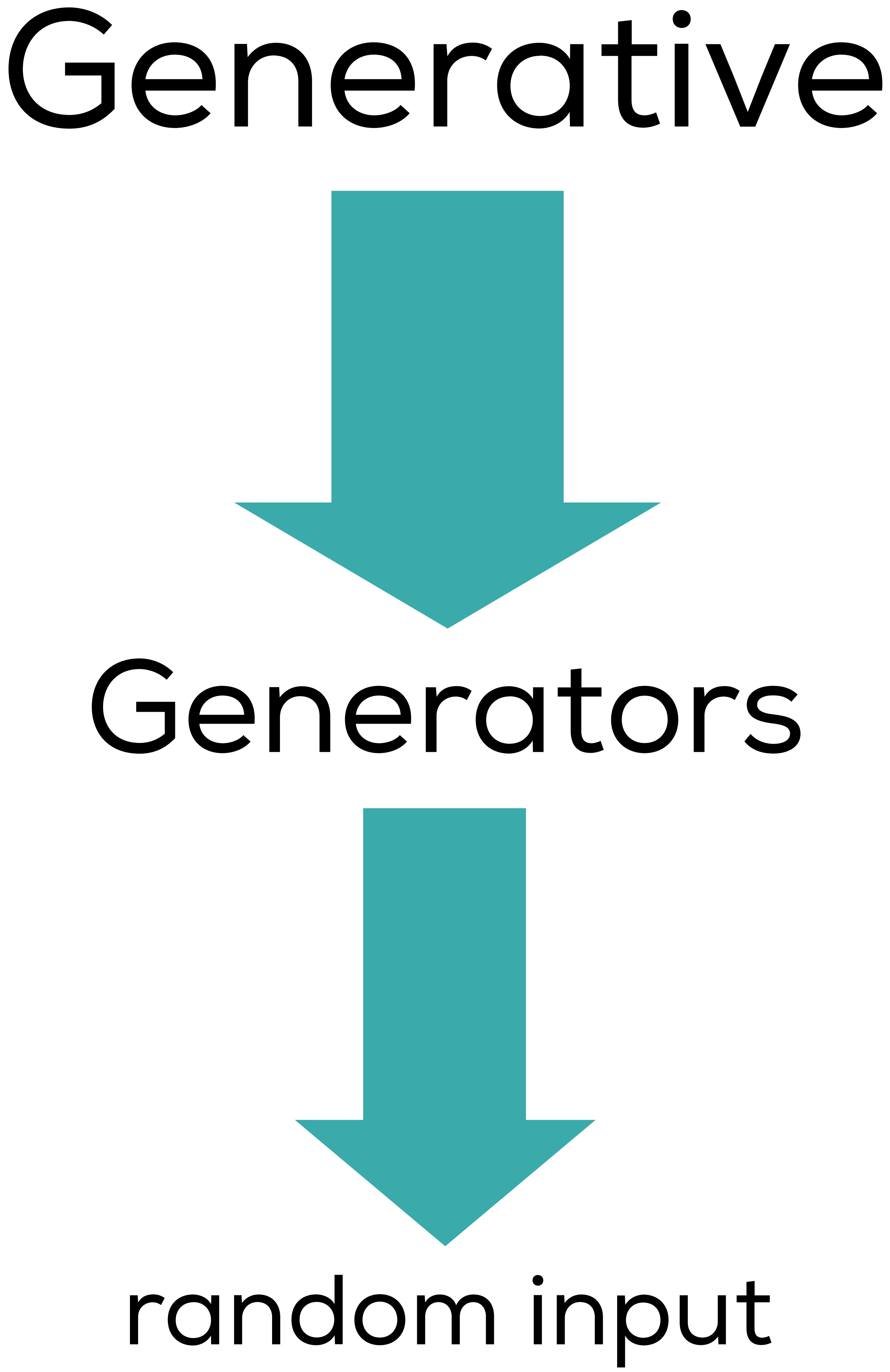
# @jessitron

Outpace™

# Generative Testing for Better Code

TDD is dead!

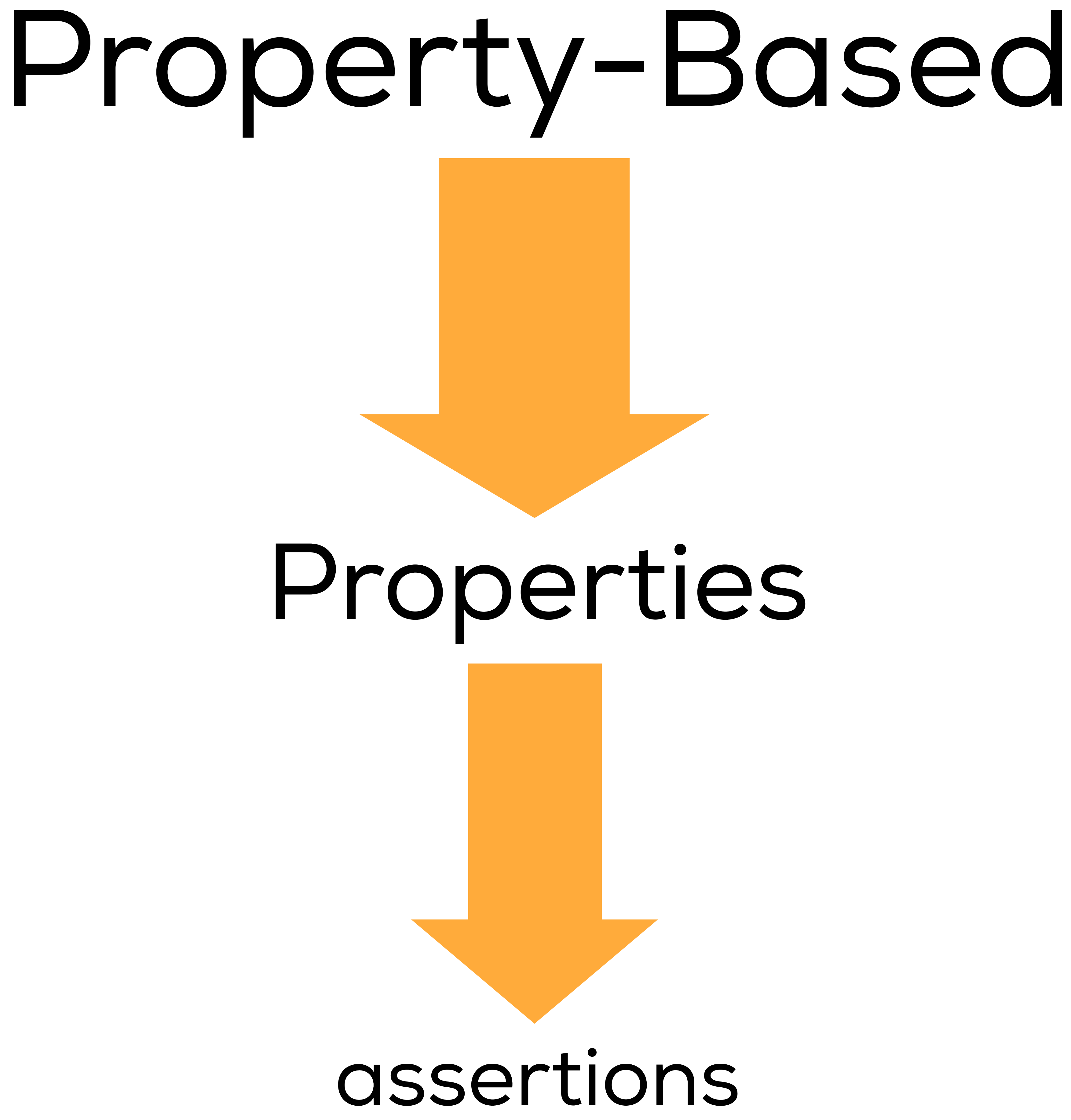
Long live TDD!



Generative

Generators

random input



Property-Based

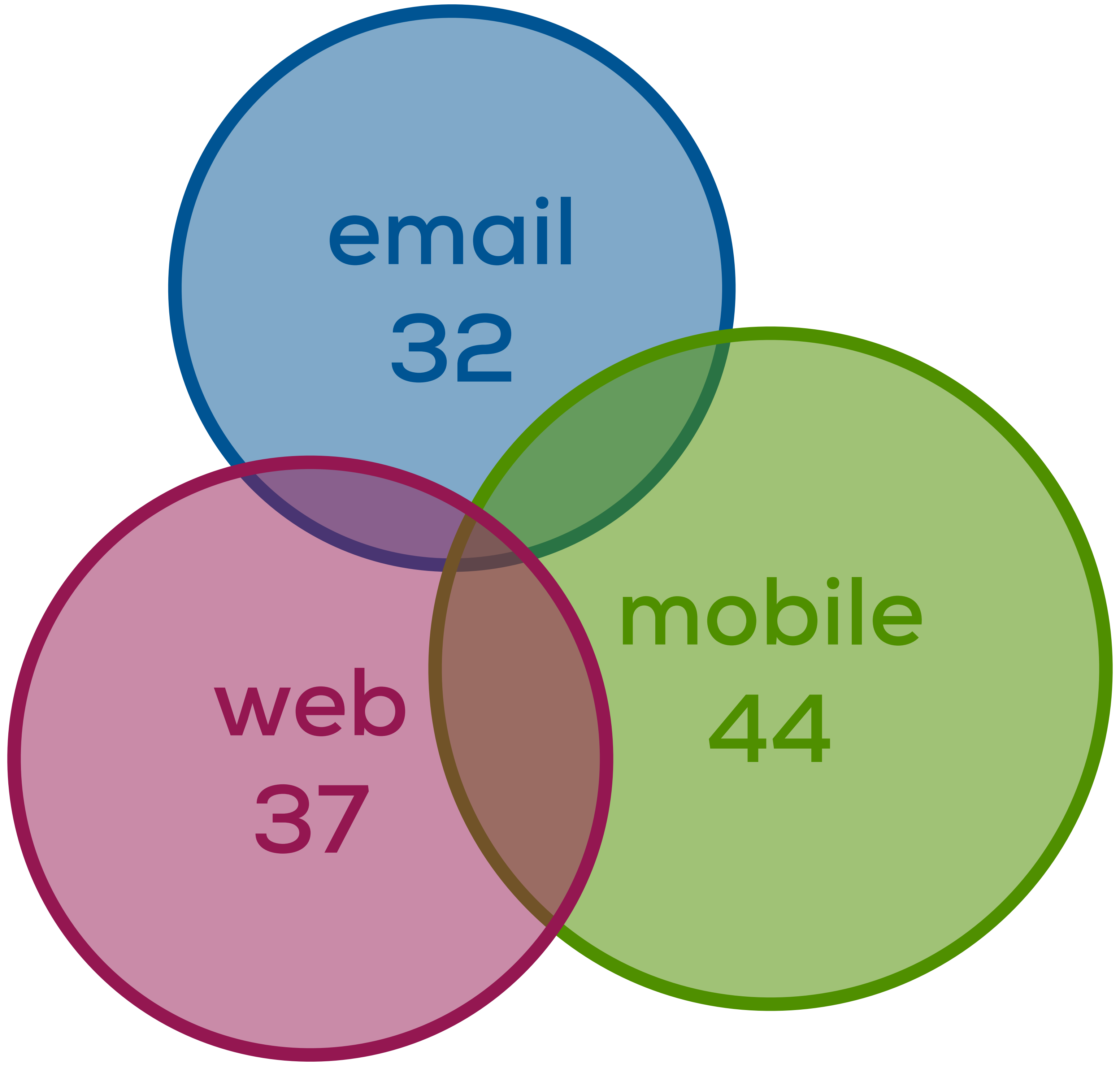
Properties

assertions

`expected =? actual`

`actual`

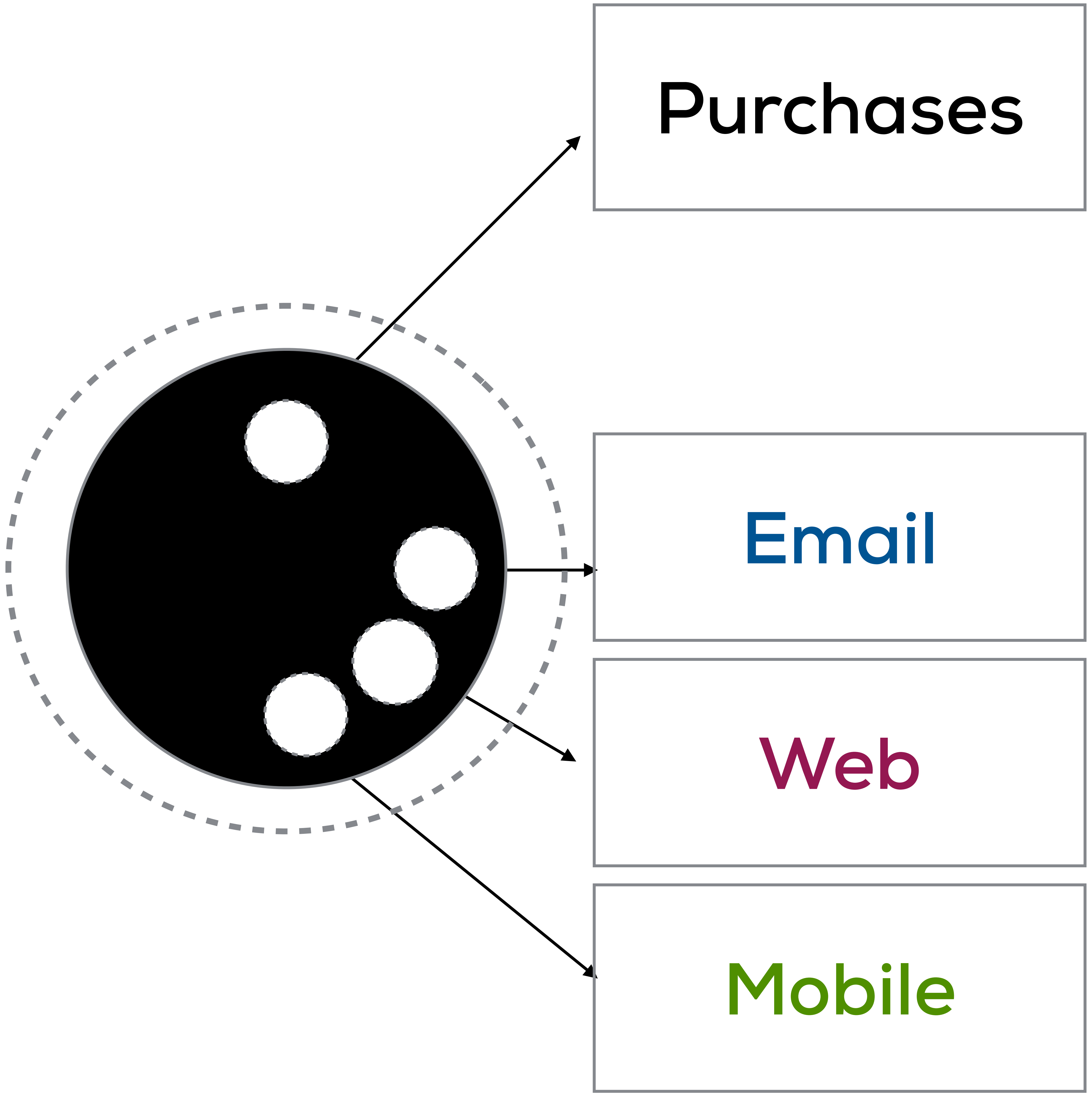
1 Test the API.

