zolodeck

Domain Driven Design with Clojure

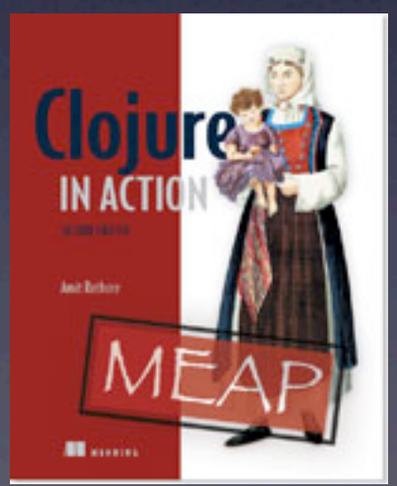
Clojure/West 2013
Portland, OR

zolodeck

Amit Rathore

CEO, Zolo Labs
CTO, Runa
Lead, ThoughtWorks

Author, Clojure in Action



2e

zolodeck

zolodeck

our first product

zolodeck

zolodeck

our first product
your digital assistant

zolodeck

zolodeck

our first product
your digital assistant
helps with professional networking and relationships

zolodeck

zolodeck

our first product
your digital assistant
helps with professional networking and relationships
(and personal ones too!)

zolodeck

zolodeck.com

zolodeck

DDD

zolodeck

not **really** going to talk about DDD

zolodeck

context map
bounded contexts
shared kernel
...

zolodeck

read the book

(eric evans)

zolodeck

today's talk

zolodeck

how to organize larger applications

zolodeck

modularity, maintainability, **agility**

zolodeck

way of thinking

zolodeck

business domain is the focus

zolodeck

I. Domain

zolodeck

1. Domain
2. Driven

zolodeck

1. Domain
2. Driven
3. Design

zolodeck

I. Domain

zolodeck

layers

zolodeck

Presentation

Transport

Domain

Persistence

Data storage

zolodeck

other stuff

zolodeck

APIs
various clients
batch-processing
real-time processing
“offline” mode
...

