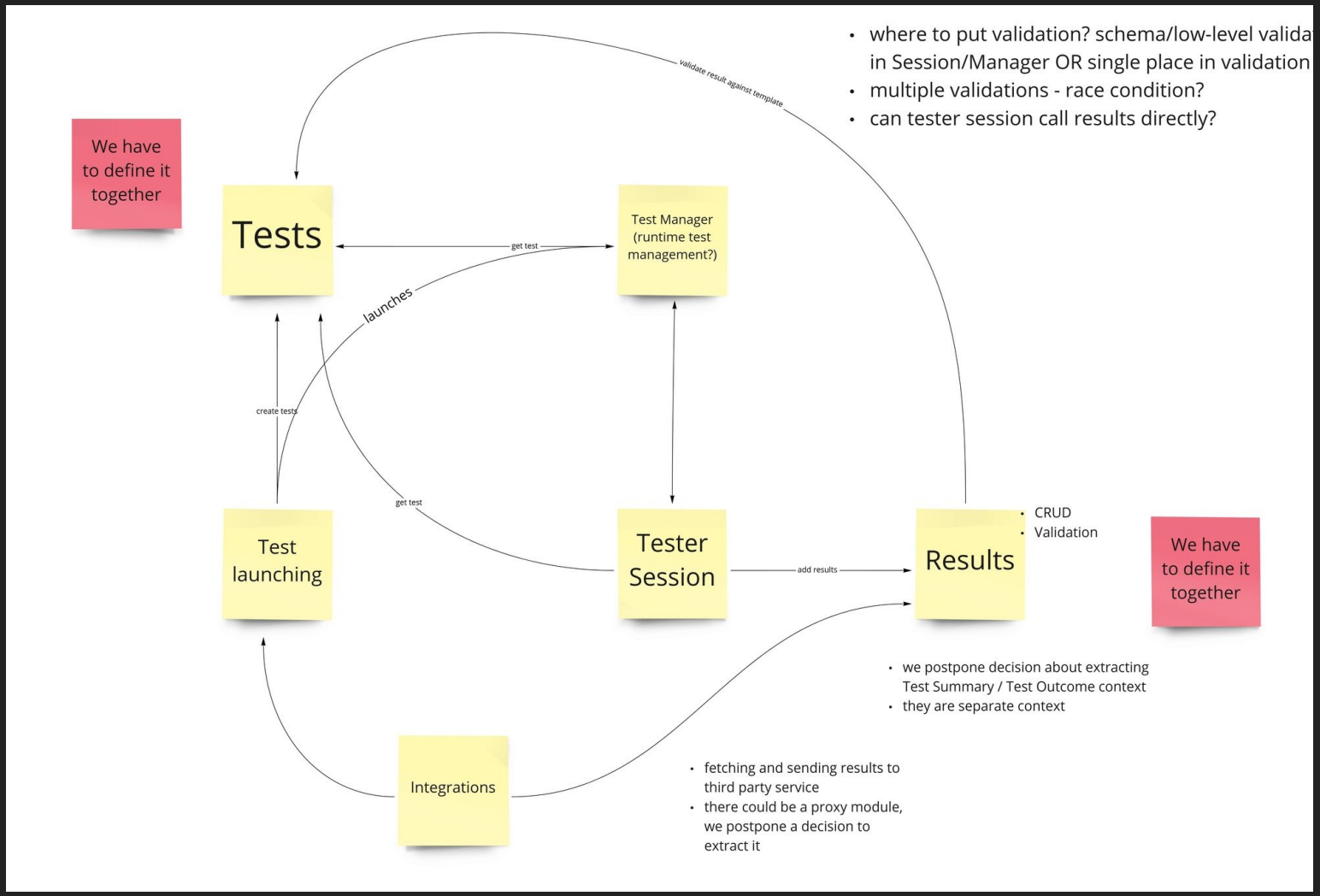


Next step - module discovery

Going deeper



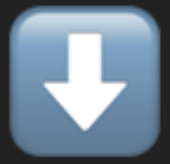
Treasure


We have to define it together
Tests
Test Manager (runtime test management?)
get test
launches
create tests
validate result against template
get test
Test launching
Tester Session
add results
Results
CRUD
Validation
We have to define it together
Integrations
where to put validation? schema/low-level valida
in Session/Manager OR single place in validation
multiple validations - race condition?
can tester session call results directly?
we postpone decision about extracting Test Summary / Test Outcome context
they are separate context
fetching and sending results to third party service
there could be a proxy module, we postpone a decision to extract it