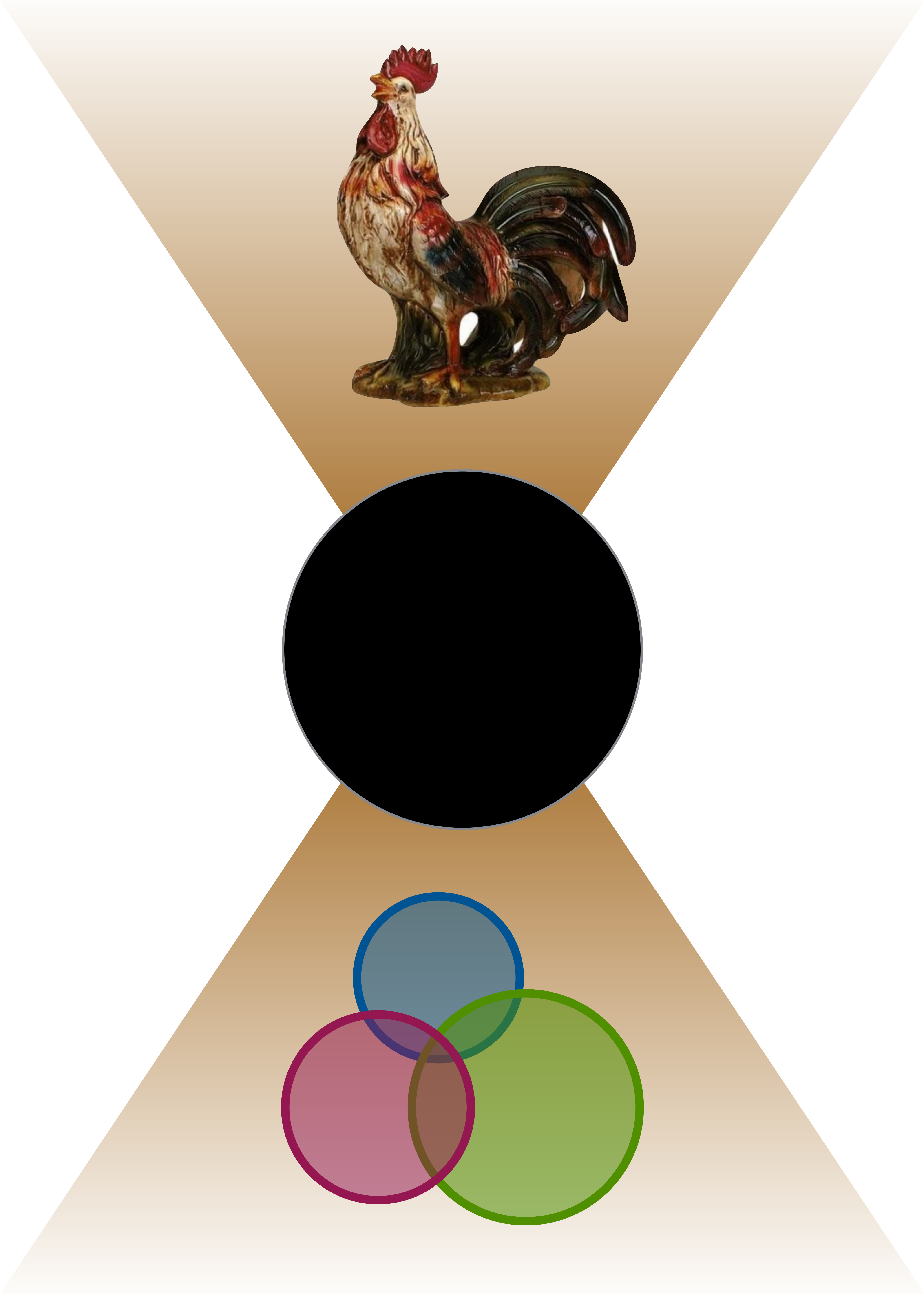


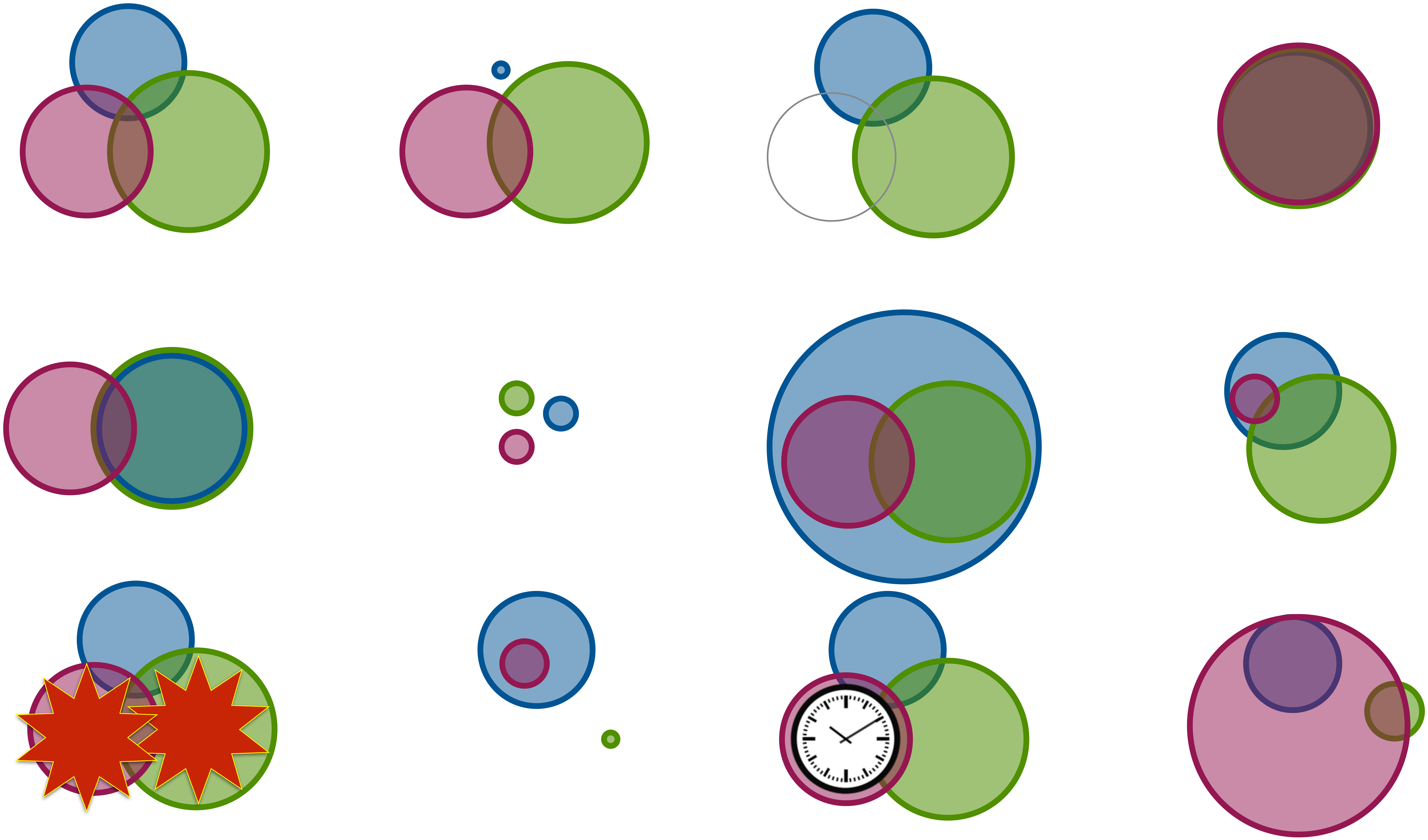
email
32

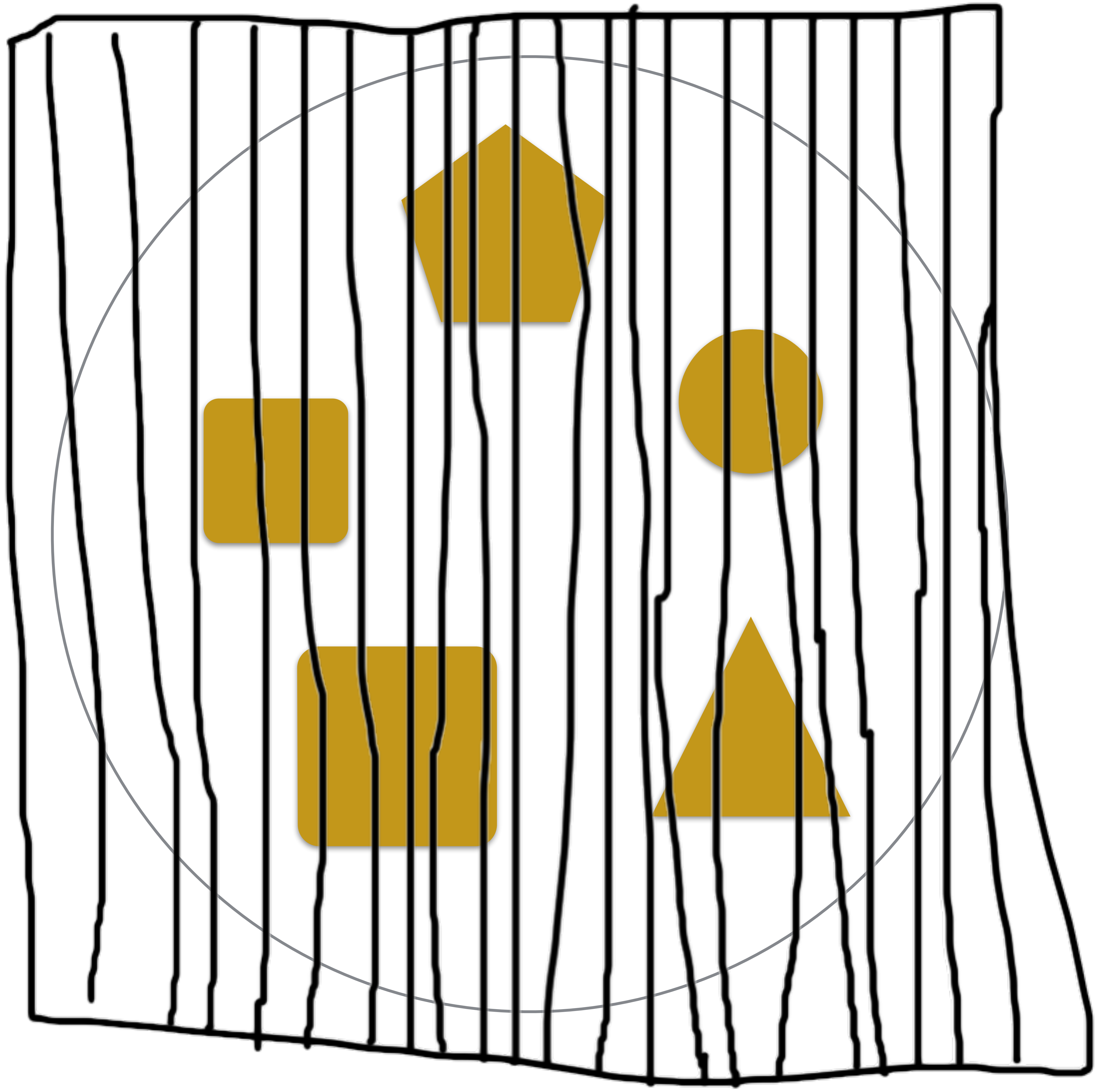
mobile
44

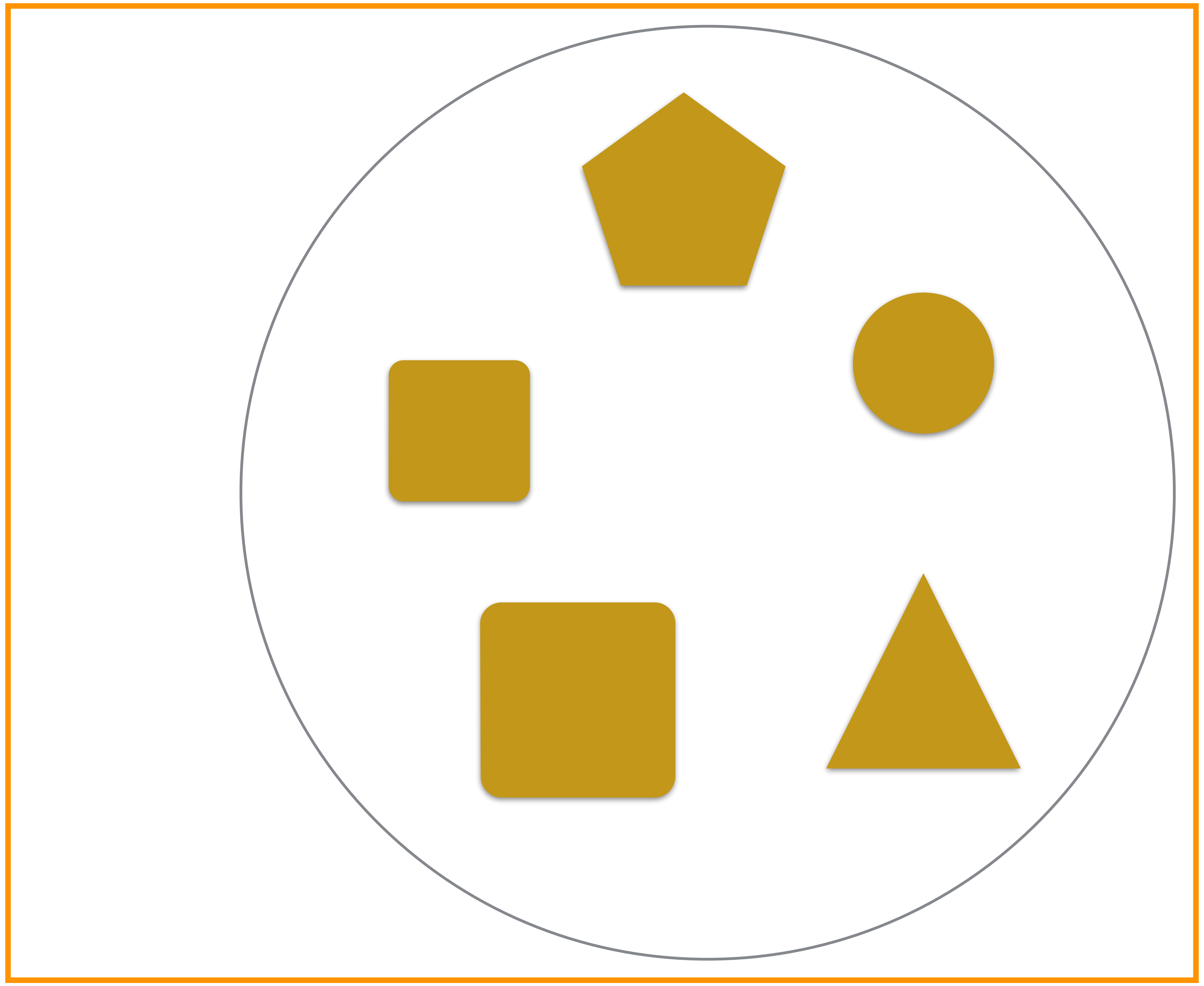
web
37

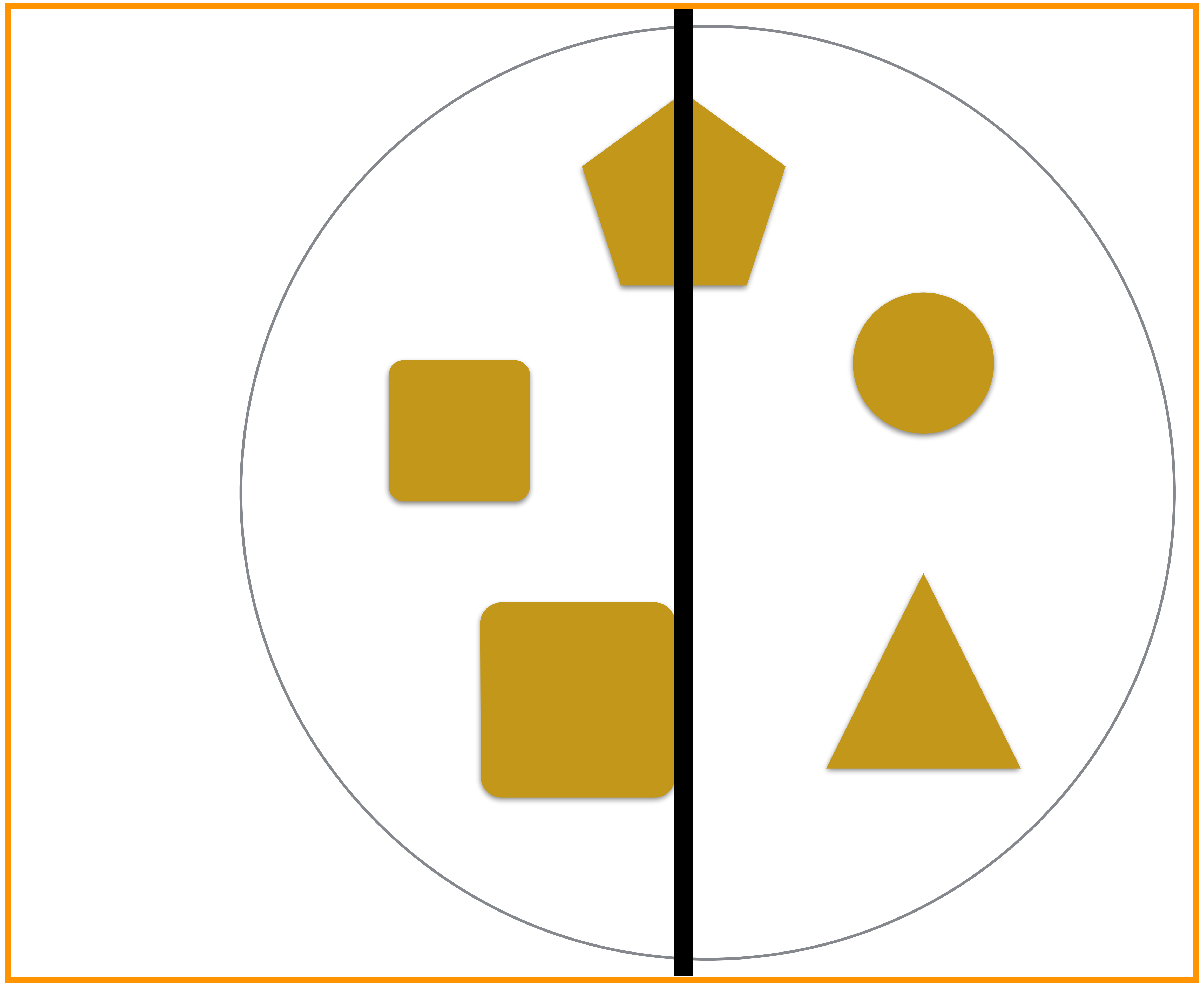


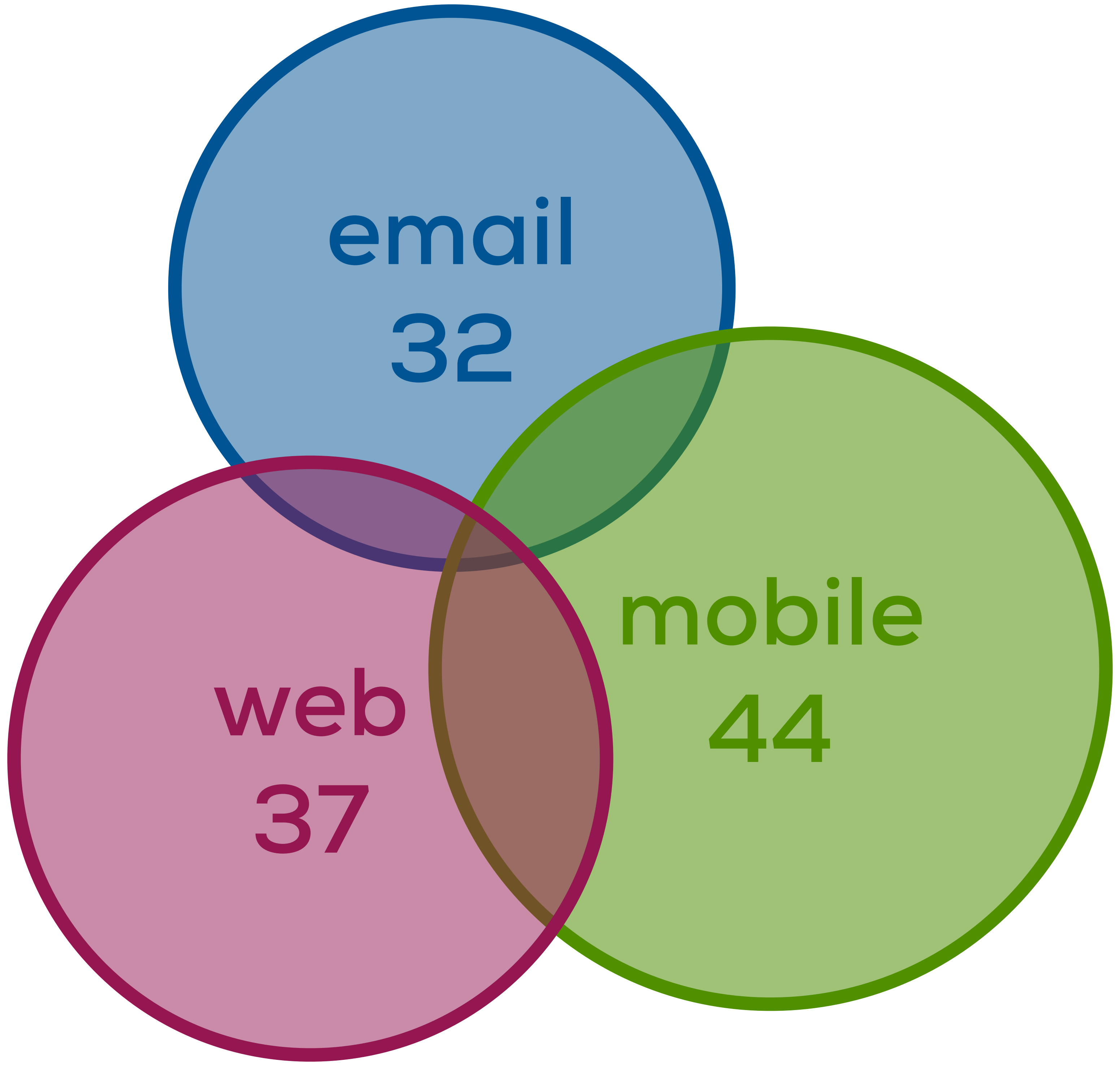











email
32
mobile
44
web
37

```
expect(influenced_purchases).to be <= total_purchases
```

**Property**: under specified circumstances, it's always true.

```
it "returns a reasonable amount of influence" do
  property_of {



  }.check do |                                |




    expect(influenced_purchases).to be <= total_purchases



  end
end
```

rantly

```
it "returns a reasonable amount of influence" do
  property_of {




  }.check do |                                |


      expect(influenced_purchases).to be <= total_purchases


  end
end
```

2

Write tests backwards

```
it "returns a reasonable amount of influence" do
  property_of {




  }.check do |                                 |
    total_purchases = purchases.size

    result = InfluenceService.new(TestPurchaseAdapter(purchases),
                 make_adapters(channel_events)).investigate(item)

    result.channels.each do |(channel, influence)|
      expect(influence.num_purchases).to be <= total_purchases
      expect(influence.relevance).to be <= 100
      expect(influence.relevance).to be >= 0
    end
  end
end
```

```
it "returns a reasonable amount of influence" do
  property_of {




  }.check do |(purchases, channel_events, item)|
    total_purchases = purchases.size

    result = InfluenceService.new(TestPurchaseAdapter(purchases),
                 make_adapters(channel_events)).investigate(item)

    result.channels.each do |(channel, influence)|
      expect(influence.num_purchases).to be <= total_purchases
      expect(influence.relevance).to be <= 100
      expect(influence.relevance).to be >= 0
    end
  end
end
```

```
 it "returns a reasonable
   property_of {
Generators.of(Generators.any_number_of(CustomGenerators.purchase),
                    CustomGenerators.channel_events,
                    CustomGenerators.item).sample

