THE ENEMY OF THE STATE

Joy Clark

Joy Clark

Consultant @ innoQ

joy.clark@innoq.com

@iamjoyclark


innoQ
Beratung
Entwicklung
Konzeption
Training

state

noun

the particular condition that someone or something is in at a specific time.

So what is the problem with state?

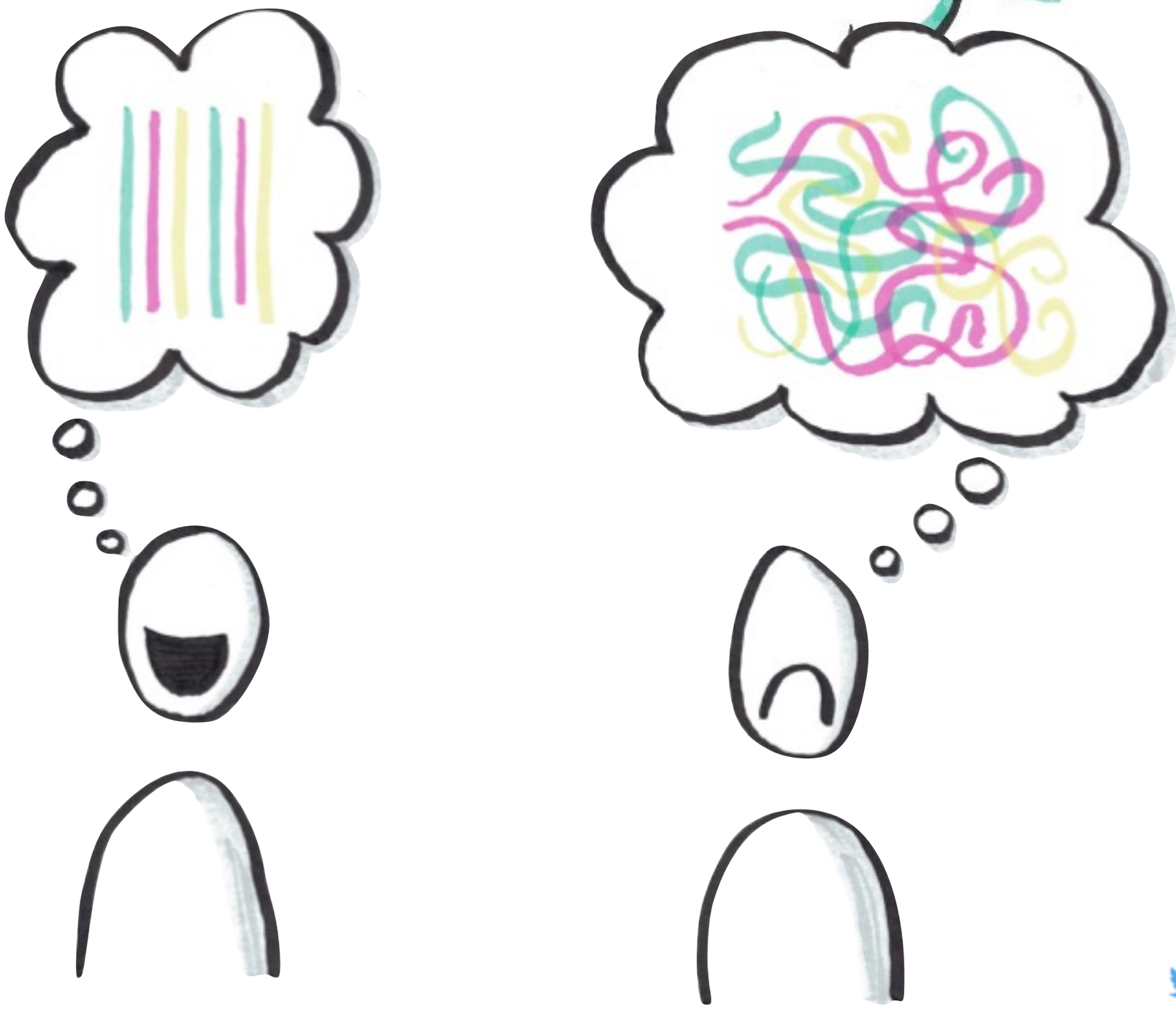
Problems with State*

1. complexity
2. concurrency
3. reproducibility

* Not guaranteed to be complete

@iamjoyclark

complexity



@iamjoyclark

ESSENTIAL
complexity

-VS-

INCIDENTAL
complexity

@iamjoyclark

some state is..

How can I identify essential state?

One Tool:

Domain Driven Design!