"The Blacksmith" by

Michele Frigo on CGsociety

conceptual clarity



technical choices

Some of choices

1. Using Rails engine for separation

2. Contexts organized into isolated modules

3. Modules communicating via classes under Api namespace, eg. tests/api/test.rb, integrations/api/request.rb

More about choices

GAT Engineering Blog

https://gat.engineering

Document your choices

Document your choices

You will be coming back to it in the times of doubt

Great tool for storing such decisions - Architecture Decision Record (ADR)

Great tool for storing such decisions - Architecture Decision Record (ADR)

Good source to start -
https://adr.github.io/

Act 2

Ride the Lightening

Source: https://www.wargamer.com/dnd/lost-mine-of-phandelver

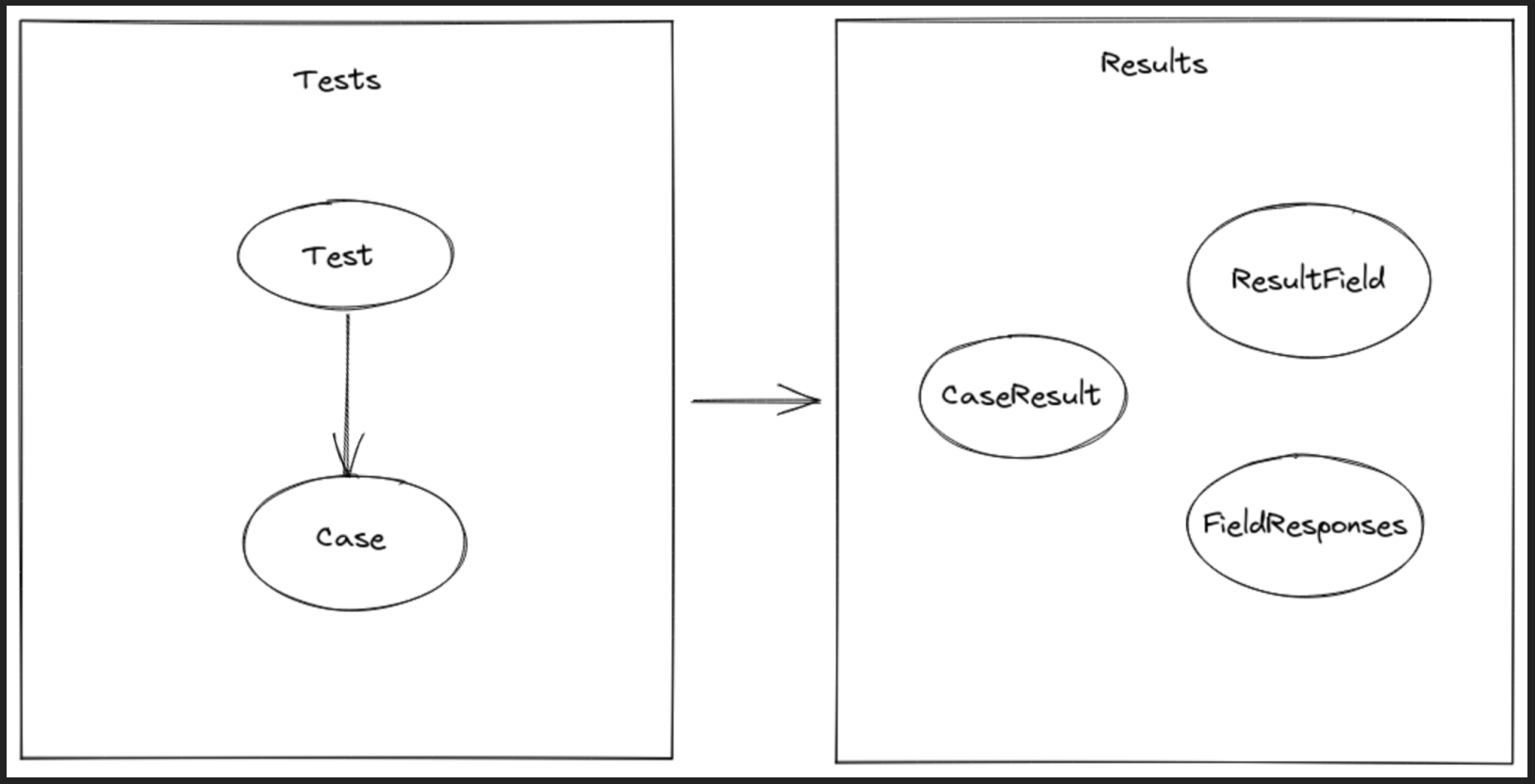
+ Implementation
+ Insight
+ Requirements