   }.check do |(
     total_purchas    = purchases.size




     result.chann                  nfluence)|
       expect(influence.num_purchases).to be <=
       expect(influence.relevance).to be <= 100
       expect(influence.relevance).to be >= 0

   end
 end
```

generatron

```
def event(channel)
  Generator.new(->() {
    whence = Generators.time.sample
    customer_id = CustomGenerators.customer_id.sample
    what = CustomGenerators.event_kinds(channel).sample
    Event.new(when_time: whence, who: customer_id, what: what)
  })
end
```

3 Generators compose from small to large.

3 Generators compose from small to large.

4 Generators are worth the time.

4 Generators are worth the time.

5 Specify valid input - completely.

```
Generators.of(Generators.any_number_of(CustomGenerators.purchase)
                  CustomGenerators.channel_events,
                  CustomGenerators.item).sample
```

5 Specify valid input - completely.

```
it "returns a reasonable amount of influence" do
  property_of {
    Generators.of(Generators.any_number_of(CustomGenerators.purchase),
                  CustomGenerators.channel_events,
                  CustomGenerators.item).sample
  }.check do |(purchases, channel_events, item)|
    total_purchases = purchases.size

    result = InfluenceService.new(TestPurchaseAdapter(purchases),
                 make_adapters(channel_events)).investigate(item)

    result.channels.each do |(channel, influence)|

      expect(influence.num_purchases).to be <= total_purchases

    end
  end
end
```

```
expect(relevance.(fewer_interactions)).
       to be < relevance.(original)
```

## Relational Property:

compares two outputs

```
check do |(purchases, channel_events, item, channel)|
  unless channel_events[channel].empty?
    original = service_test(channel_events, purchases, item)
    channel_events[channel].pop # mutation
    fewer_interactions = service_test(channel_events,
                                      purchases, item)
    relevance = ->(r) {r.channels[channel].relevance}
    expect(relevance.(fewer_interactions)).
           to be < relevance.(original)
  end
end
```