@iamjoyclark

DDD Building Blocks

Entities

- constant identity
- value can change over time

ValueObject

- immutable
- no lifecycle

Aggregates

- extra grouping
- transactional boundary

Entities and Aggregates represent the *essential state of the system*

Other state in an application is

state handling can be a source of

Technical

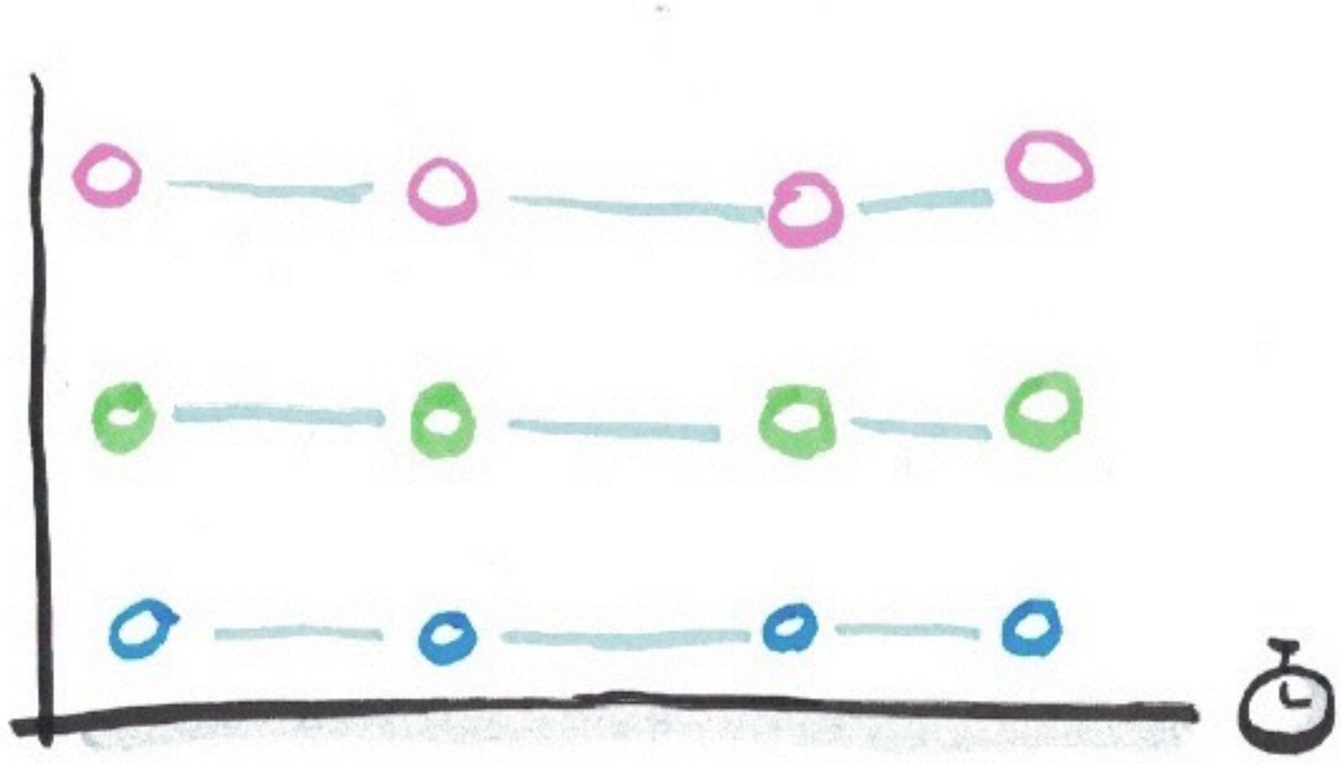
is

Debt

Why does state increase complexity?

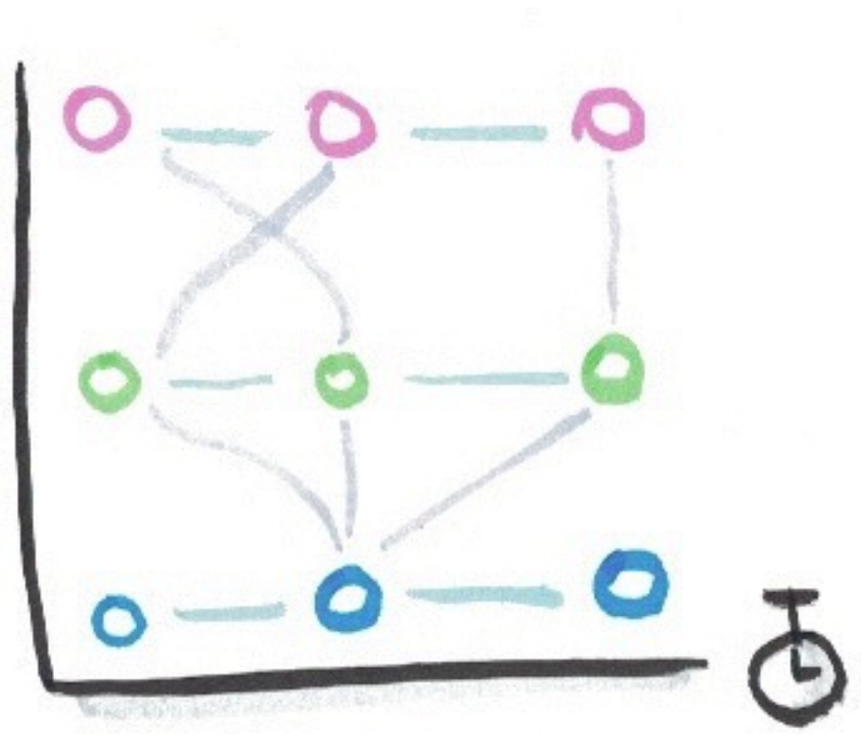
@iamjoyclark

State intertwines the value of an entity with time



@iamjoyclark

And if there are dependencies between stateful components...