= Monsters

Evil Twins

"The Matrix Reloaded" (2003)

Tests & Results



Data used together

1. Testers - results + test and case

2. Customers - results + test and case

3. Results exported - some test and case data needed

Pain

- overhead of fetching data from two places
- a need to provide for batch fetching to prevent n+1

High coupling of the modules

Solution

Fusion

Succubus

"Succubus" by Tira-Owl on deviantart

Succubus

(a.k.a. ActiveRecord)

"Succubus" by Tira-Owl on deviantart

Isn't ActiveRecord just amazing?

- Keep all your models in one place - /app/models is cozy place for everything

- Keep all your models in one place - /app/models is cozy place for everything
- Access any model you want from any place in your app, why wouldn't you?

- Keep all your models in one place - /app/models is cozy place for everything
- Access any model you want from any place in your app, why wouldn't you?
- Traverse your models as you see fit, and then just update what you need

It's beautiful

It's evil

What DDD tells us to do?

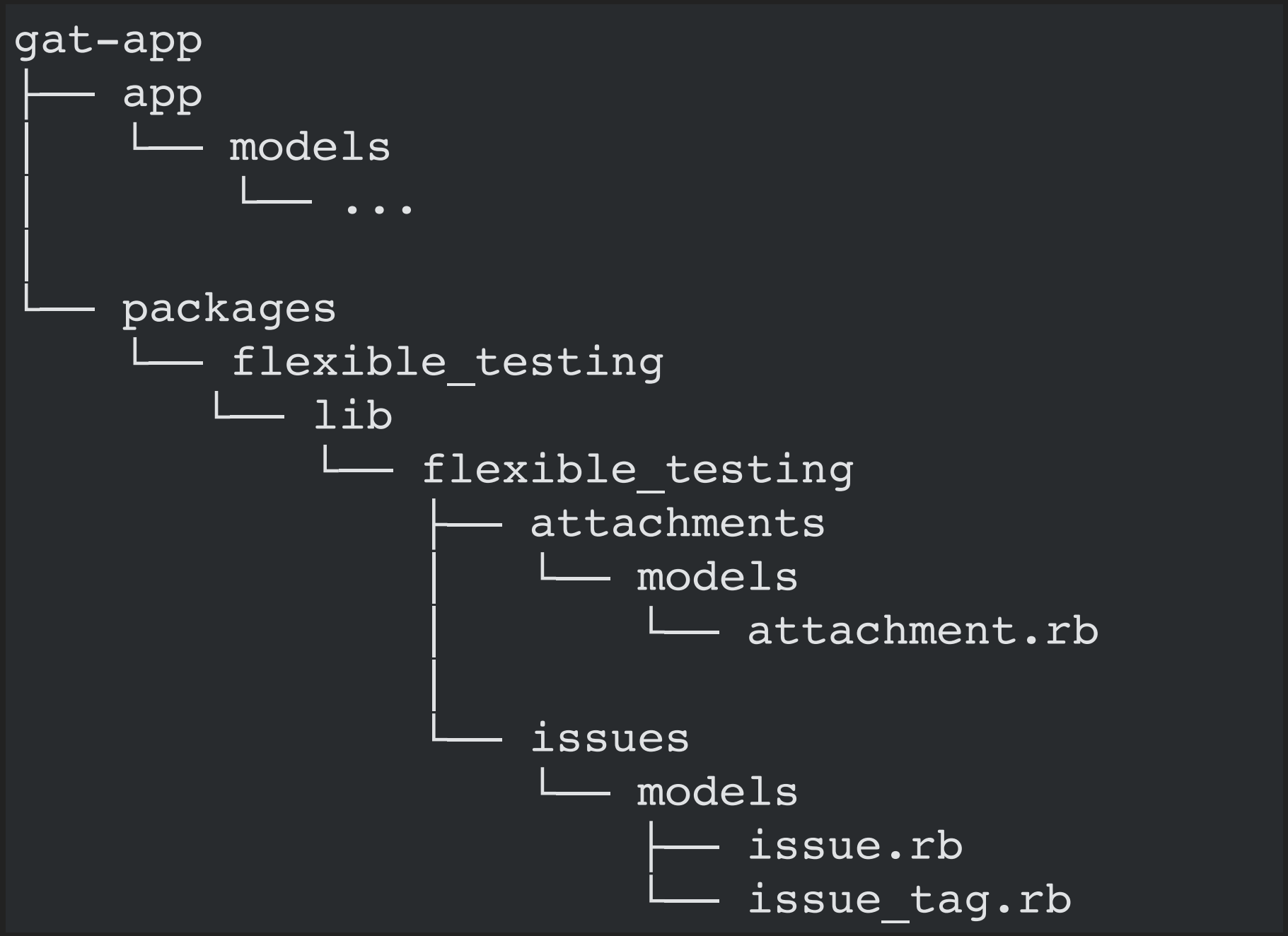
- Divide application into independent modules that own their data