## Incomplete Property:
## output limited, and not fully specified

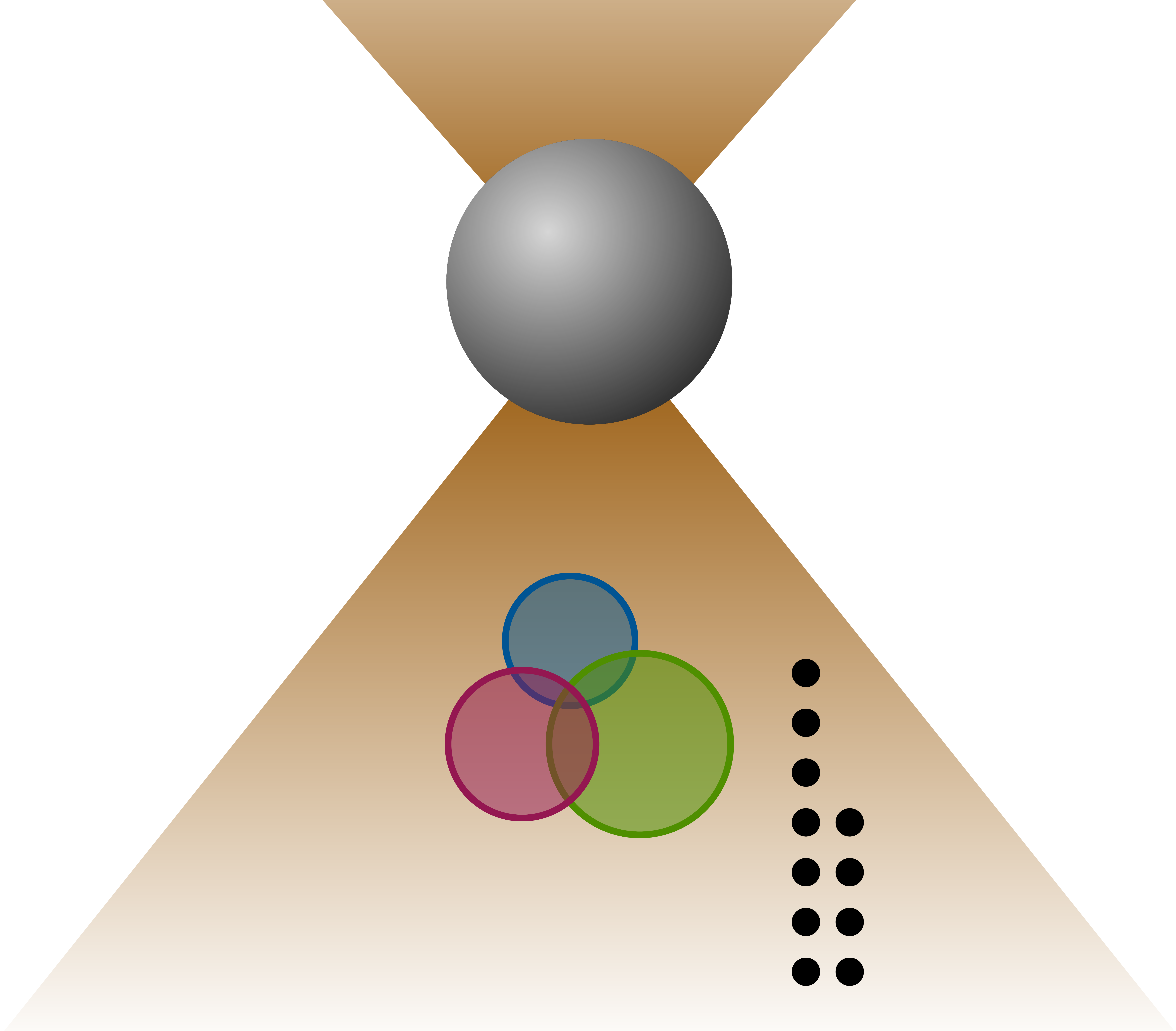
```
check do |(purchases, channel_events, item, channel)|
  unless channel_events[channel].empty?
    original = service_test(channel_events, purchases, item)
    channel_events[channel].pop # mutation
    fewer_interactions = service_test(channel_events,
                                      purchases, item)
    relevance = ->(r) {r.channels[channel].relevance}
    expect(relevance.(fewer_interactions)).
           to be < relevance.(original)
  end
end
```

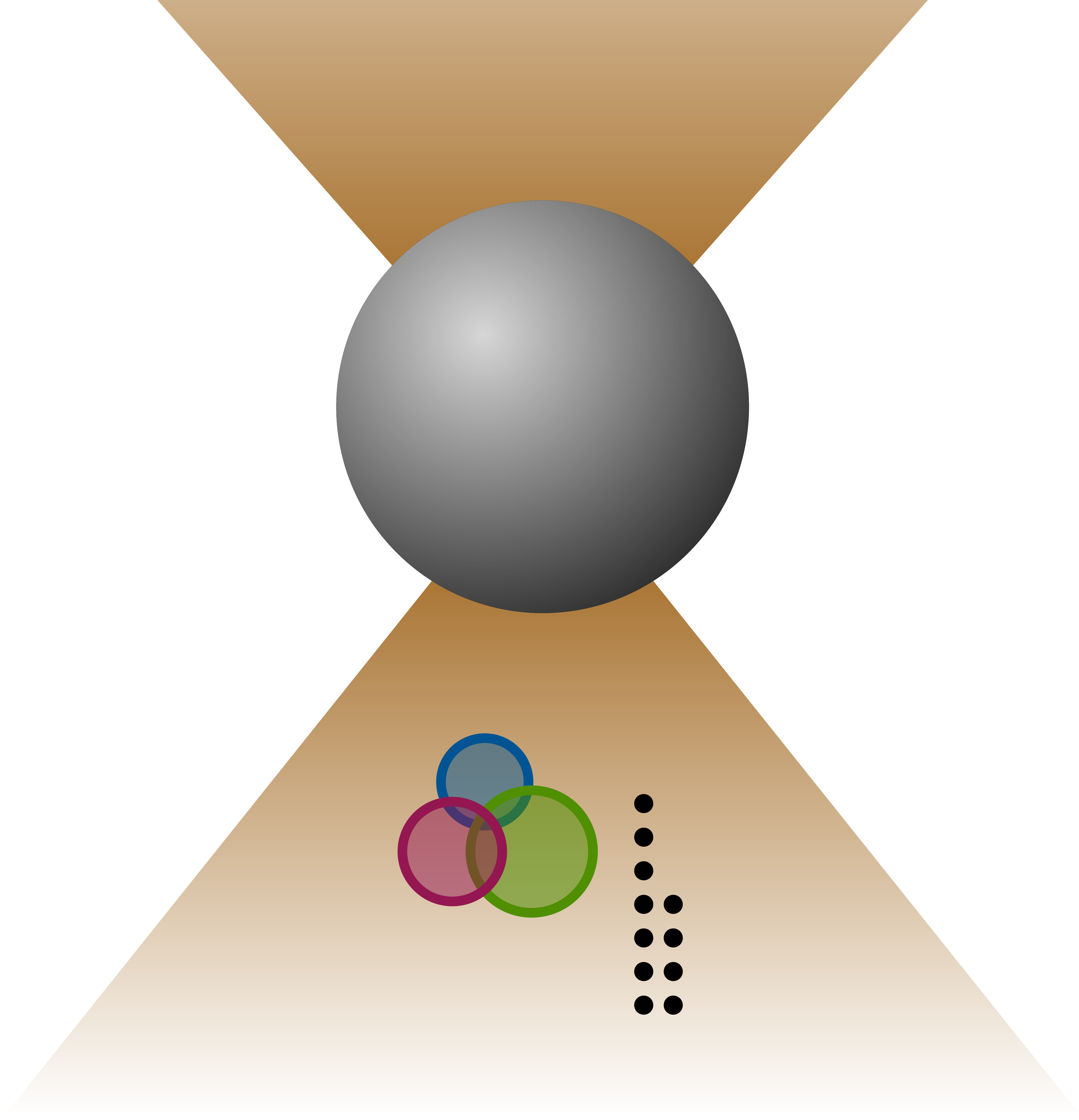
```
expect(relevance.(fewer_interactions)).
       to be < relevance.(original)
```

6 State why the output is correct.

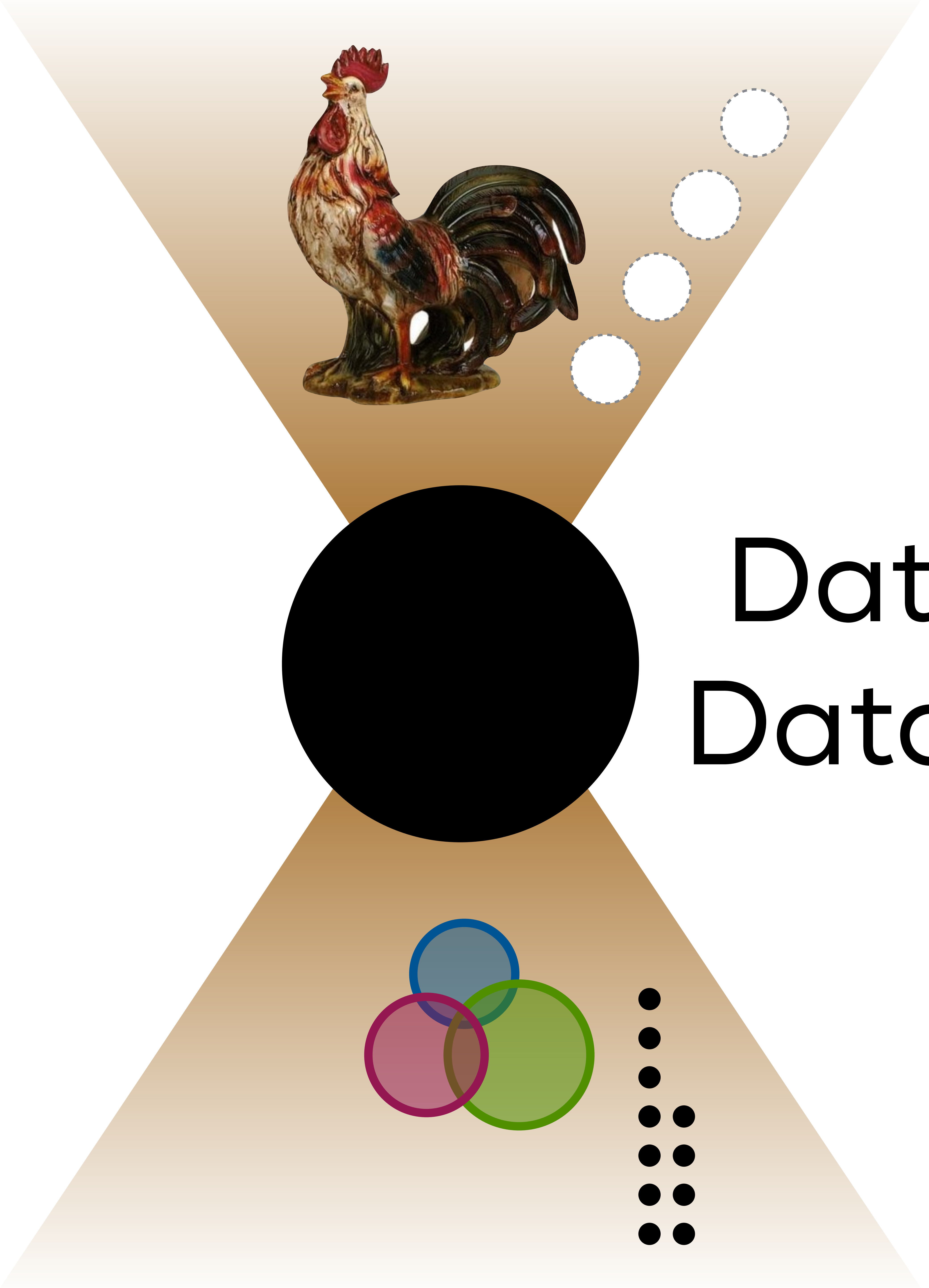
```
PurchaseAttribution
total purchases
purchase =>
     events
channel =>
     purchases
     relevance
```