What can we do about it?

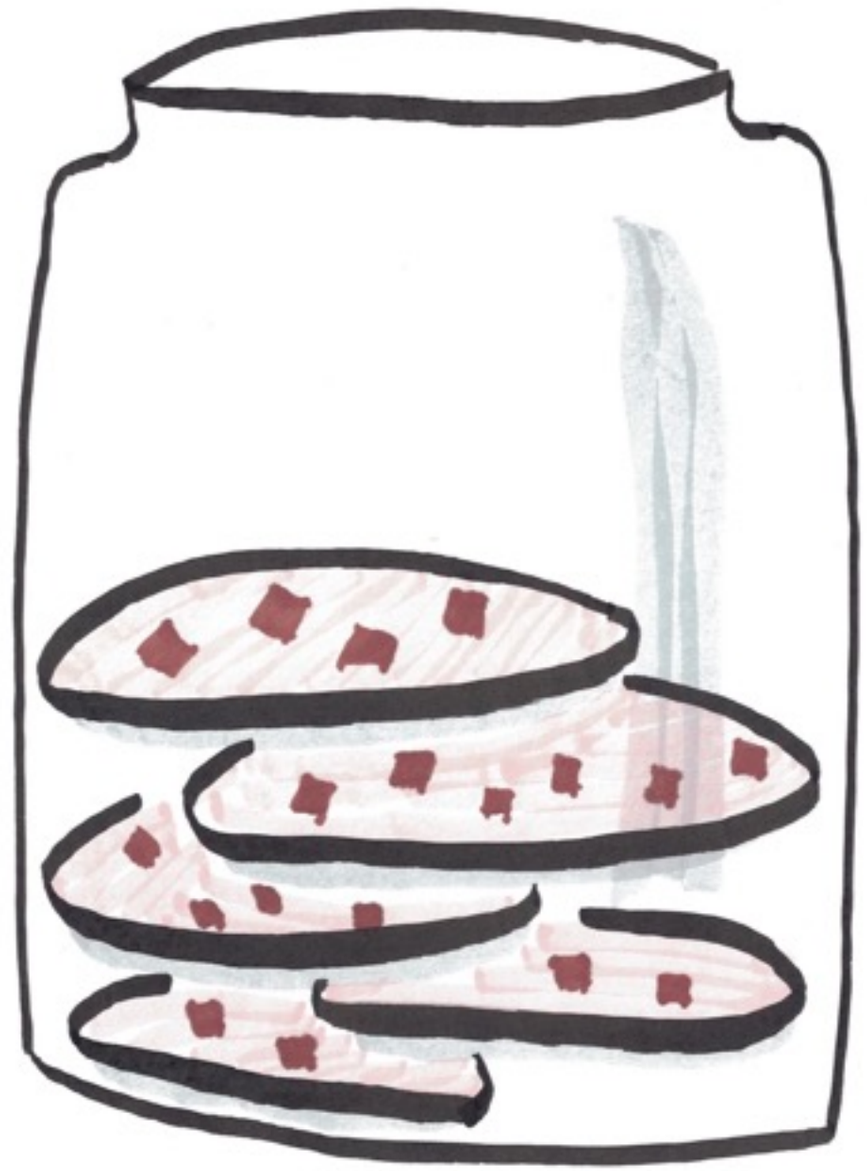
@iamjoyclark

ENCAPSULATION

How do different paradigms deal with state?

@iamjoyclark

An example entity...