- Divide application into independent modules that own their data
- Access your data through specified classes which keep state rules for trees of objects

- Divide application into independent modules that own their data
- Access your data through specified classes which keep state rules for trees of objects
- Separate interfaces for command and query, to limit side-effects of the code

Solution

Keep AR models in the corresponding modules


gat-app
app
models
...
packages
flexible_testing
lib
flexible_testing
attachments
models
attachment.rb
issues
models
issue.rb
issue_tag.rb

Separate read and write models

```ruby
module FlexibleTesting
  class ApplicationRecord < ActiveRecord::Base
    self.abstract_class = true
  end
end

module FlexibleTesting
  class ReadOnlyRecord < FlexibleTesting::ApplicationRecord
    self.abstract_class = true

    def readonly?
      true
    end
  end
end
```

```ruby
# Used for queries
module FlexibleTesting
  module TestManager
    module Api
      module Models
        class TestWindow < ReadOnlyRecord
          ...
        end
      end
    end
  end
end

# Used for commands
module FlexibleTesting
  module TestManager
    module Models
      class TestWindow < ApplicationRecord
        ...
      end
    end
  end
end
```

Pass ids or Data Transfer Objects instead of AR models

```ruby
module FlexibleTesting
  module Tests
    module Api
      class Test
        class << self
          # ...
          # @return [Dry::Monads::Result<FlexibleTesting::Tests::Api::DTO::Test, ...]
          #     Test as DTO in case of success, or Failure object
          def create(params)
            FlexibleTesting::Tests::Tests::Actions::Create.call(params: params)
          end
        end
      end
    end
  end
end

module FlexibleTesting
  module Tests
    module Api
      module DTO
        class Test < TestParams
          attribute :id, Types::UUID
          attribute :spec, TestSpec
          attribute :cases, Types::Array.of(Case)
        end
      end
    end
  end
end
```

Doppelganger

Source: https://www.dndbeyond.com/monsters/doppelganger

Ubiquitous Languague

Ubiquitous Languague

One of key concepts of DDD

Ubiquitous Languague

One of key concepts of DDD

Communication is the key

Ubiquitous Languague

One of key concepts of DDD

Communication is the key

Shared language for
people and code

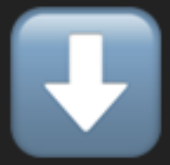
Some terms change their meaning

Some terms change their meaning

without warning

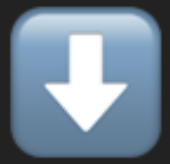
flexible testing

flexible testing

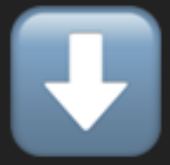


SaaS platform

flexible testing



SaaS platform



On-Demand platform

Test

Test

Test Run

Test

Test Run

= the same?

Ubiquitous Language?