7 Return everything you need.

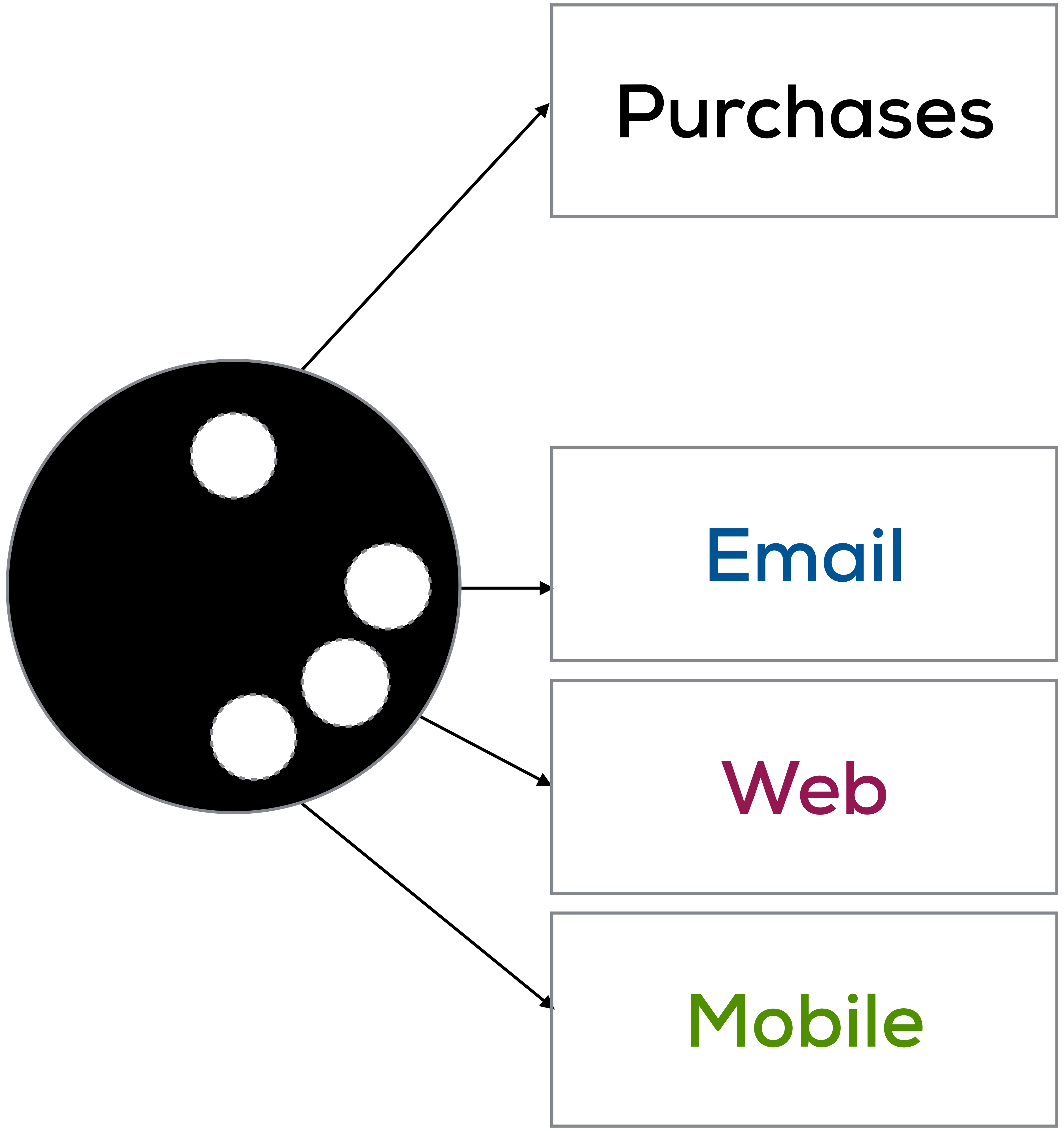




8 Create the visibility you need.

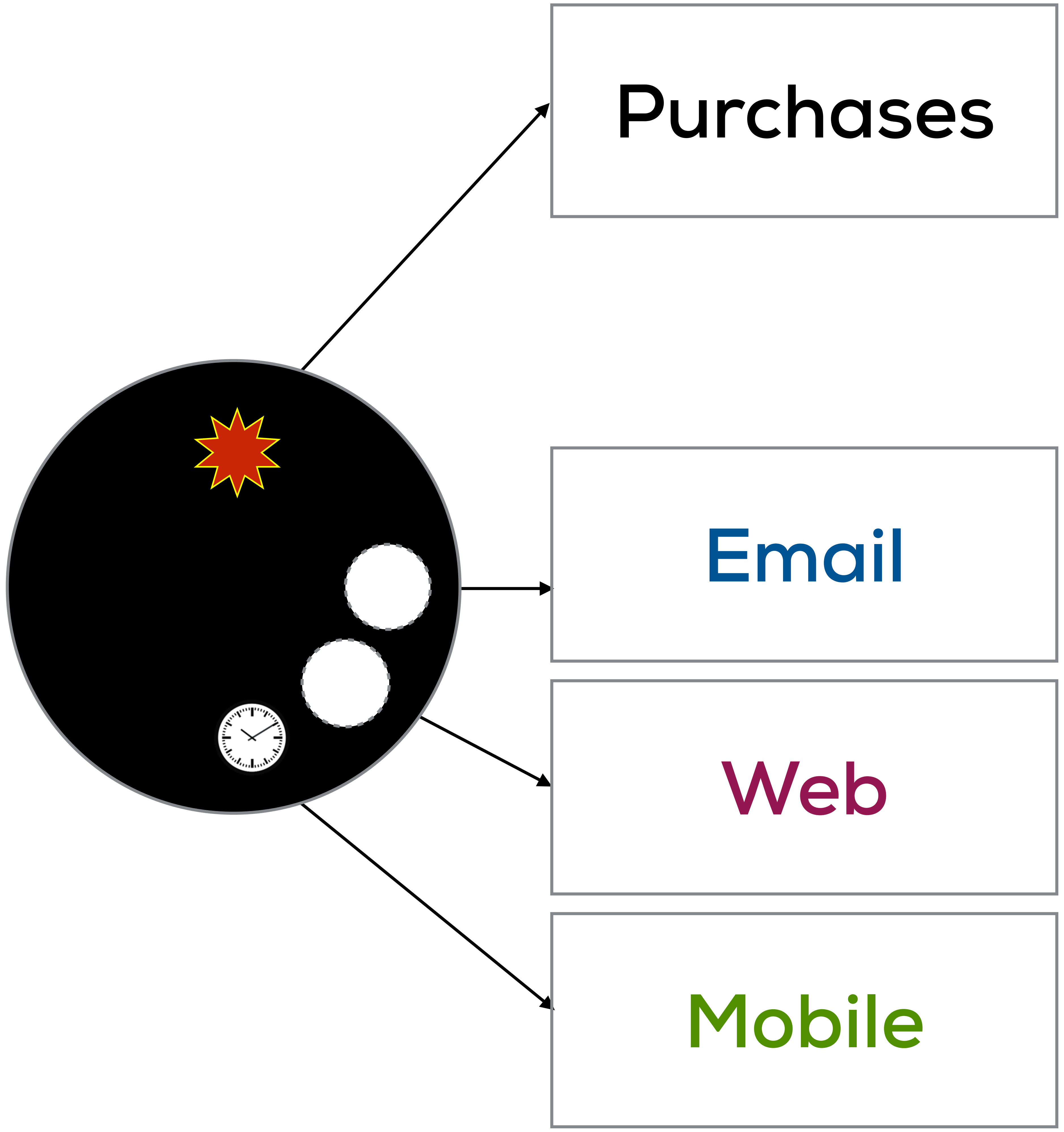


# Data In, Data Out

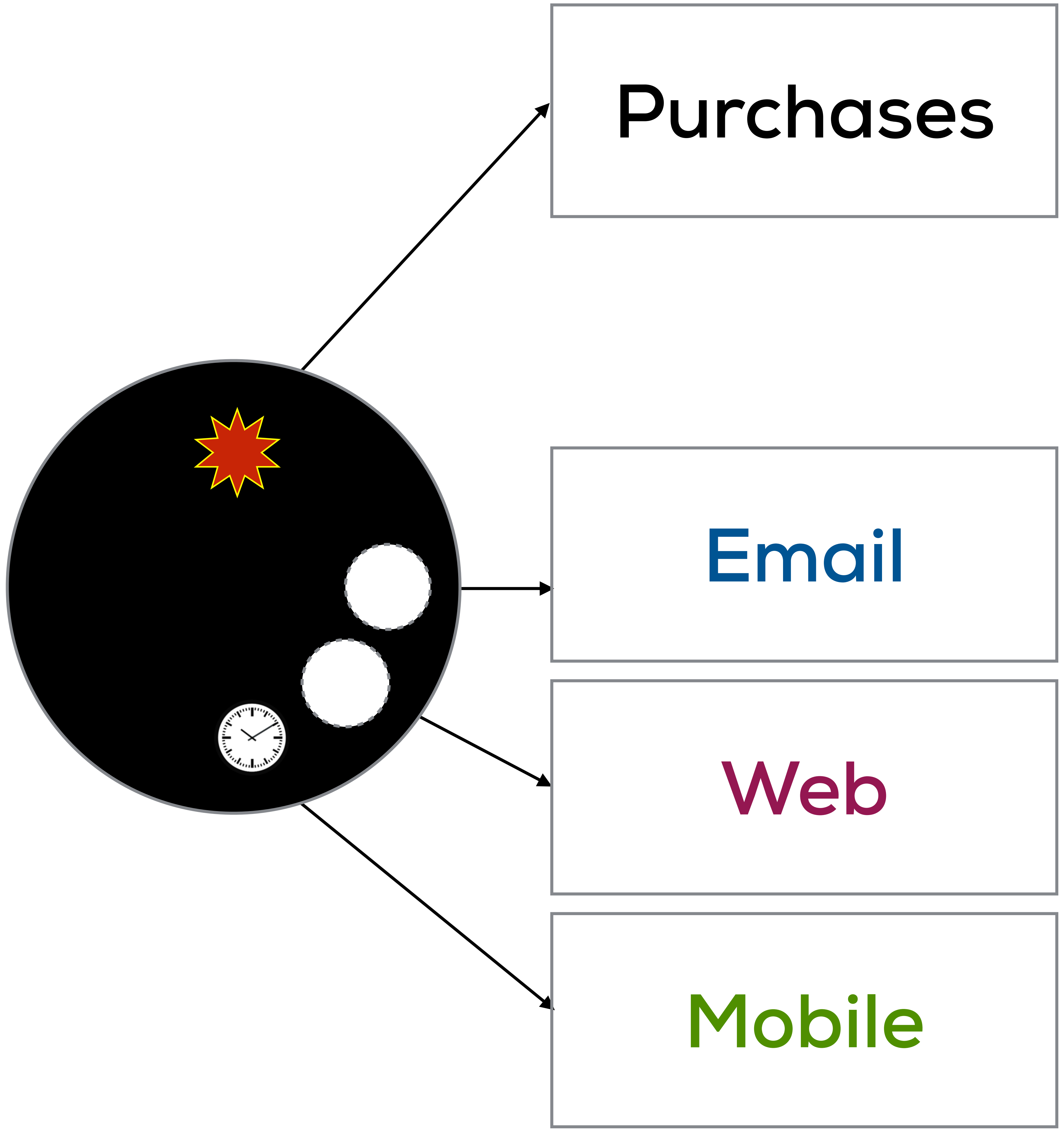


9

Use ports and adapters for external dependencies



Expect failure. Create failure.



b

Ask the hard questions

Fix the simplest case that fails.

```
[[Purchase by  at -35305655793-04-28 18:49:27 -0600,
  Purchase by  at -39062112491-05-24 14:24:58 -0600,
  Purchase by  at -59702385776-02-03 22:36:41 -0600,
  Purchase by  at -41865688111-09-19 14:45:15 -0600,
  Purchase by  at -54724415537-09-16 22:56:46 -0600,
  Purchase by  at -35097555165-06-22 21:22:47 -0600,
  Purchase by  at -55646762305-04-25 11:01:04 -0600,
  Purchase by  at -27123383184-08-06 13:28:25 -0600,
  Purchase by  at -48686476976-05-26 02:13:39 -0600,
  Purchase by  at -66566943307-10-29 15:56:14 -0600,
  Purchase by  at -40583444671-09-09 17:53:31 -0600,
  Purchase by  at -27576960979-01-06 04:01:25 -0600,
  Purchase by  at -56390460527-12-08 06:12:27 -0600,
  Purchase by  at -47279113115-08-05 21:48:35 -0600,
  Purchase by  at -45043691009-12-13 20:43:57 -0600,
  Purchase by  at -61333949847-09-21 13:00:52 -0600,
  Purchase by  at -16767473371-04-14 06:10:37 -0600,
  Purchase by  at -42553484473-01-28 05:56:37 -0600,
  Purchase by  at -69137219711-09-22 02:25:15 -0600,
  Purchase by  at -4438901838-03-13 02:47:04 -0600,
  Purchase by  at -6597679432-03-09 06:19:35 -0600,
  Purchase by  at -49502328516-04-07 07:35:26 -0600,
  Purchase by  at -34417185577-06-13 02:17:06 -0600,
  Purchase by  at -57807426455-12-06 16:37:34 -0600,
  Purchase by  at -31995381873-04-20 05:40:14 -0600,
  Purchase by  at -3676257181-06-29 09:51:14 -0600,
  Purchase by  at -36553360873-04-15 23:25:03 -0600,
  Purchase by  at -33153784630-09-11 12:28:32 -0600,
  Purchase by  at -5369570892-11-20 10:52:56 -0600,
  Purchase by  at -38713850818-04-07 09:45:14 -0600,
  Purchase by  at -35175145749-01-24 02:30:33 -0600,
  Purchase by  at -64192026206-12-28 09:08:10 -0600,
  Purchase by  at -37429213879-08-16 02:55:26 -0600,
  Purchase by  at -25557703902-09-13 13:16:56 -0600,
  Purchase by  at -52166240207-03-09 07:24:52 -0600,
  Purchase by  at -65258943847-02-20 14:10:57 -0600,
  Purchase by  at -50524079495-05-04 22:18:33 -0600,
  Purchase by  at -55068693665-09-19 05:05:21 -0600,
  Purchase by  at -70097414732-06-20 04:23:24 -0600,
  Purchase by  at -68532071431-11-07 19:01:26 -0600,
  Purchase by  at -15484798636-05-16 10:10:28 -0600,
  Purchase by  at -26014401665-01-22 10:40:39 -0600,
  Purchase by  at -57579389761-09-13 18:01:39 -0600,
  Purchase by  at -25267301314-03-21 20:38:41 -0600,
  Purchase by  at -15591407490-07-26 03:06:03 -0600,
  Purchase by  at -12385715268-04-21 09:09:18 -0600,
  Purchase by  at -41563819920-06-14 10:17:42 -0600,
  Purchase by  at -60028856529-07-12 03:46:45 -0600],
 {:web=>
   [Event: 1737683248413877657 did ad_show at -29443119863-02-11 19:02:27 -0600,
    Event: 2287326864789819693 did ad_click at -42700308373-06-07 22:36:18 -0600],
  :mobile=>[],
  :email=>
   [Event: 1796205055738615925 did click at -11105347506-12-05 13:58:38 -0600,
    Event: 1265837519996765579 did click at -59382212411-05-08 19:24:04 -0600,
    Event: 469172104420140357 did email_sent at -32125295433-08-26 03:12:39 -0600,
    Event: 860282602109566003 did email_sent at -22736181958-02-01 14:36:51 -0600,
    Event: 1384386596160009167 did click at -18292529311-08-09 18:39:49 -0600,
    Event: 1682720690208993824 did email_sent at -17780119302-09-14 01:33:48 -0600,
    Event: 1114595631533910303 did click at -14964403057-08-16 21:05:22 -0600,
    Event: 239227986960488290 did email_sent at -20849051975-12-07 19:01:56 -0600,
    Event: 795324022754106461 did email_sent at -26930198139-10-22 07:49:45 -0600,
    Event: 316531875407260475 did click at -47587658361-07-04 20:27:27 -0600,
    Event: 25906794329584858 did read at -61054112577-10-03 17:23:46 -0600,
    Event: 1781536921414098494 did email_sent at -35939242811-01-08 03:08:56 -0600,
    Event: 1440847623358112352 did email_sent at -68211190328-09-04 16:14:03 -0600,
    Event: 1926158199870687492 did email_sent at -16521176615-10-23 16:53:42 -0600,
    Event: 1302433779695373913 did email_sent at -48912654079-07-14 05:20:11 -0600,
    Event: 343156651822247838 did email_sent at -18670034320-06-21 16:24:58 -0600,
    Event: 2017213513687849233 did email_sent at -30600067080-08-24 18:02:44 -0600,
    Event: 1041995533505817110 did click at -10128867045-06-22 05:58:54 -0600,
```

```
minimal failed data is:
[[],
 {:web=>[],
  :mobile=>[],
  :email=> [Event: 860282602109566
 {:item_name=>"dadPbF"},
 :email]
```

```
found a reduced success:
[[],
 {:web=>[],
      :mobile=>[],
      :email=>[]},
 {:item_name=>"dadPbF"},
 :email]
```

```
minimal failed data is:
[[],
 {:web=>[],
  :mobile=>[],
  :email=> [Event: 860282602109566
 {:item_name=>"dadPbF"},
 :email]
```