Character #1

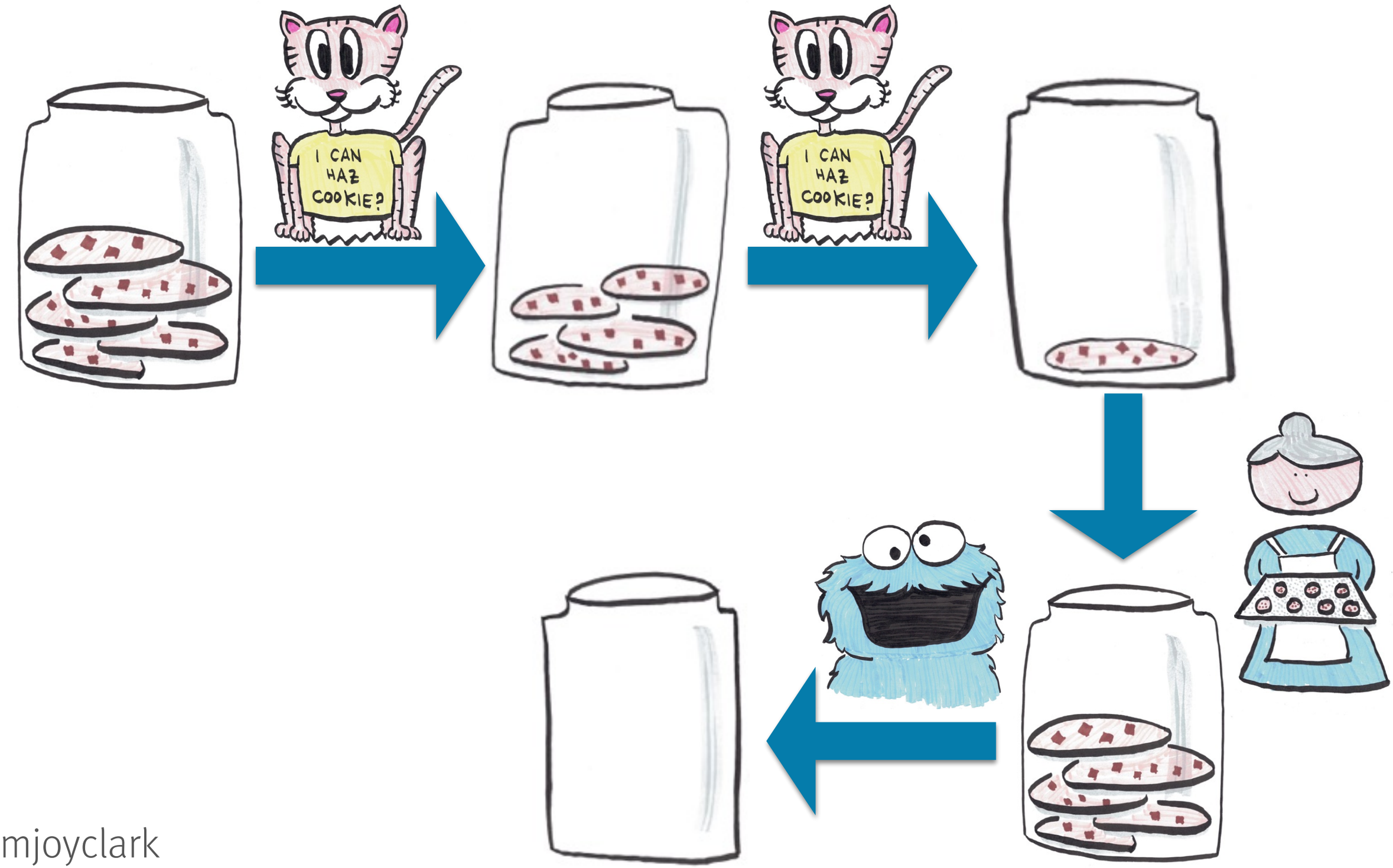
@iamjoyclark

Character #2

Character #3

@iamjoyclark

The value changes over time!



@iamjoyclark

Commodore 64 Basic

```
1 LET JAR = 0
10 DATA "GRANNY", "MONSTER", "GRANNY",
"CAT", "END"
20 READ A$
100 IF A$="GRANNY" THEN GOSUB 1010
200 IF A$="MONSTER" THEN GOSUB 2010
300 IF A$="CAT" THEN GOSUB 3020
400 IF A$="END" THEN GOTO 10000
500 GOTO 20
1010 LET JAR = JAR + 4
1015 PRINT "YOU LOOK SKINNY, GET SOME
COOKIES"
1020 RETURN
2010 IF JAR = 0 THEN PRINT "NO COOOKIE,
ME SAD"
2020 IF JAR > 0 THEN PRINT "OM NOM NOM"
2030 LET JAR = 0
2040 RETURN
3020 PRINT "MEOW"
3030 LET JAR = JAR - 1
3040 RETURN
10000 END
READY.
```

Jens Bendisposto

https://twitter.com/jbendisposto/status/904662544245297152

Object-Oriented

```
class CookieJar
  def initialize(initial_count)
    @count = initial_count
  end

  def remove_cookies(amount)
    if amount > @count
      @count = 0
    else
      @count -= amount
    end
  end

  def add_cookies(amount)
    @count += amount
  end
end
```

Lucas Dohmen

https://gist.github.com/moonglum/f763eb257a4ede22e3008d8612255350

Functional (with MONADS!)

```
takeCookies n jar = if n > jar
  then 0
  else jar - n

cat = modify (takeCookies 1)
grandma = modify (+10)
cookieMonster = put 0

workflow = do
  cat
  cookieMonster
  grandma
  cookieMonster

run = execState workflow 5
```