Solution

Solution

(in progress)

Keep discipline in the vocabulary

Keep discipline in the vocabulary

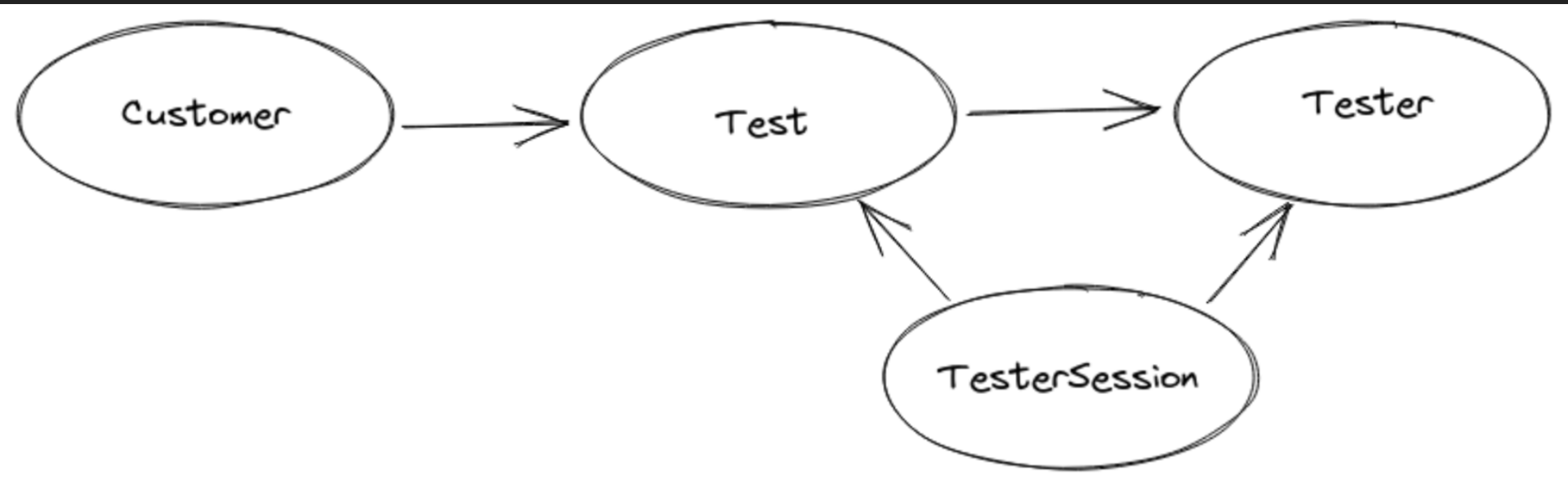
Discussions - language needs adjustments?

