Software Architecture Stories

Guille Polito
@guillep

CNRS - UMR9189
CRIStAL

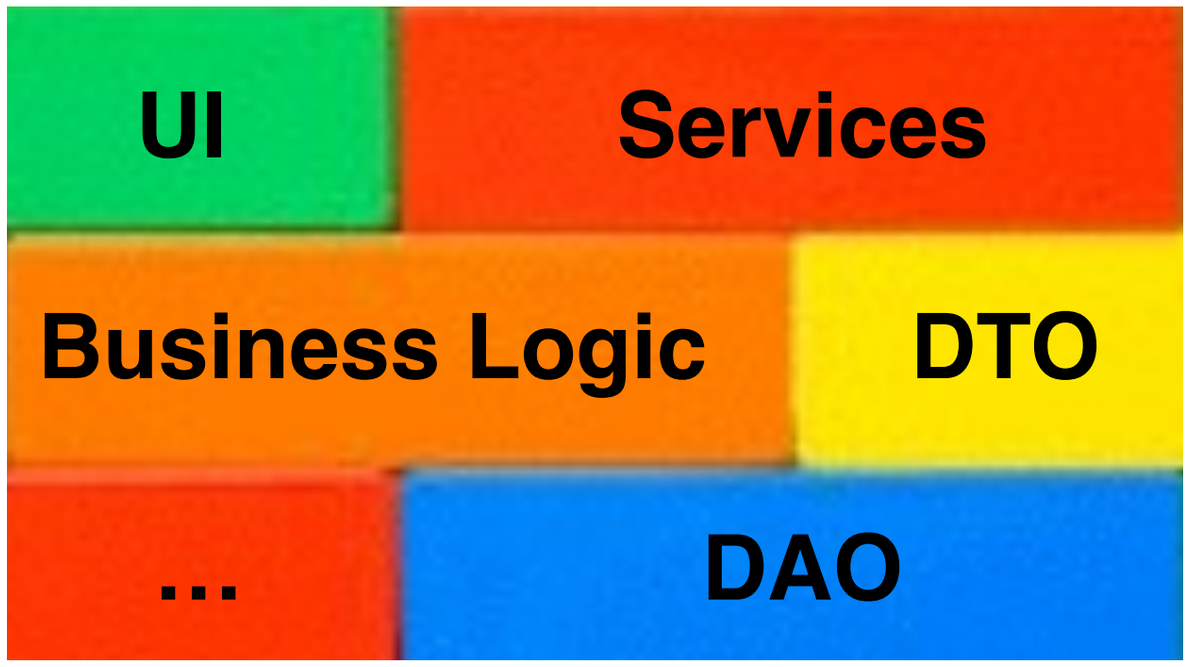
RMod

cnrs