zolodeck

Presentation

Transport

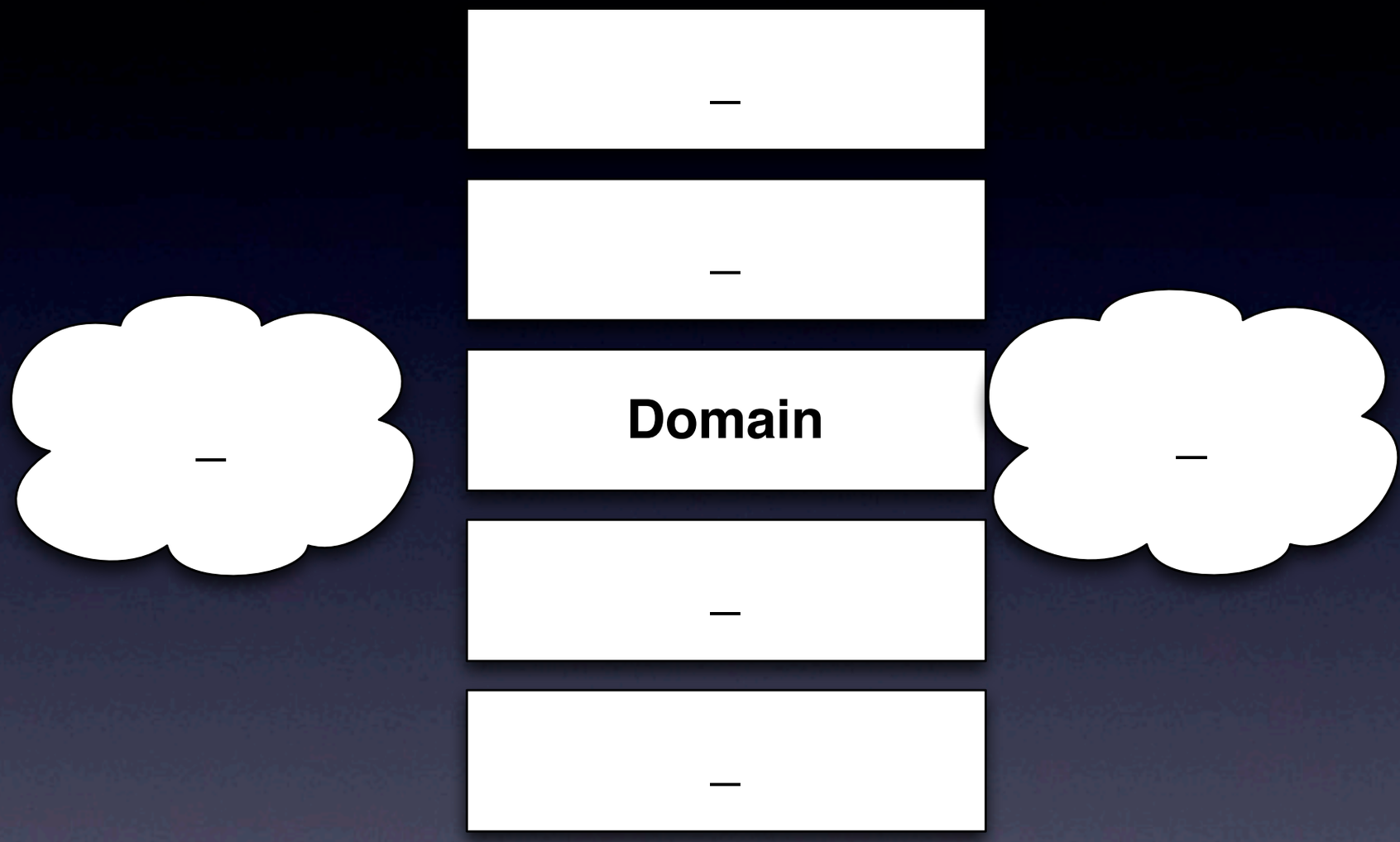
Other

Domain

stuff

Persistence

Data storage


Domain

zolodeck

domain

zolodeck

subject area

zolodeck

differentiation

zolodeck

expertise

zolodeck

domain model

zolodeck

model

zolodeck

conceptualization

zolodeck

limited simulation of the real world

zolodeck

selected abstractions

zolodeck

basis

zolodeck

solve particular problems

zolodeck

business logic

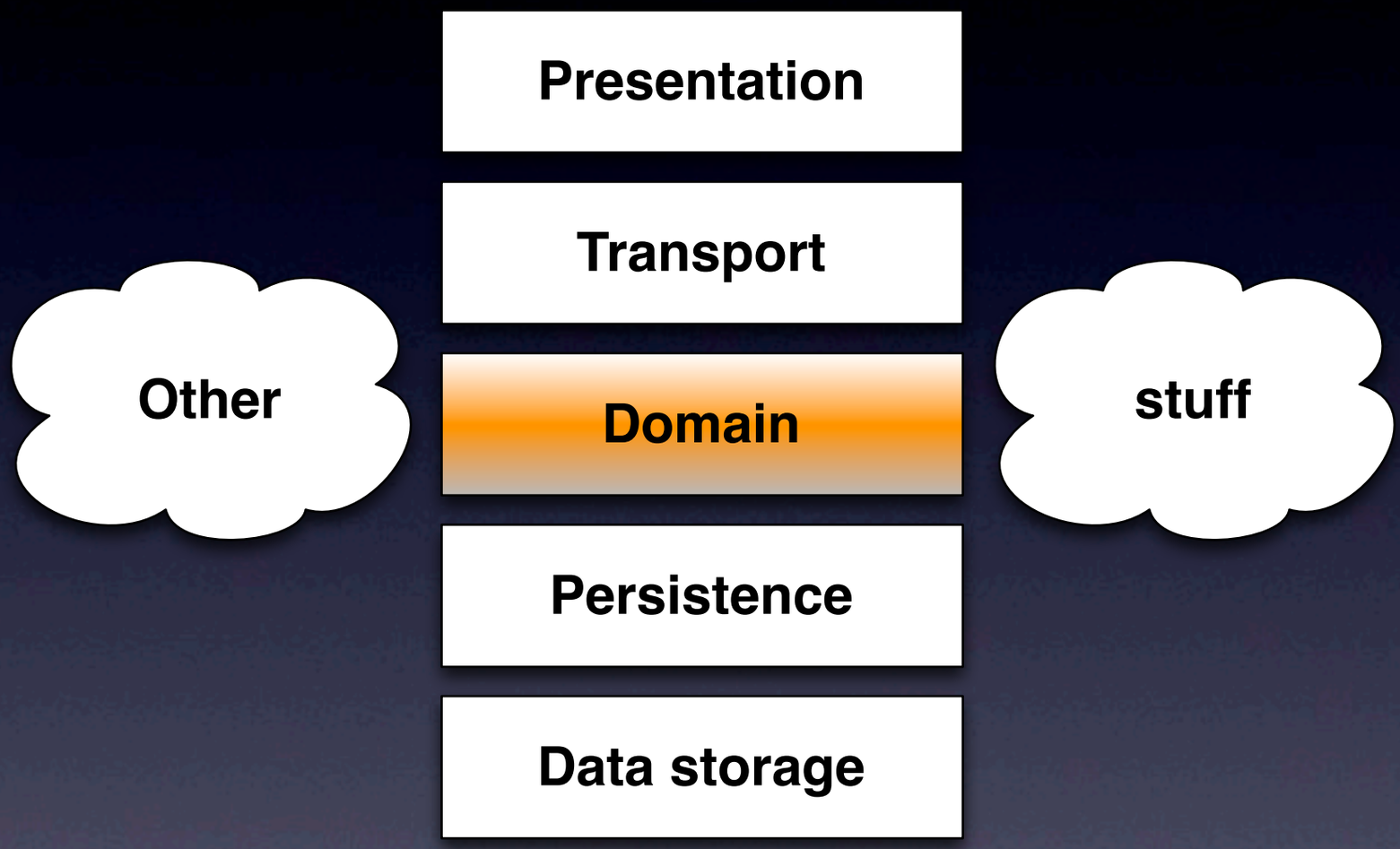
zolodeck

quality of the model

zolodeck

determines success

zolodeck


Presentation
Transport
Other
Domain
stuff
Persistence
Data storage

zolodeck

what does it look like?

zolodeck

OO world

zolodeck

nouns

zolodeck

processes are owned by nouns

zolodeck

designing the domain model

zolodeck

the right nouns own the right processes

zolodeck

nouns = classes

zolodeck

processes = methods

zolodeck

and of course, OOP has issues

zolodeck

Actually I made up the term "object-oriented",
and I can tell you I did **not** have C++ in **mind**.

Alan Kay

zolodeck

"dude, not everything is an object"

zolodeck

dude, not everything is an object

(the kingdom of nouns)