# Shrinking

```
it 'counts search events' do
    input = [{ event: :search
             { event: :not_search },
             { event: :search



    expect(actual).to eq(expected)

end
```

```
it 'counts search events' do
    input = [{ event: :search
             { event: :not_search },
             { event: :search



    what_matters = ->(r) { r[:search] }

    expect(what_matters(actual)).
     to eq(what_matters(expected))

end
```

```
it 'counts search events' do
    input = [{ event: :search      },
             { event: :not_search },
             { event: :search      }]



    what_matters = ->(r) {

    expect(
     to eq(

end
```

```
it 'counts search events' do
    input = [{ event: :search     },
             { event: :search     }]
            + other_events



    what_matters = ->(r) {

    expect(
     to eq(

end
```

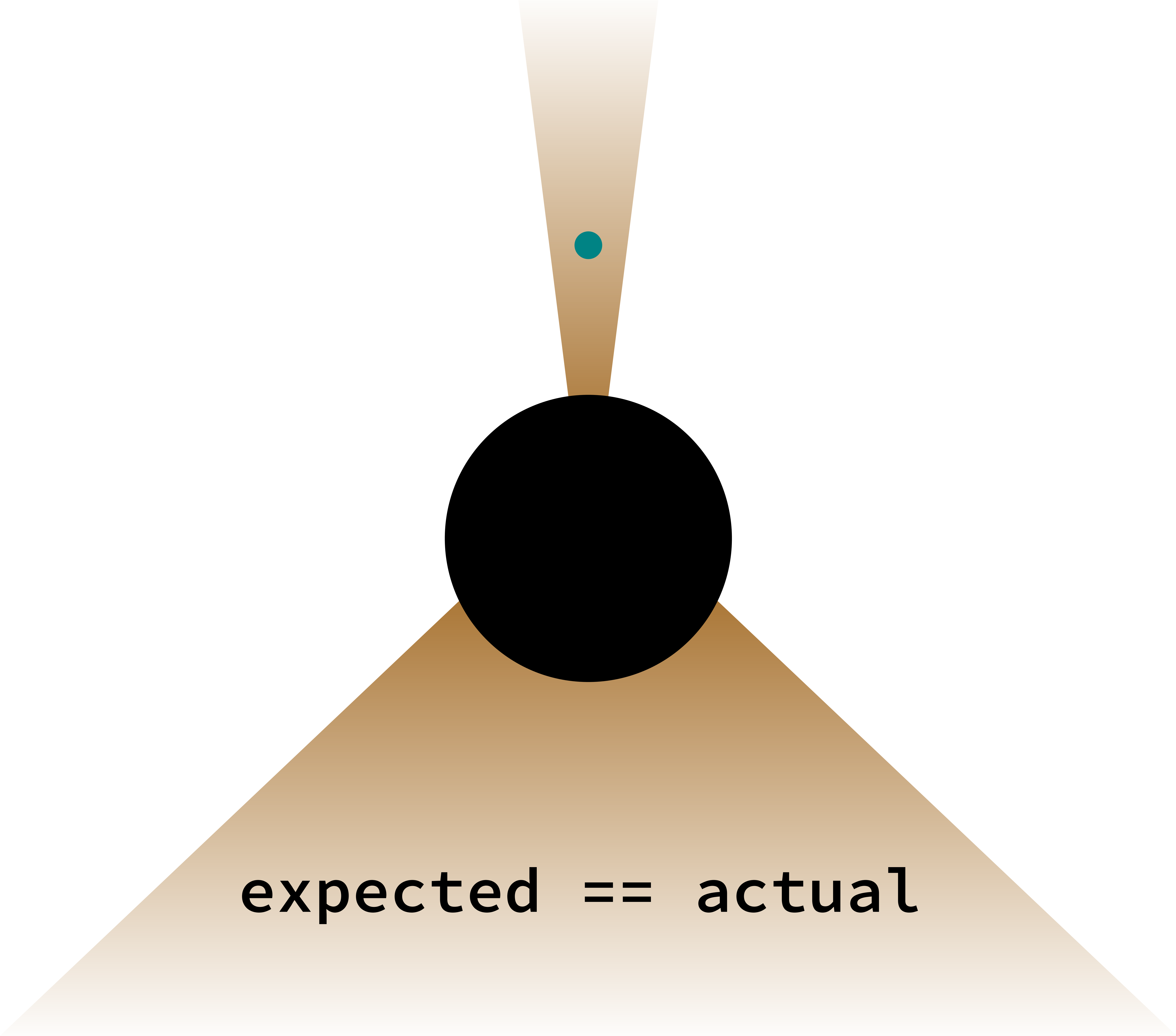
```
it 'counts search events' do
  other_events = Generators.any_number_of(
           CustomGenerators.activity_record.
           reject(search_event?)).
           sample
  input = [{ event: :search
            { event: :search
          +



  what_matters = ->(r) {

  expect(
it 'counts search events' do
  property_of {
    Generators.
            CustomGenerators.
            reject                .

  }.check do |other_events|
    input = [{ event: :search
             { event: :search
            +



    what_matters = ->(r) {
  }.check do |all_events, search_count |

    actual = Report.summarize(all_events)

    expect(actual[:search]).
     to eq(search_count)

end
```

```
  it 'counts search events' do
    property_of {
     Generators.array_len.transform(->(search_count){






       [search_events + other_events, search_count])
                                .
     }).sample
    }.check do |           ,              )
```

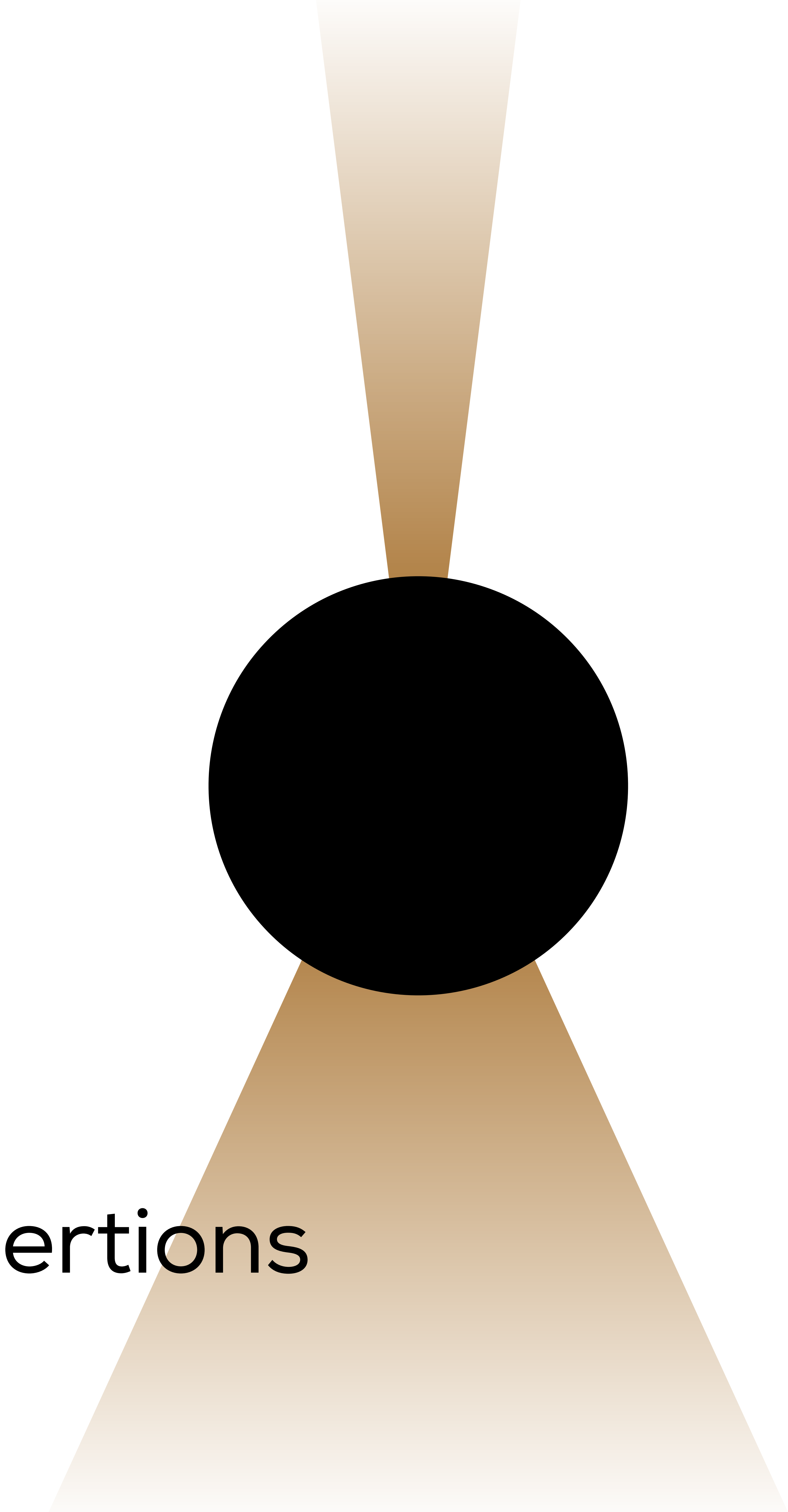
```
it 'counts search events' do
  property_of {
   Generators.array_len.transform(->(search_count){
      search_events = CustomGenerators.
                      activity_record_for(:search).
                      sample_n(search_count)




     [search_events + other_events, search_count])
   }).sample
  }.check do |           ,              )
```

```
it 'counts search events' do
  property_of {
   Generators.array_len.transform(->(search_count){
      search_events = CustomGenerators.
                      activity_record_for(:search).
                      sample_n(search_count)
      other_events = Generators.any_number_of(
             CustomGenerators.activity_record.
             reject(search_event?)).sample
     [search_events + other_events, search_count])
   }).sample
  }.check do |          ,              )
```

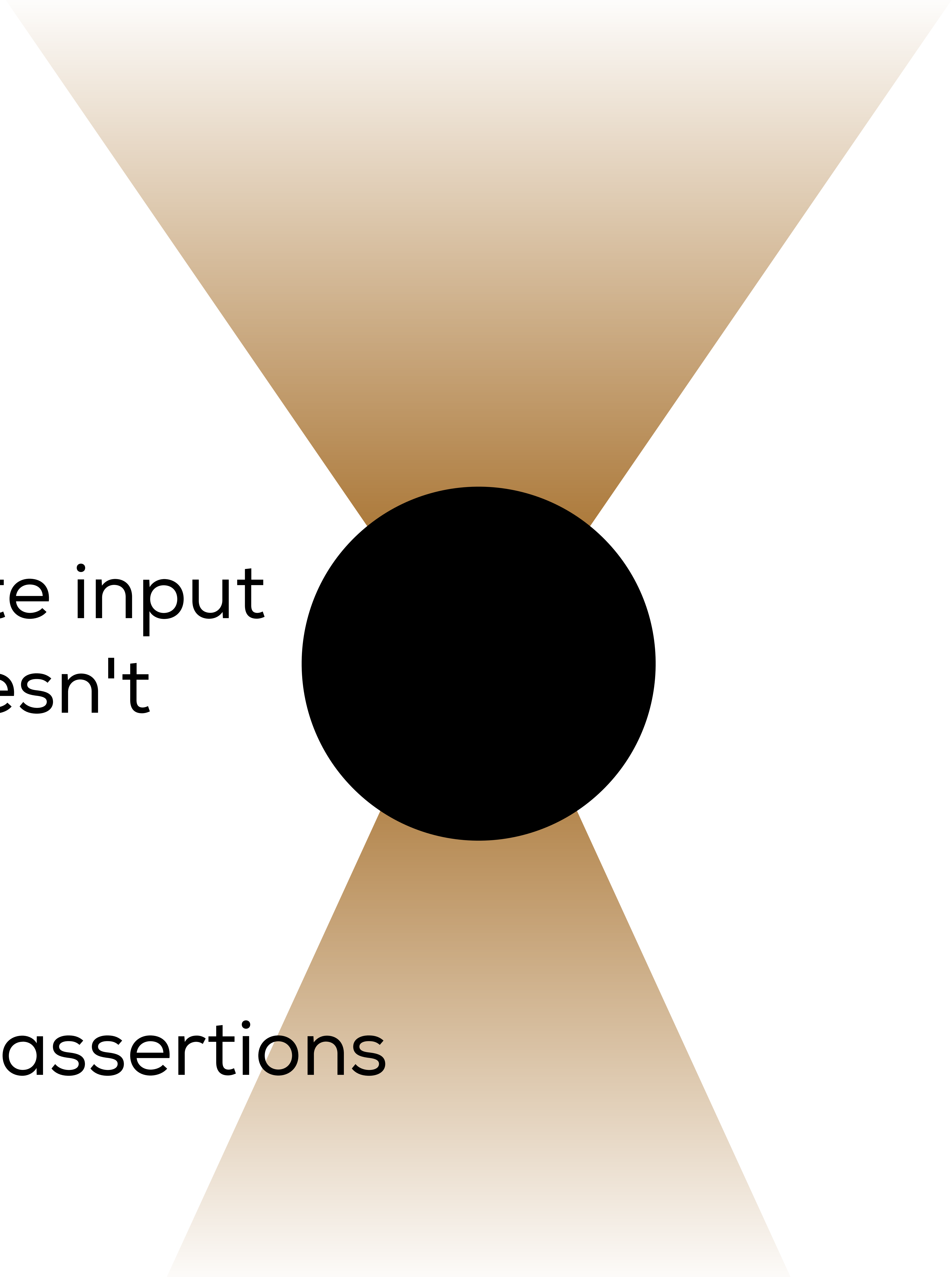
```
property_of {
  Generators.array_len.transform(->(search_count){
    search_events = CustomGenerators.
                    activity_record_for(:search).
                    sample_n(search_count)
    other_events = Generators.any_number_of(
            CustomGenerators.activity_record.
            reject(search_event?)).sample
    [search_events + other_events, search_count])
  }).sample
}.check do |          ,              |

  expect(
end
```



```
expected == actual
```