Dragon

"The red dragon" by Agnieszka Kwiecińska on artstation

New requirement revealing a concept that does not fit our current design

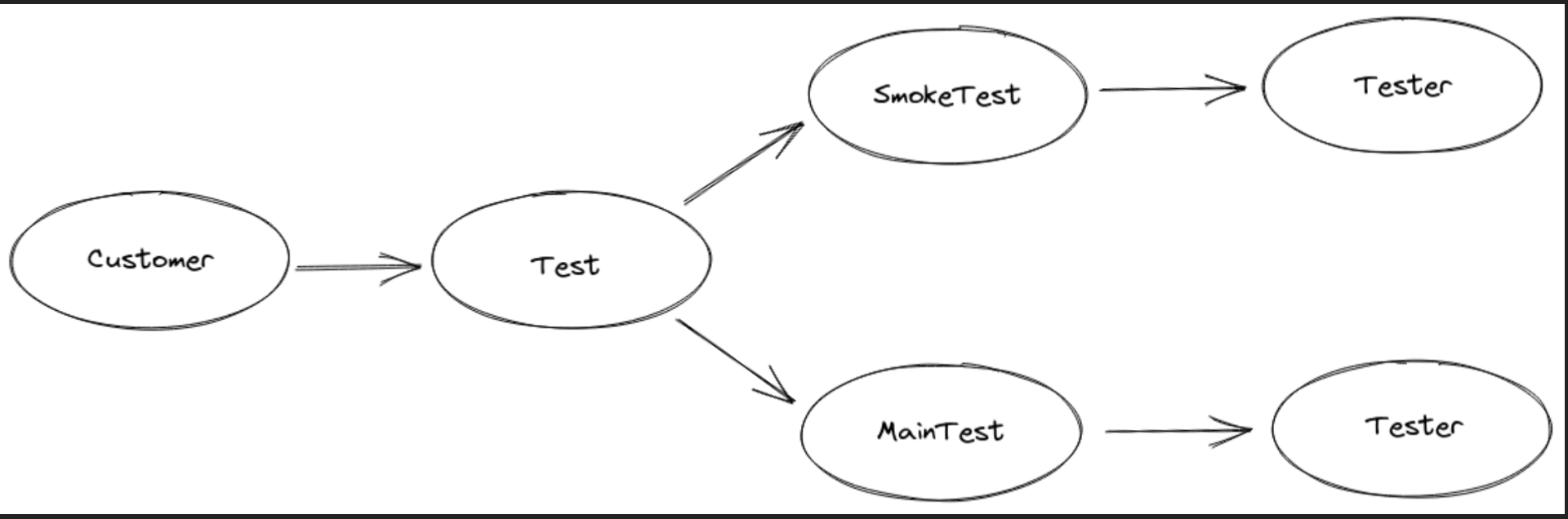
Current flow



New requirements

1. Before the actual test there will be a "smoke test"
2. "Smoke test" will consist of fixed, simple instructions (check entrypoint, log in)
3. If "smoke test" fails, the actual test will not be executed

Option 1

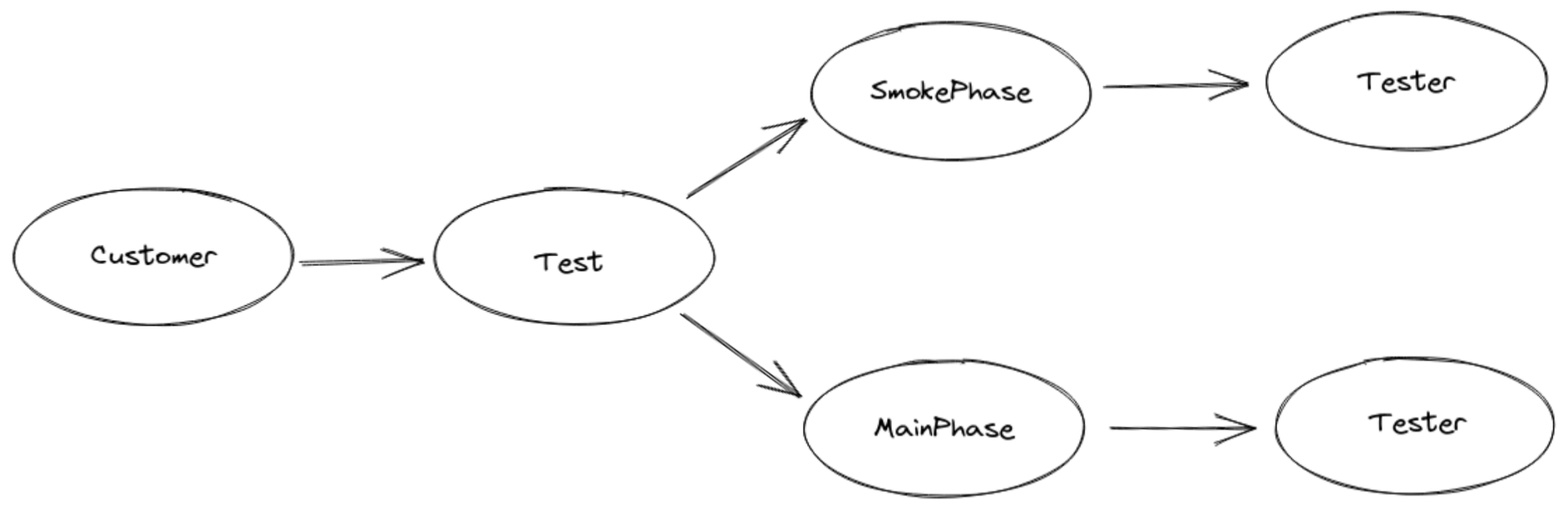


Problems

Test != Test

Test has many Tests

Option 2



Problems

Tester solves "Phases", not "Tests"

TesterSession relies on test_id + tester_id

Cracks in the model - model needs adjustments

At the same time...