Martin Kühl

https://gist.github.com/joyclark/843932d96dbf99ab8438cb9de485d10f

Functional

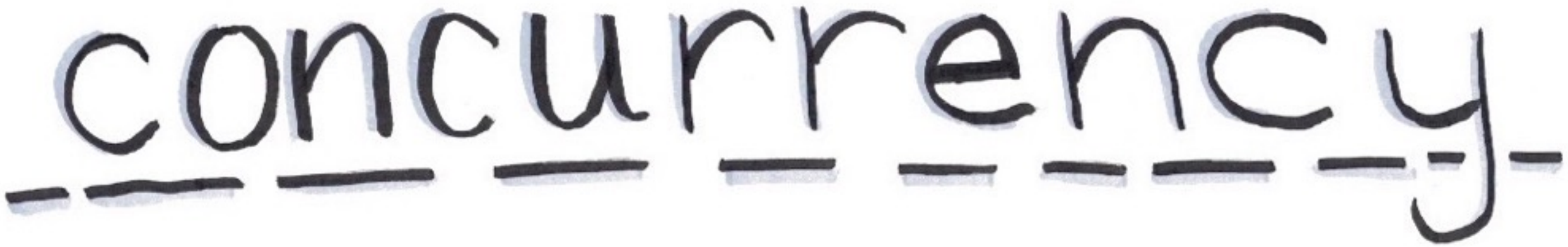
```clojure
(defn take [{cookies ::cookies} to-take]
  (if (< to-take cookies)
    {::cookies (- cookies to-take)}
    {::cookies 0}))

(defn cookie-monster [cookie-jar to-eat]
  (take cookie-jar to-eat))

(defn kitten [cookie-jar]
  (take cookie-jar 1))

(defn grandma [{cookies ::cookies} nr-baked]
  (if (pos? nr-baked)
    {::cookies (+ cookies nr-baked)}
    {::cookies cookies}))
```