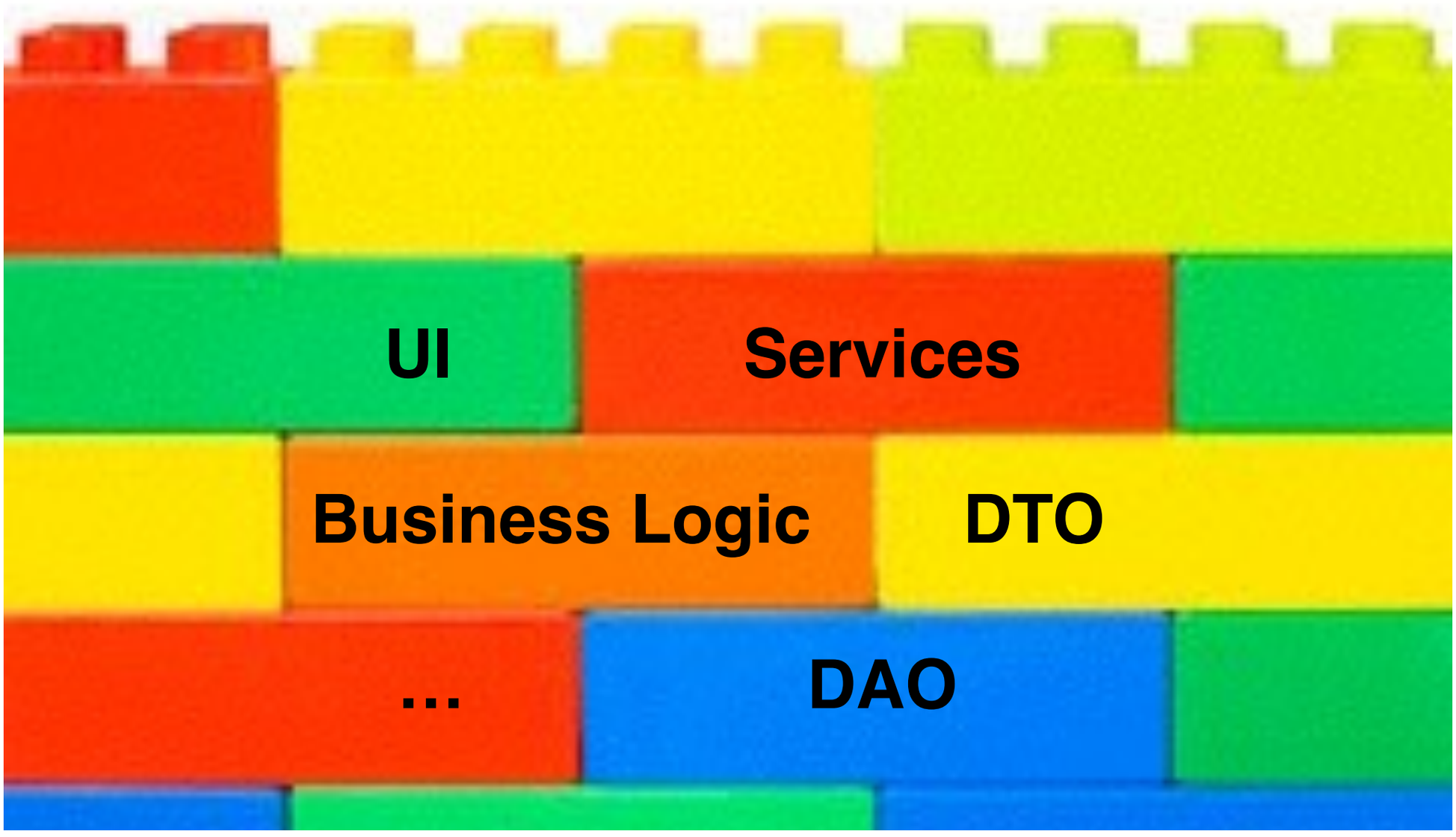
Université de Lille

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

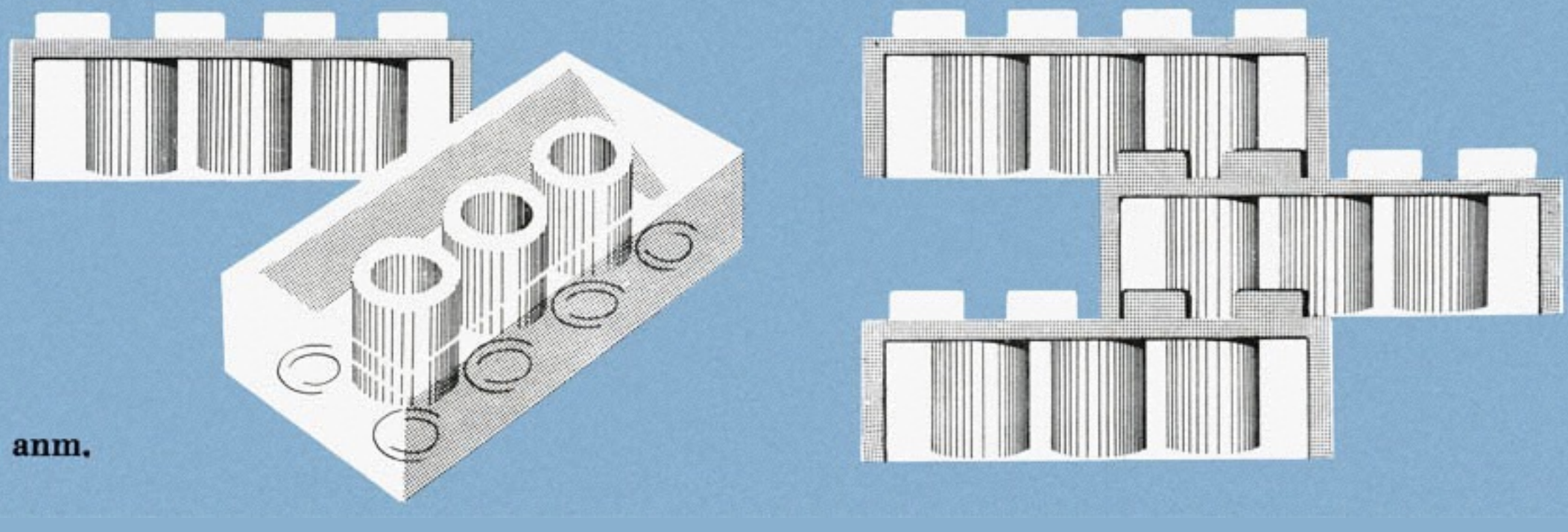
Inria