Possible reward - Deeper Model

Solution

Solution

(in progress)

Be brave - adjust your domain model

Be brave - adjust your domain model

Look for a deeper model

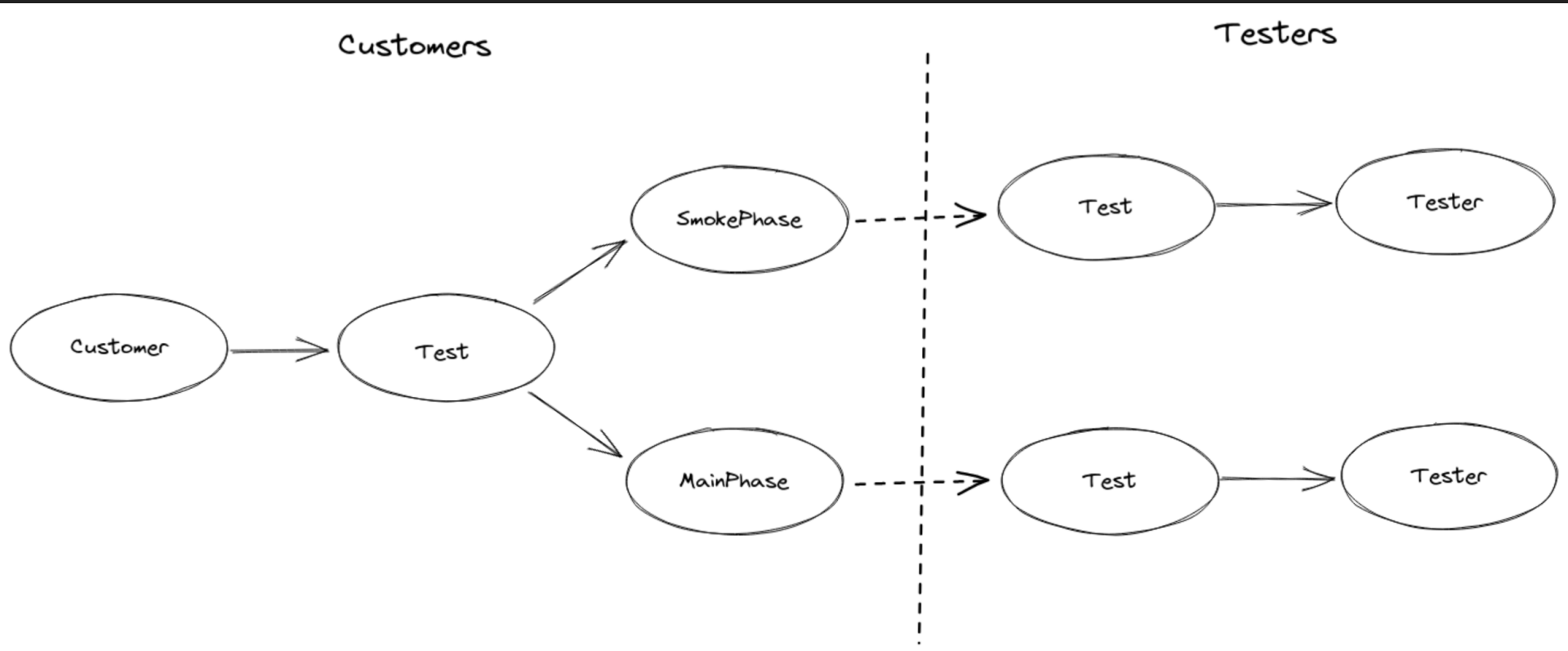
Bounded Context

Ubiquitous Language is limited to a Bounded Context

In two Bounded Contexts terms can have different meaning

Deeper Model

possibly



"The Kill" by Lanaluu on deviantart

Act 3

Act 3

What's next

Winter is coming

Winter is coming

and we are preparing

Retrospect

Retrospect

yearly revisions, to learn from mistakes

Spread knowledge

Spread knowledge

bookclub for ancient manuscript of Eric Evans

Iterate

Iterate

yet another Event Storming,
after all the gathered insight

Is it worth it?

Yes!

clarity

encapsulated
responsibilities

visible dependencies

clear extension points

Next adventure awaits

Source: https://www.dndbeyond.com/sources/lmop

wanna join?

Global App Testing

We're hiring!

Global App Testing

https://gat.engineering

for the blog and open positions

Thank you!


https://twitter.com/PedroTheCurious


https://github.com/bryszard

Piotr Brych @

Q&A