Joy Clark

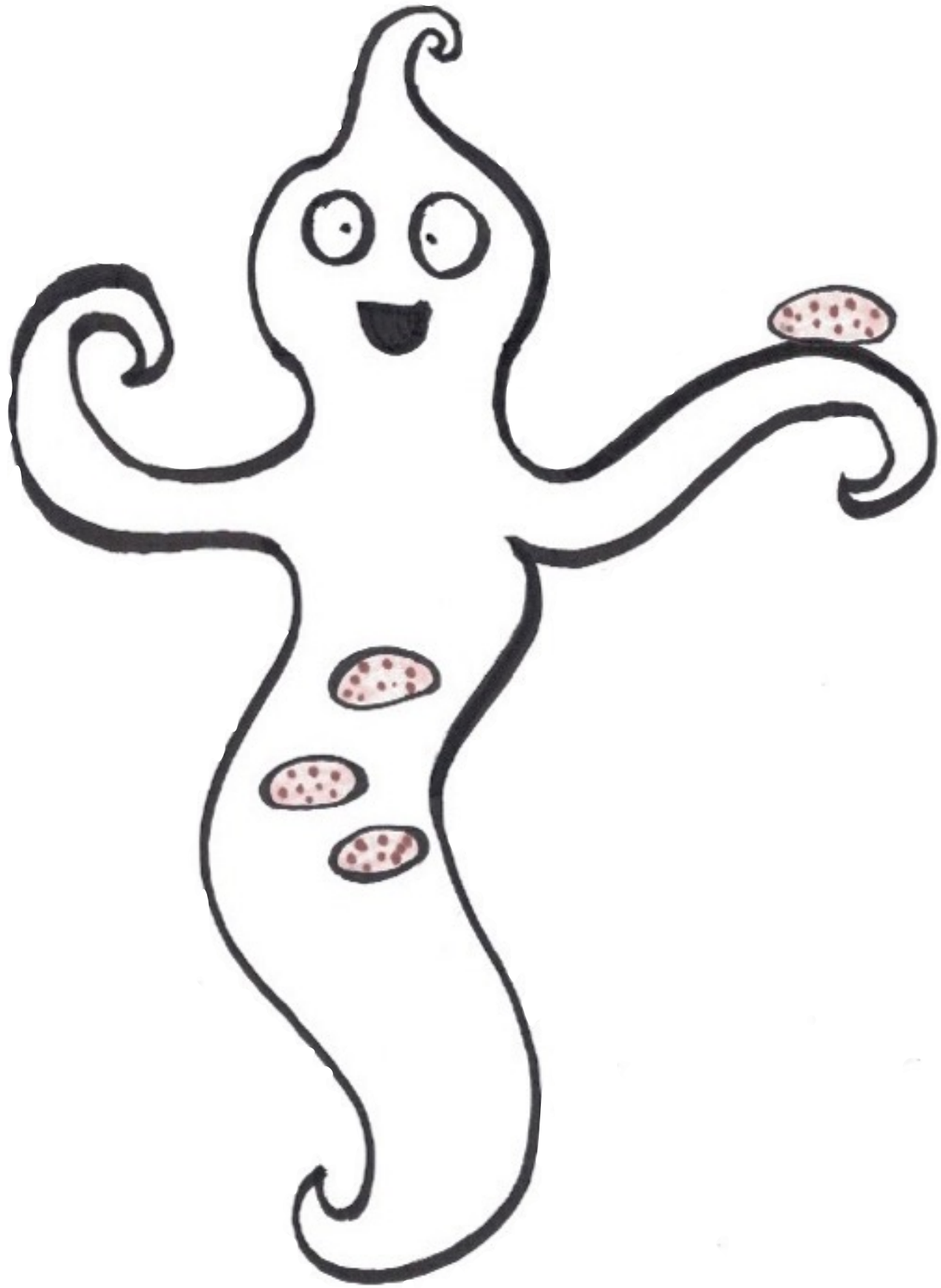
https://github.com/joyclark/cookie-jar

well that wasn't

that bad...



@iamjoyclark



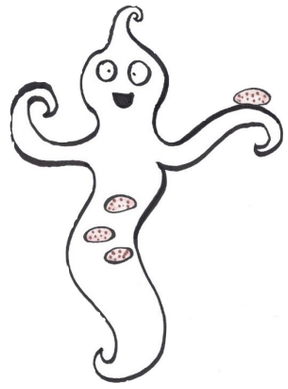
We are not alone! A friendly ghost with an appetite for cookies has started haunting the kitchen. And if the ghost reaches into the cookie jar at the same time as someone else...

@iamjoyclark


Boom!

In the kitchen...

```
cookie_jar = CookieJar.new(5)

cookie_jar.remove_cookies(3)

                ⋮

amount > @count => false

                ⋮

@count -= amount

                ⋮

cookie_jar.count_cookies => -2
```



In an otherworldly dimension...

```
cookie_jar.remove_cookies(4)

                ⋮

amount > @count => false

                ⋮

@count -= amount
```

@iamjoyclark

What do we do?

@iamjoyclark

Object-Oriented: Locks

```
public IList<Cookie> TakeCookies(int aNumberOfCookies)
{
    var cookies = new List<Cookie>();

    lock (_jarLid)
    {
        // take cookies out!!
    }

    return cookies;
}
```

Benjamin Wolf

https://github.com/programming-wolf/CookieJar

Object-Oriented: Locks

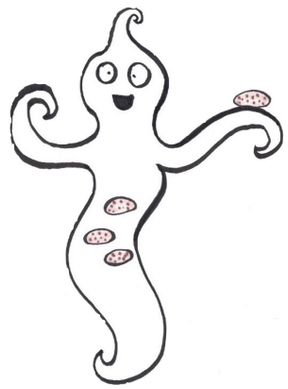
```
class CookieJar
  @@jarLid = Mutex.new

  def remove_cookies(amount)
    @@jarLid.synchronize do
      if amount > @count
        @count = 0
      else
        @count -= amount
      end
    end
  end
end
```

Lucas Dohmen

https://gist.github.com/moonglum/f763eb257a4ede22e3008d8612255350

In the kitchen...

```
cookie_jar = CookieJar.new(5)
cookie_jar.remove_cookies(3)


## The lid is off for kitten!
amount > @count => false
@count -= amount


cookie_jar.count_cookies => 2
```



In an otherworldly dimension...

```
cookie_jar.remove_cookies(4)


## The lid is off for kitten!
## We'll wait until it's back on


## The lid is off for ghost!
amount > @count => false
@count = 0

cookie_jar.count_cookies => 0
```