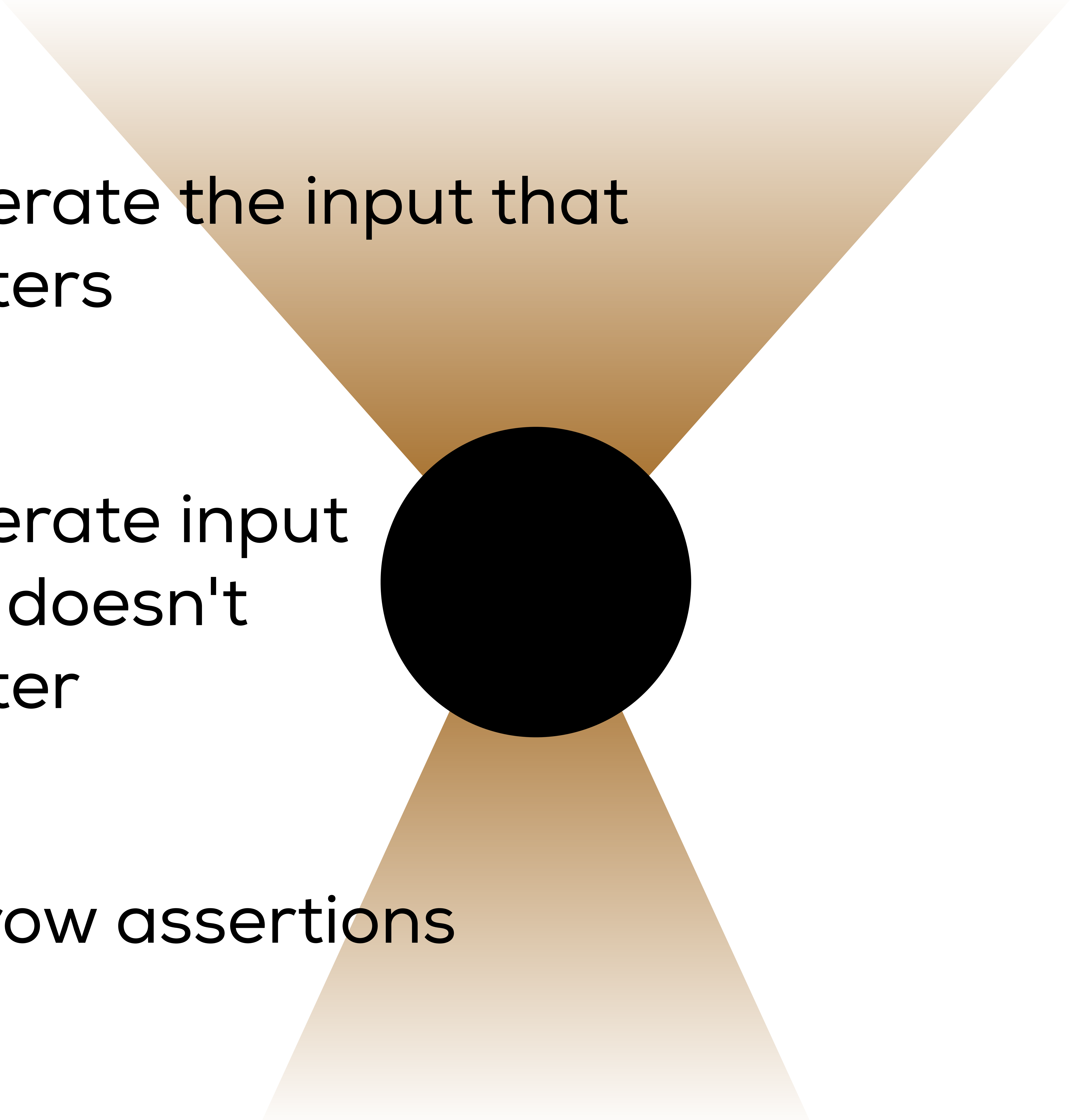


Narrow assertions



Generate input that doesn't matter

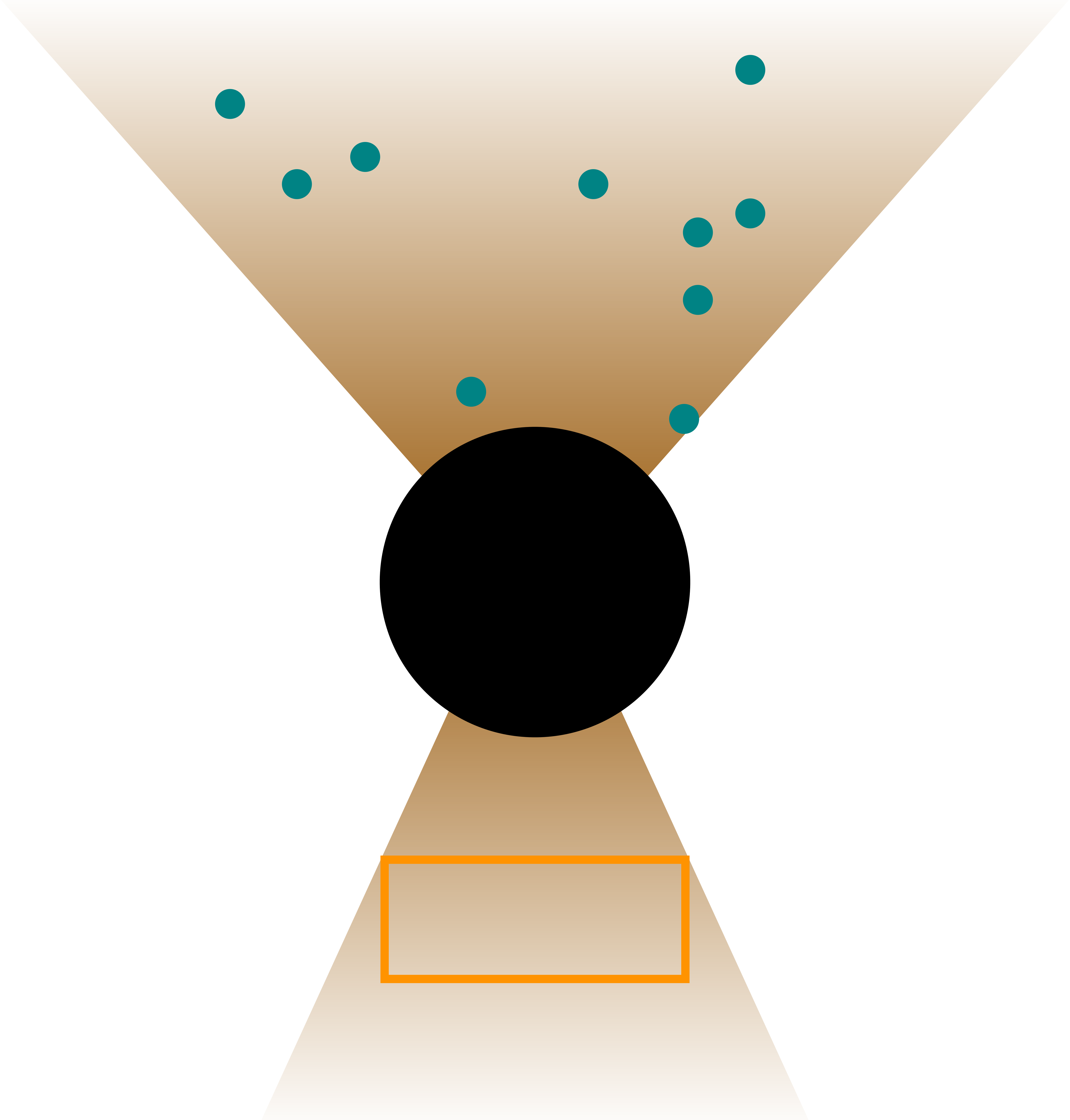
Narrow assertions

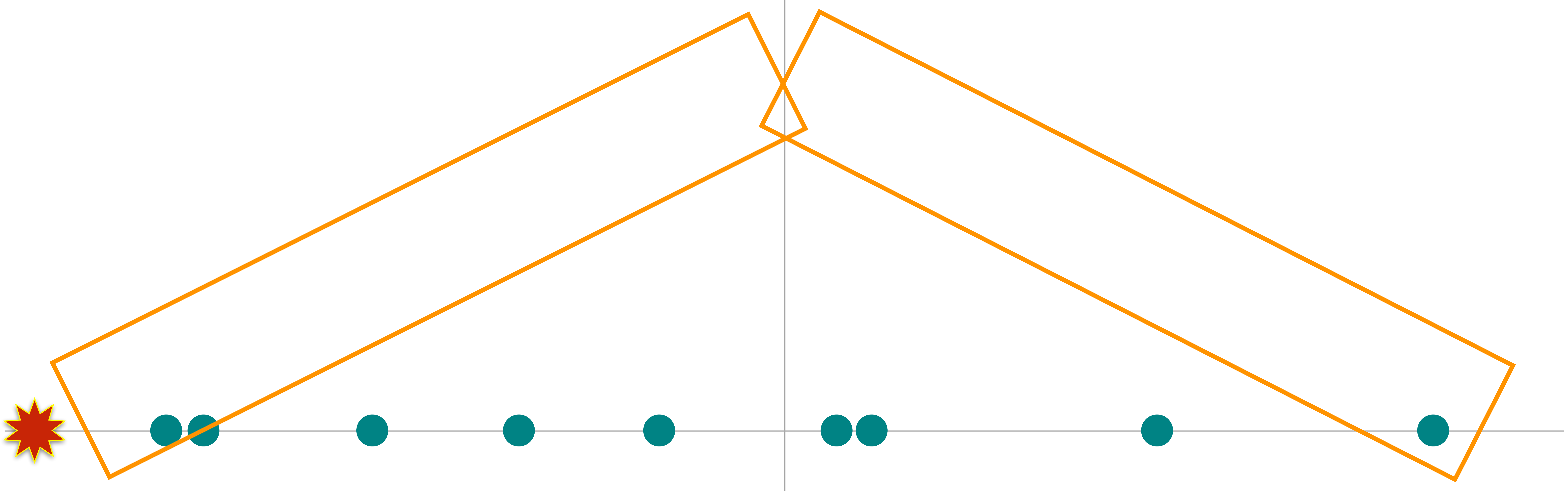


Generate the input that matters

Generate input that doesn't matter

Narrow assertions





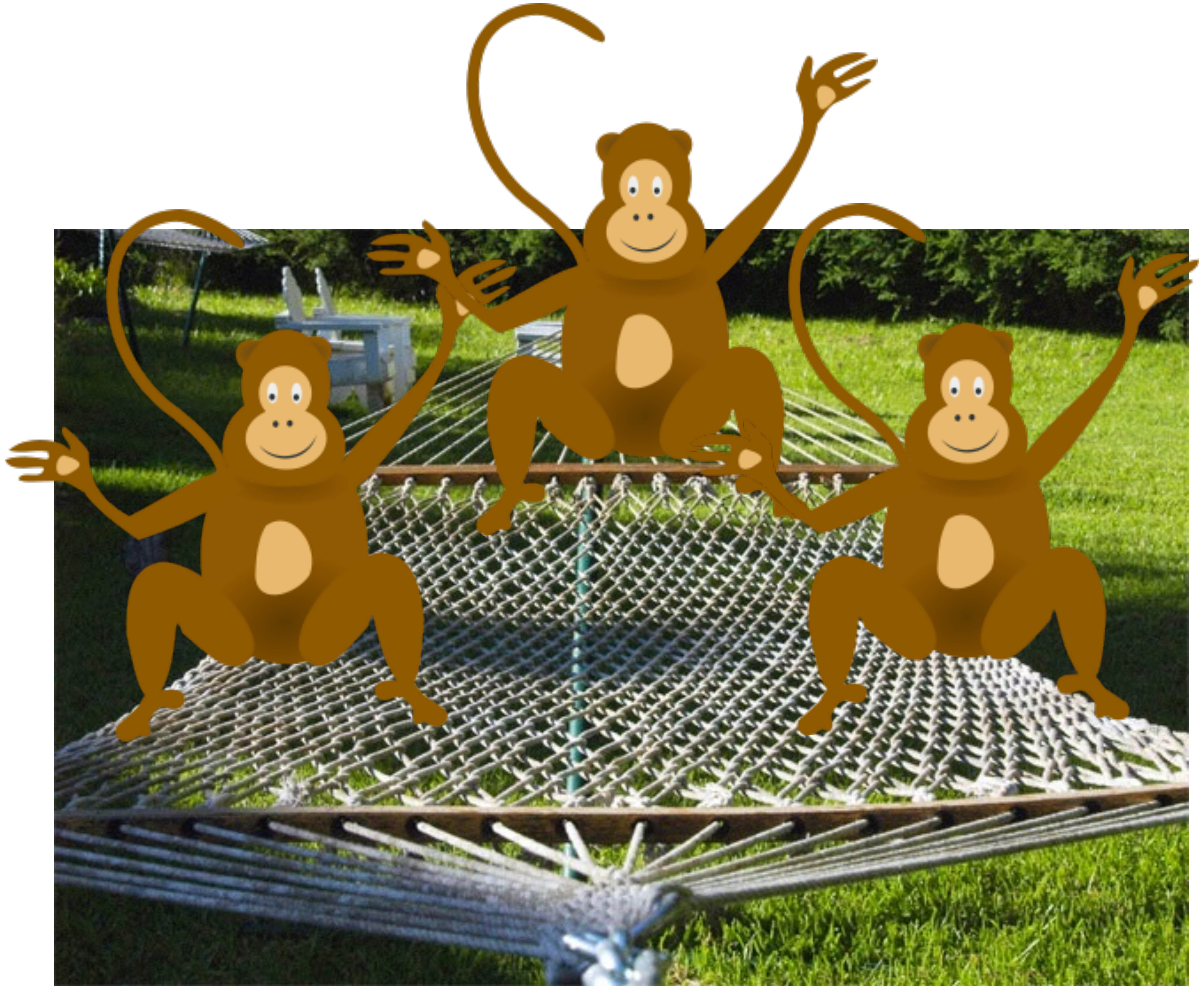
c

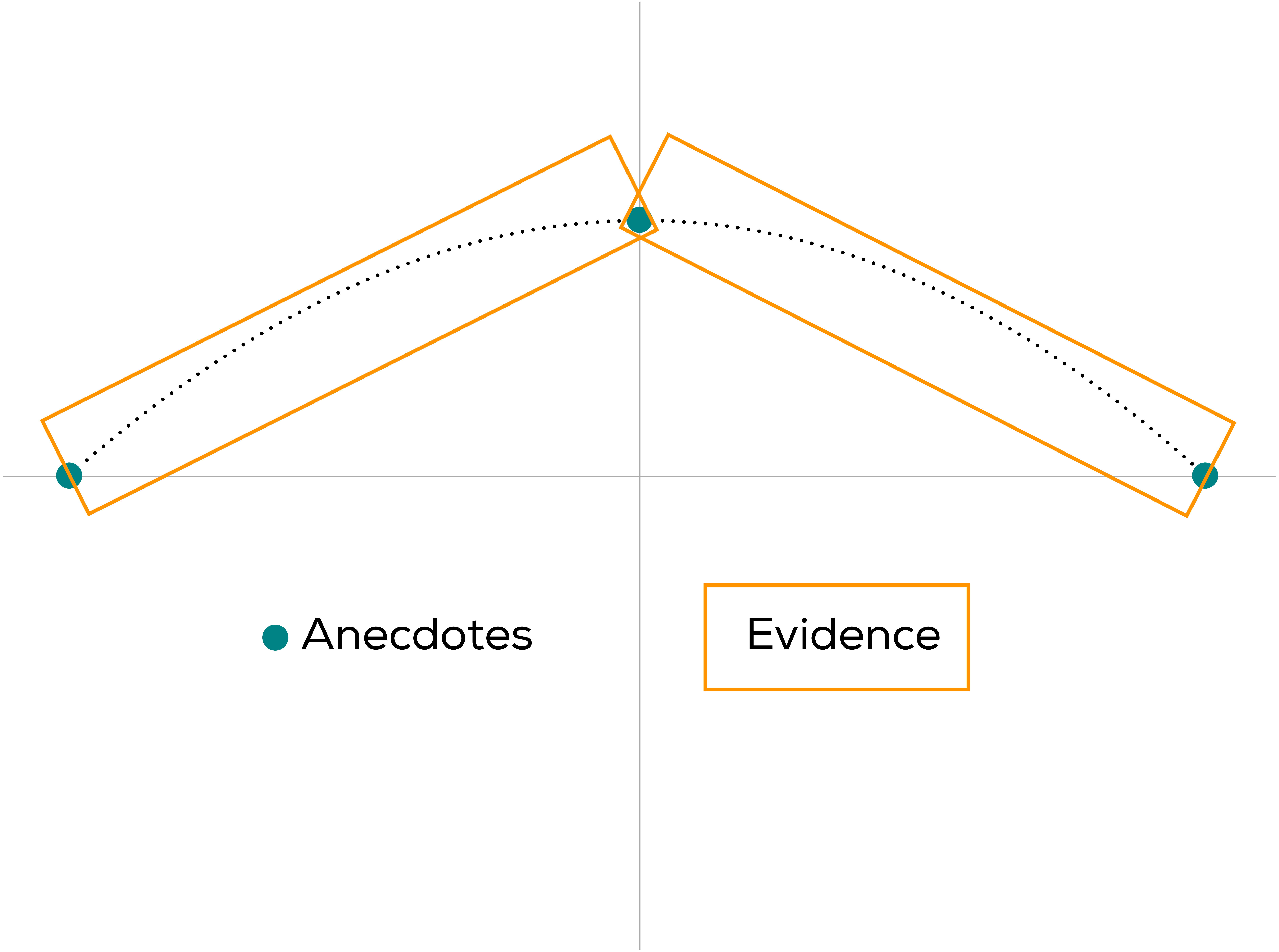
Fixing the test
fixes the mental model.

d

# Define Success.

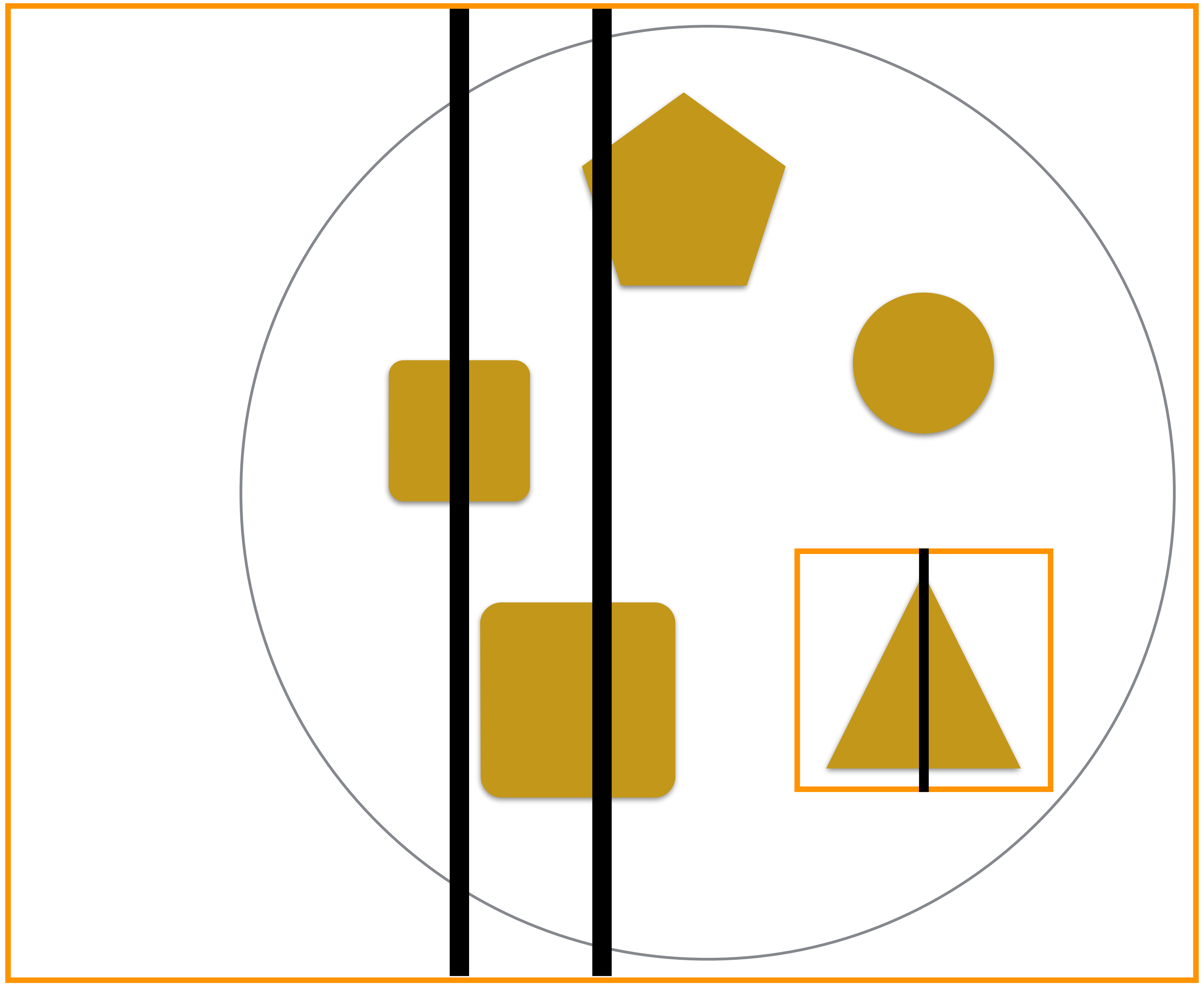
care + randomness =





● Anecdotes

Evidence

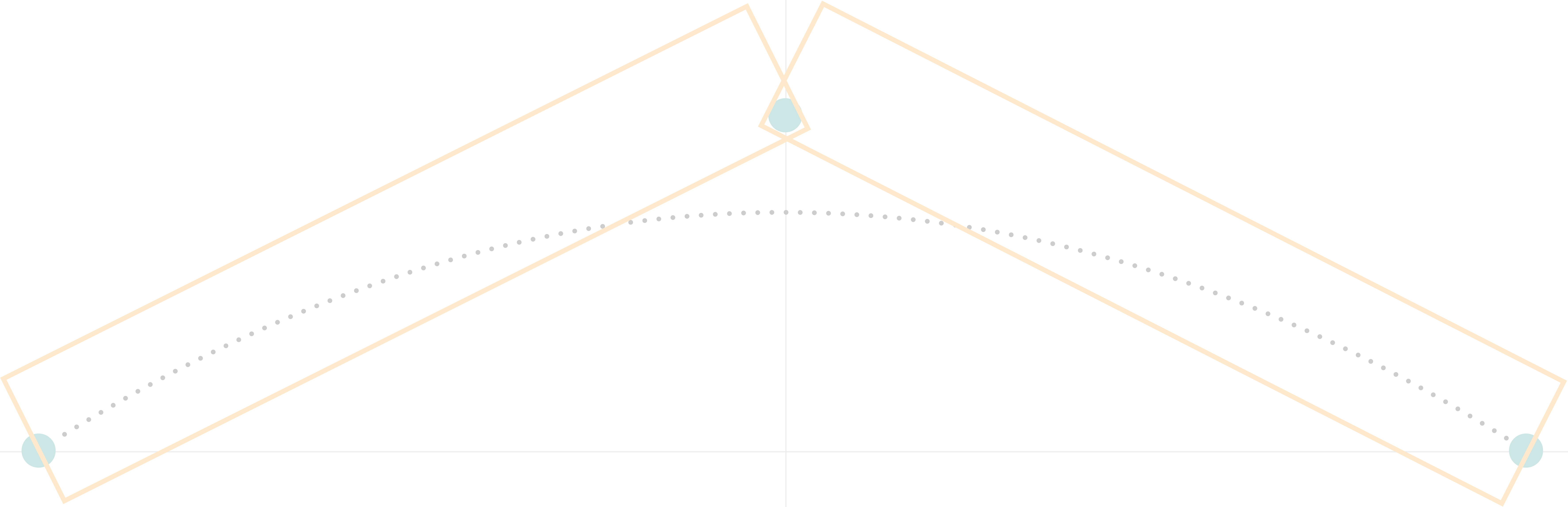


If you plod along
and just keep going,
you die on the vine.

*Dustin Updyke*