zolodeck

OOP has a real modeling handicap

zolodeck

Clojure

zolodeck

functional programming

zolodeck

verbs

zolodeck

equal prominence

zolodeck

verbs = functions

zolodeck

nouns = data

zolodeck

verbs transform nouns

zolodeck

verbs transform nouns

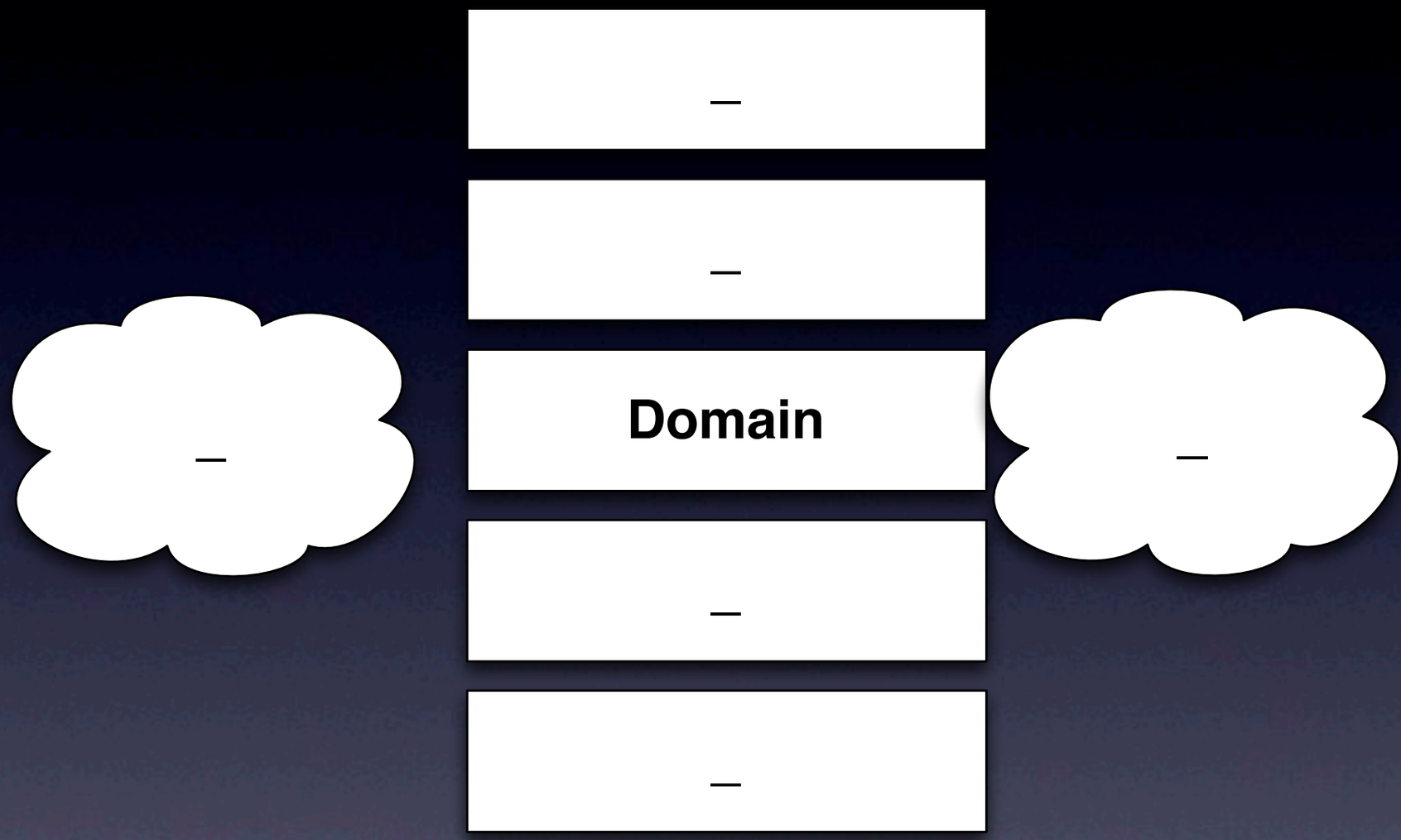
functions transform data

zolodeck

domain model = functions that transform data

zolodeck

specifically, those that live...



... here

zolodeck

purity matters

zolodeck

no side-effects

zolodeck

no dependency on data storage

zolodeck

no dependency on external services

zolodeck

in other words...

zolodeck

straight Clojure functions and data-structures

zolodeck

clean data passed to domain functions

zolodeck

validate outside the boundary

zolodeck

simple conditional code

zolodeck

missing data elements...

zolodeck

... or checking basic types of data elements

zolodeck

more complex validations...

zolodeck

validateur, bouncer, clj-schema

zolodeck

canonical data

zolodeck

domain model deals with canonical representation

zolodeck

domain-specific, semantic checks

zolodeck

thin layer, inside domain model

zolodeck

invalid or illegal states...

zolodeck

throw RuntimeException

zolodeck

slingshot, ex-info

zolodeck

uniform error handling, outside the domain layer

zolodeck

Data instantiation

Generic validation

Domain-Specific Validation

Domain Model

Domain

zolodeck

all these functions...

zolodeck

organized as collections of namespaces

zolodeck

zolo.domain.*

zolodeck

zolo.domain.user
zolo.domain.contact
zolo.domain.message
zolo.domain.interaction
zolo.domain.score
...

zolodeck

hmmm...

zolodeck

look a lot like Class names...

zolodeck

zolo.gateway.*

zolodeck

zolo.gateway.facebook
zolo.gateway.linkedin
zolo.gateway.twitter
zolo.gateway.email
...

zolodeck

other namespace collections...

zolodeck

zolo.api.*

zolodeck

zolo.api.user
zolo.api.contact
zolo.api.stats
zolo.api.recommendations
zolo.api.reminders
...