inventors for the digital world
50 YEARS



Enterprise Architectures



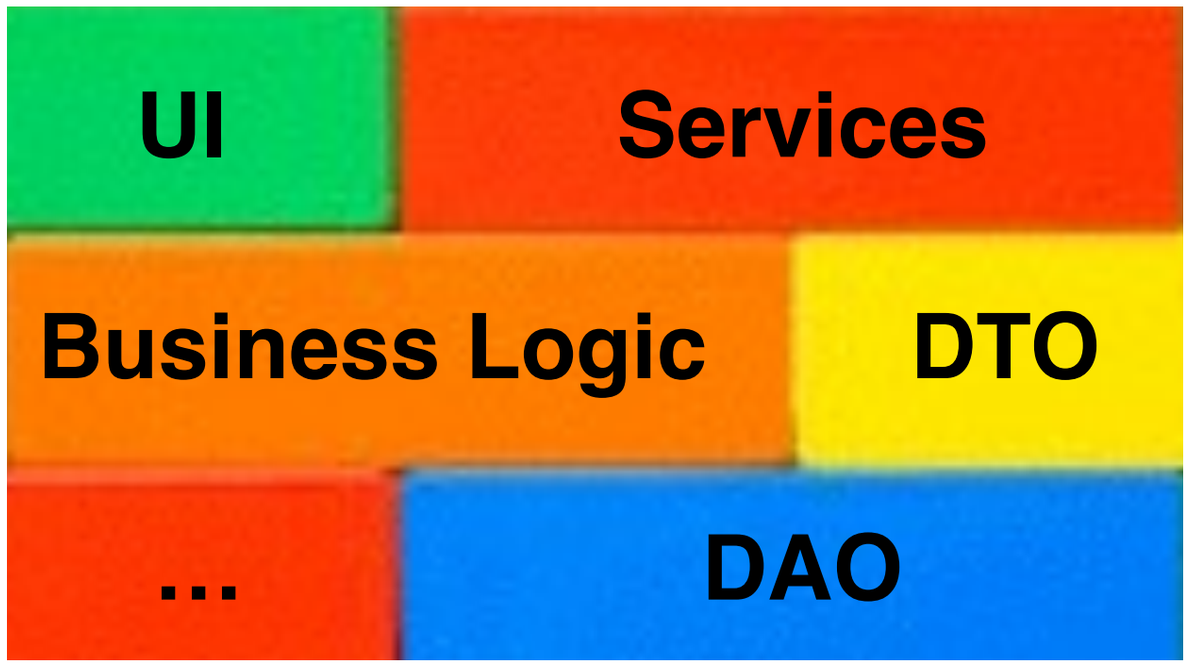
What I feel about Enterprise Architectures