To think big,
you have to ignore low-hanging fruit.

*Maryan Mirzakhani*



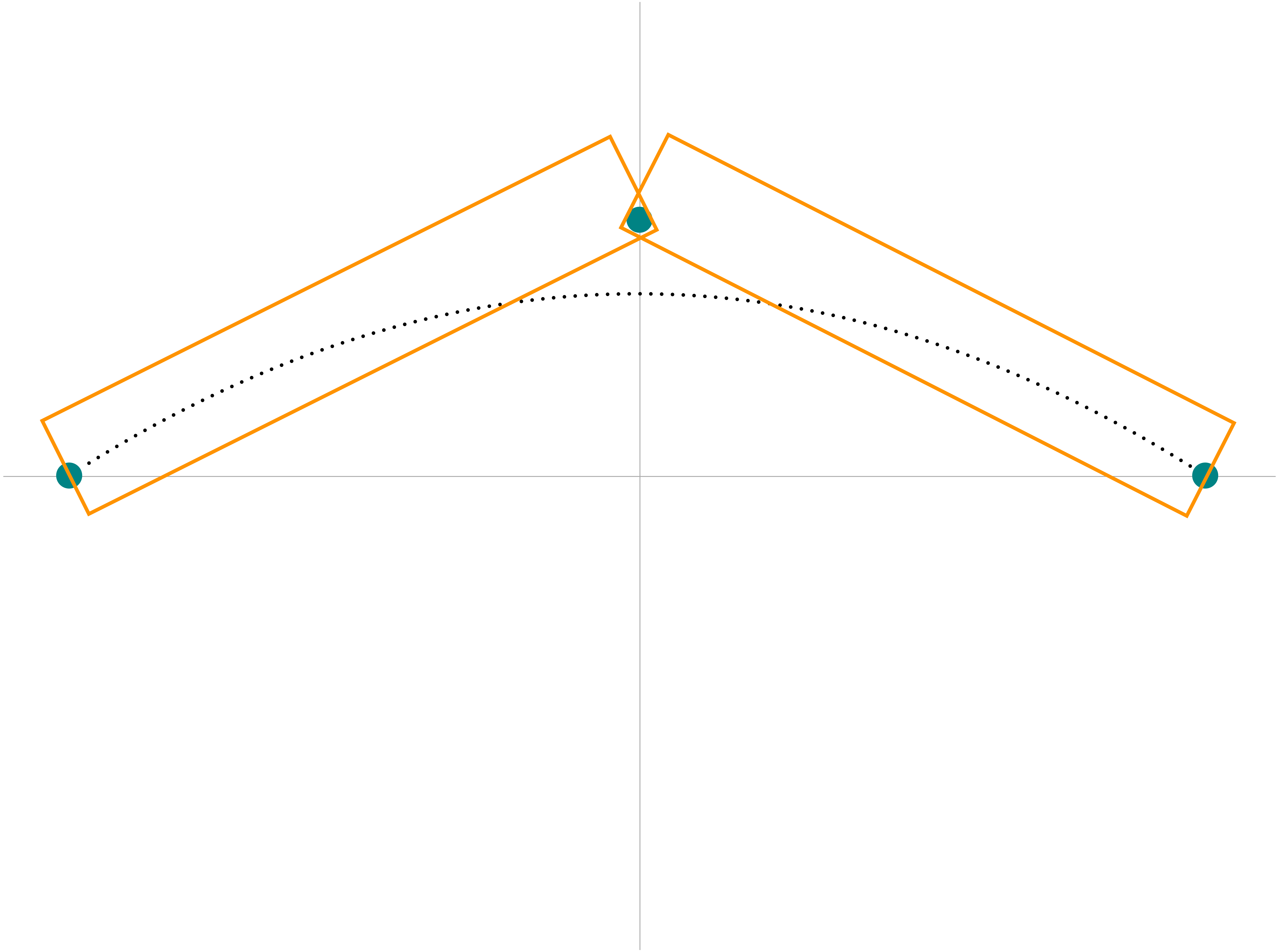
github.com/jessitron/generatron

**@jessitron**

Outpace™

What do we know?

How do we know it?



Moving to the outside is better because

- you can refactor
- you test the seams between classes
- you can test combinations of features

Property-based testing makes this easier because

- you can generate the combinations of features
- input is composed from smaller generators
- output is specified in a way more meaningful than a huge hard-coded structure

Properties:

- complete/incomplete/relational
- round trip
- backward