zolodeck

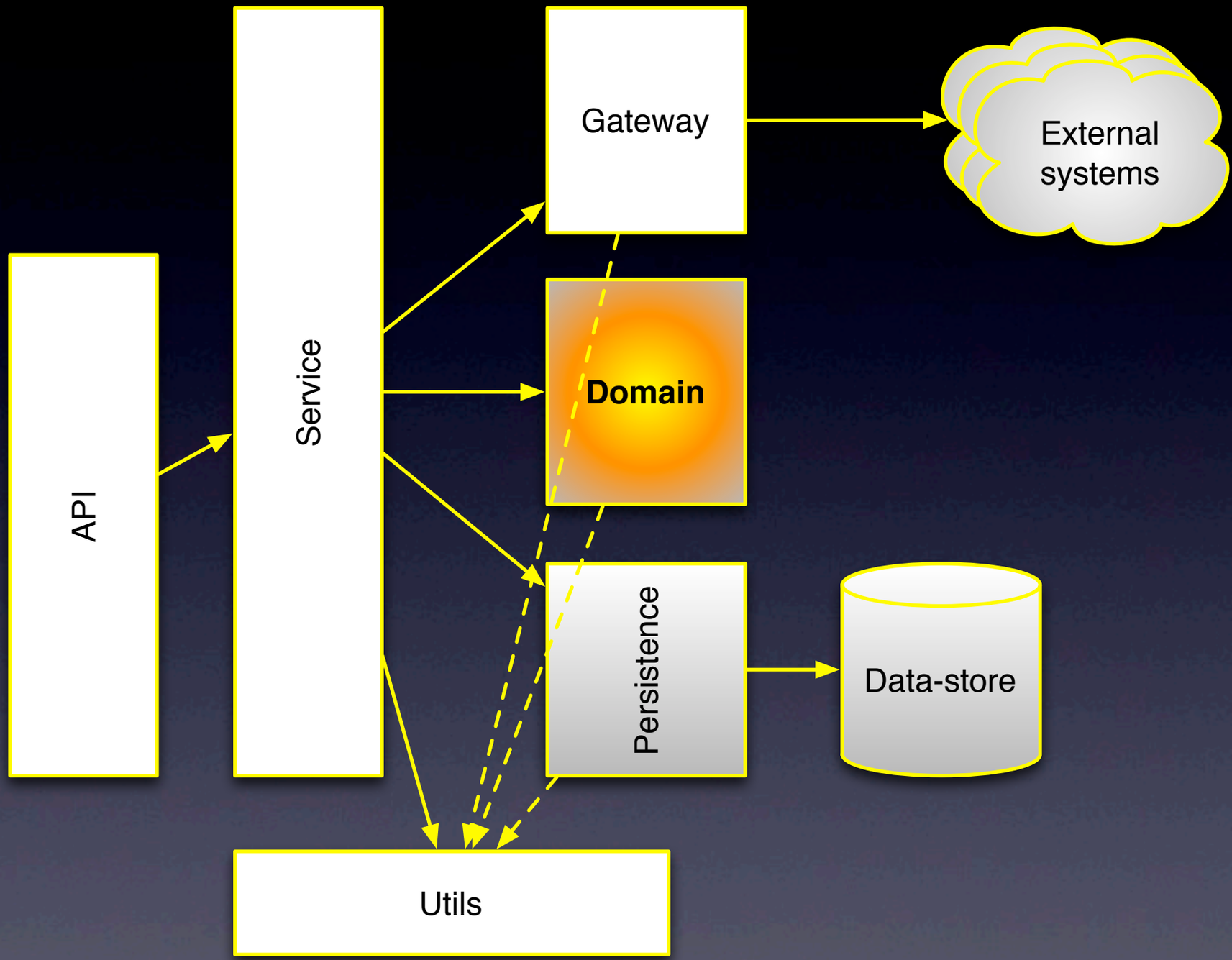
zolo.service.*

zolo.demonic.*

(datomic)

zolodeck

how things fit


API
Service
Gateway
External systems
Domain
Persistence
Data-store
Utils

zolodeck

utils

zolodeck

zolo.utils.calendar
zolo.utils.string
zolo.utils.maps
...

zolodeck

service

zolodeck

verbs that don't belong in the domain, per se

zolodeck

verbs that don't belong in the domain, per se

GRASP - service pattern

zolodeck

zolo.service.user-service
zolo.service.contact-service
zolo.service.reminder-service
...