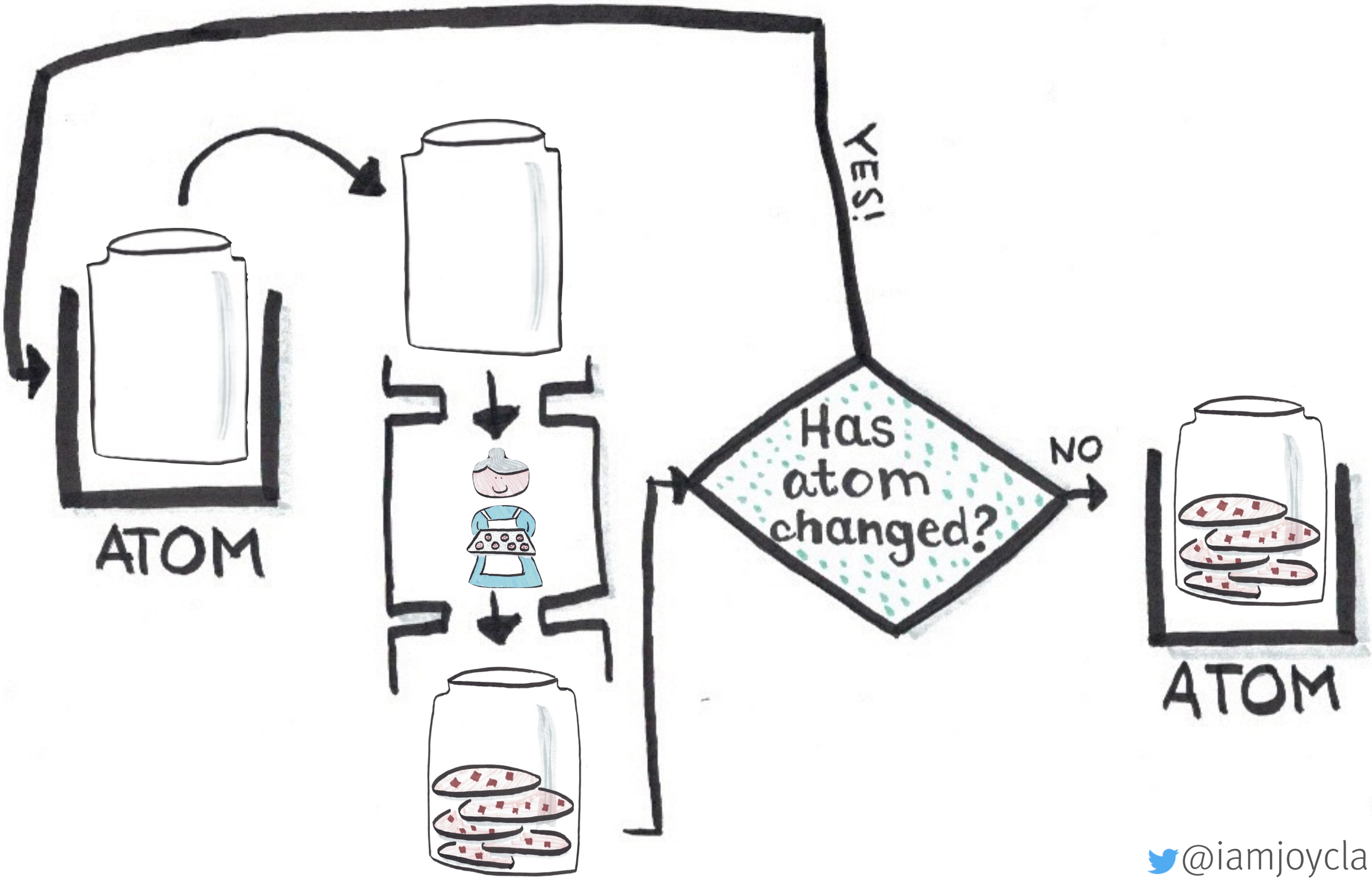
Functional

```
(def cookie-jar (atom {::cookies 5}))

(defn ghost [cookie-jar]
  (take cookie-jar 2))

(swap! cookie-jar grandma 5)
(swap! cookie-jar kitten)
(swap! cookie-jar ghost)

@cookie-jar // => 7: retrieves current value of cookie-jar
```

Joy Clark

https://github.com/joyclark/cookie-jar

swap! explained



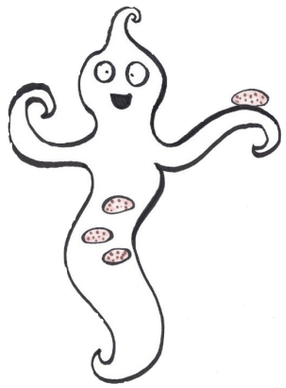
@iamjoyclark

In the kitchen...

```
(def cookie-jar
  (atom {::cookies 5}))

(swap! cookie-jar grandma 5)
```

```
swap!
 retrieve {::cookies 5}
 (grandma {::cookies 5} 5)
 save! {::cookies 10}
```

@cookie-jar => {::cookies 8}



In an otherworldly dimension...

```
(swap! cookie-jar ghost)
```

```
swap!
 retrieve {::cookies 5}
 (ghost {::cookies 5})
 atom changed, retry!

 retrieve {::cookies 10}
 (ghost {::cookies 10})
 save! {::cookies 8}
```

concurrency

is really hard.

But it's something we *absolutely have to consider!*

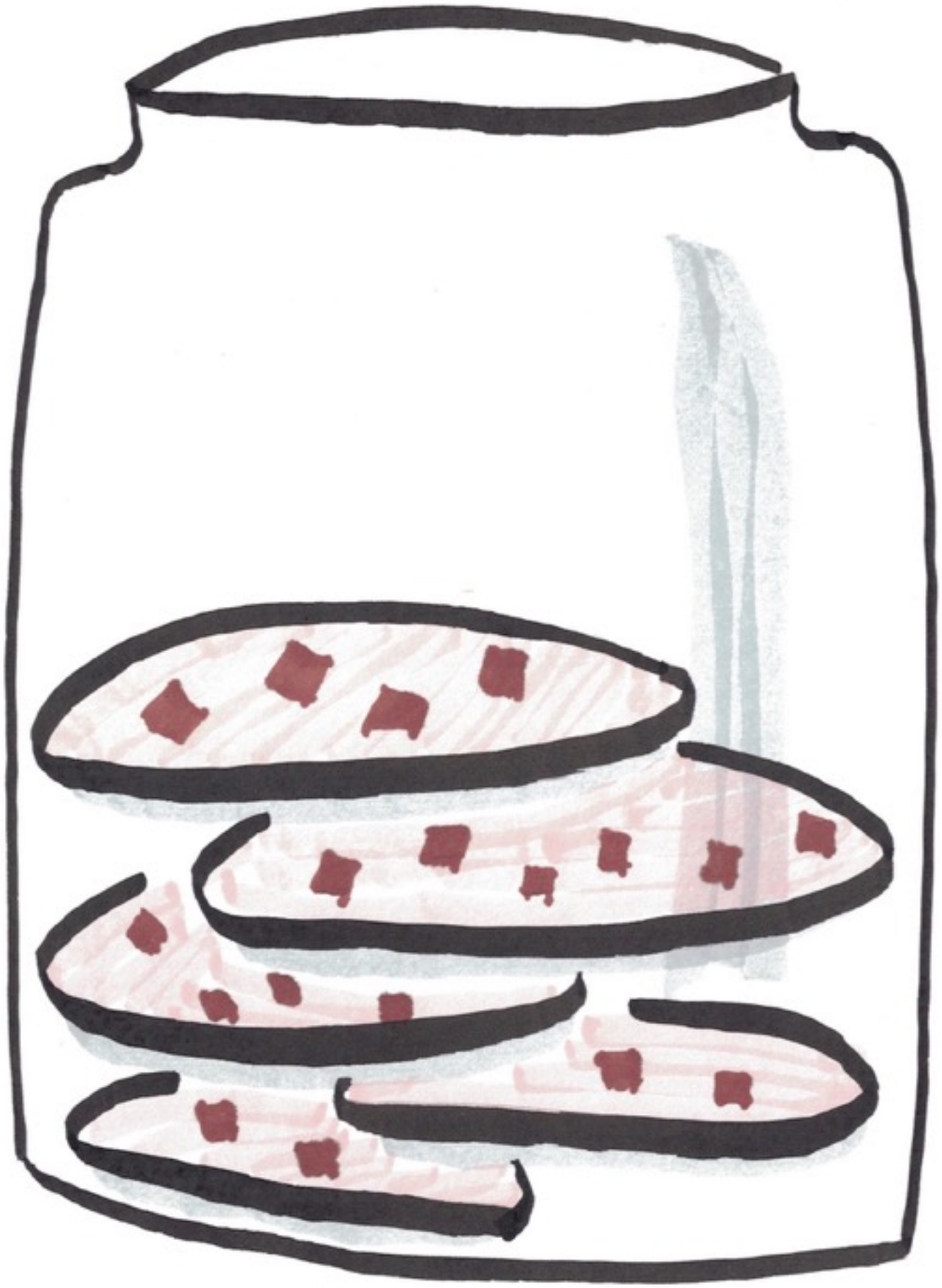
@iamjoyclark

but that’s it, right?

...

reproducibility

@iamjoyclark



@iamjoyclark

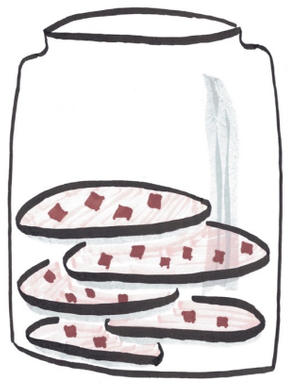
Is there a way to reproduce the chain of events?

@iamjoyclark

Event Sourcing

- The state of the application is determined by a sequence of business events.
- Reproducibility of the states

Event Sourcing Cookie Jar

created	1 eaten	4 eaten	10 baked	6 eaten
				

current state

@iamjoyclark

reproducibility_

can be achieved by

persisting the changes

that have been made in

our system

@iamjoyclark

In summary...

How to Attack complexity

Think about the problem you are trying to solve.

Identify essential state and encapsulate it.

Eliminate unnecessary state.

How to Attack concurrency

Use locking and synchronized blocks.

Prefer immutable values which are thread safe.

Look into what that your language offers for managing concurrency

@iamjoyclark

How to Attack reproducibility

Derive the state from the events or transactions which have taken place

Make sure that you are logging all necessary information

More Cookie Jar Examples

The Implementations

- Clojure (this repo)
- C#
- Haskell
- Elixir
- Scala
- Prolog
- Commodore 64 Basic
- Java
- Ruby
- Groovy

Do you have an example? Let me know!

https://github.com/joyclark/cookie-jar

Managing state is hard...

...so let's get started!

https://github.com/joyclark/cookie-jar

Joy Clark | @iamjoyclark | joy.clark@innoq.com

www.innoq.com