zolodeck

api

zolodeck

fns for RESTful access

zolodeck

domain

zolodeck

examples of domain functions

zolodeck

score relationship strength

merge contact's social identities

remind user to follow up

suggest contacts to connect with

...

zolodeck

hmm...

zolodeck

<verb> ... <noun>*

zolodeck

reminds you of OOAD?

zolodeck

a little bit?

zolodeck

a lot of that thinking transfers over

zolodeck

around organizing code

zolodeck

namespaces' names = nouns

zolodeck

namespaces' names = nouns

(or verbs)

zolodeck

related functions belong in a namespace

zolodeck

a few public functions

zolodeck

a bunch of private (helper) functions

zolodeck

main namespace called core

zolodeck

main namespace called core

(thanks emacs)

zolodeck

inline code? or a function?

zolodeck

err on the side of many functions

zolodeck

function names are documentation

zolodeck

another reason for creating functions

zolodeck

"I've seen that before"

zolodeck

DRY

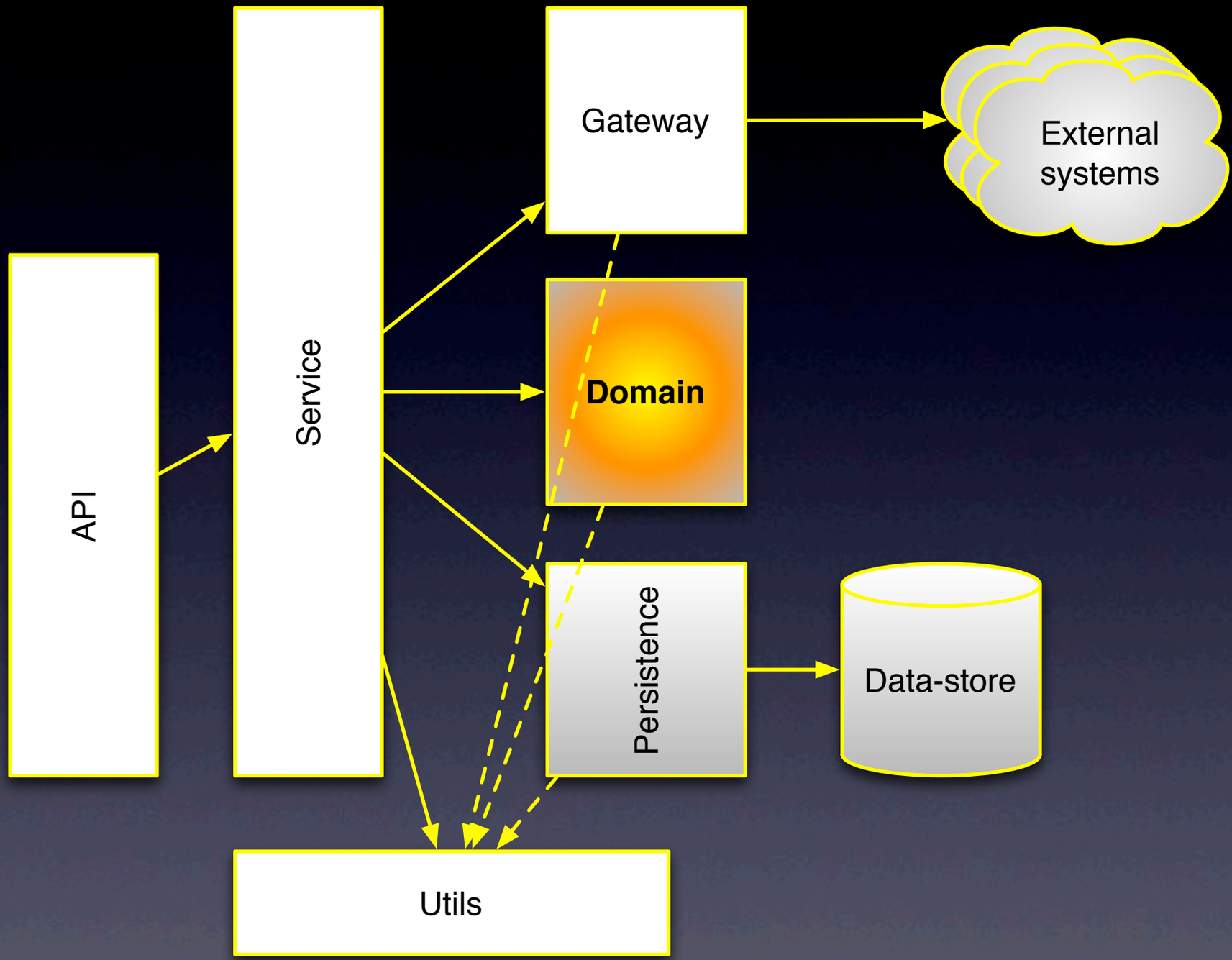
zolodeck

Don't **R**epeat **Y**ourself

zolodeck

which namespace to put it in?

zolodeck



zolo.*

zolodeck

if business folks care, put it in domain...

zolodeck

otherwise, elsewhere

zolodeck

test fns in “mirror” test namespaces

zolodeck

domain model = core of your app

zolodeck

bottom up

zolodeck

manage complexity through simplicity

zolodeck

express complex concepts (functions)

express complex concepts (functions)

by building upon simpler, lower level concepts (functions)

zolodeck

I. Domain

zolodeck

2. Driven

zolodeck

start with domain functions

zolodeck

thin slices around the domain model

zolodeck

build the app as series of these slices

zolodeck

puts the business in control

zolodeck

testing

zolodeck

domain = pure functions

zolodeck

unit-testing

zolodeck

test-driven development

zolodeck

stubbing/mocking rarely needed at this level

zolodeck

mocking/stubbing useful to test higher levels

zolodeck

binding, atticus, conjure

...

zolodeck

a few more thoughts

zolodeck

single responsibility

zolodeck

a function should do only one thing

zolodeck

unix-like philosophy to functions

zolodeck

small functions

zolodeck

small functions

(LOC)

zolodeck

8-9 lines in Java

zolodeck

8-9 lines in Java

4-5 lines in Ruby

zolodeck

8-9 lines in Java

4-5 lines in Ruby

3-4 lines in Clojure

zolodeck

8-9 lines in Java

4-5 lines in Ruby

3-4 lines in Clojure

(0 lines in Haskell)

zolodeck

optimize for readability

OH:

“I can’t understand it, it must be advanced FP”

zolodeck

nope

zolodeck

nope

(it's just clever code)

zolodeck

if you don't understand, demand simplification

zolodeck

or just change it yourself

zolodeck

or just change it yourself

(now)

zolodeck

collective code ownership

zolodeck

collective code ownership

Seriously.

zolodeck

domain model is collectively owned

zolodeck

domain model is collectively owned

(scale the team)

zolodeck

1. Domain

zolodeck

1. Domain

2. Driven

zolodeck

3. Design

zolodeck

DDD - design in the large

zolodeck

and the small

zolodeck

desired goal = higher flexibility

zolodeck

ubiquitous language

zolodeck

common vocabulary

zolodeck

developers, business folks

zolodeck

good, flexible domain model

zolodeck

basis for solving problems

zolodeck

basis for a DSL runtime

zolodeck

domain specific languages

zolodeck

script the domain model

zolodeck

solve class of problems

zolodeck

Data instantiation

DSL scripts

Generic validation

DSL runtime

Domain-Specific Validation

Domain Model

Domain

zolodeck

iterative approach

zolodeck

domain model evolves

zolodeck

fns move around

zolodeck

temporary namespaces...

zolodeck

eventually collapse into domain, service, etc.

zolodeck

zolo.stats.* -> zolo.domain.*

zolodeck

explicit temporary location

zolodeck

zolo.homeless

zolodeck

final thoughts

zolodeck

manage complexity from the domain

zolodeck

remove incidental complexity

zolodeck

as simple as possible, no simpler

zolodeck

YAGNI

zolodeck

you **a**ren't **g**oing to **n**eed **i**t

zolodeck

domain model
small functions
DRY
testing
basis of DSL runtime
ubiquitous language
YAGNI

zolodeck

DDD = agility

zolodeck

especially good for startups

zolodeck

and for non-startups

zolodeck

thanks!

amit@zololabs.com
@amitrathore

P.S. - Sign up at www.zolodeck.com