The Stud-and-Tube coupling
Lego AS, 1958

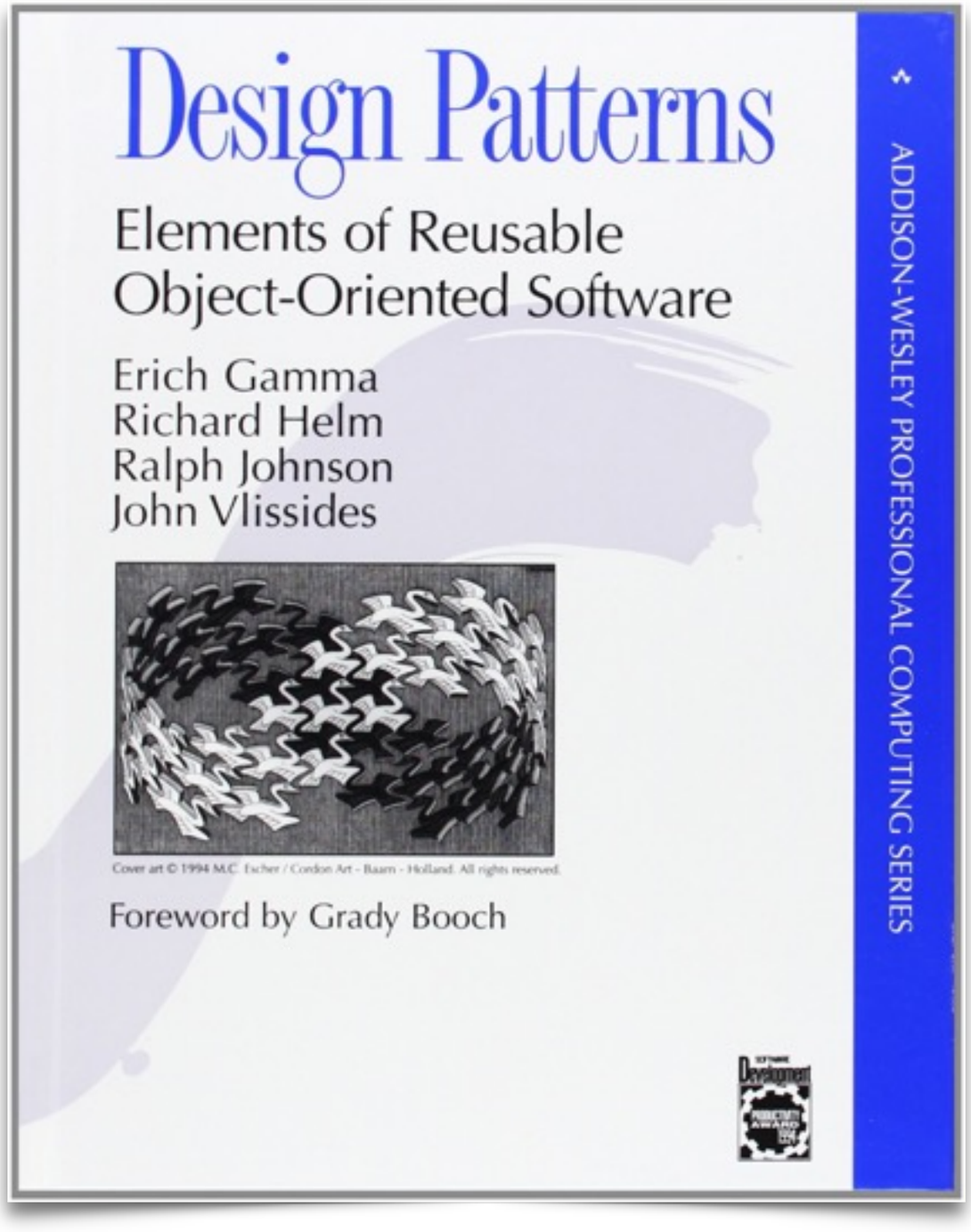
The Lego System

6 bricks => 915,103,765 combinations



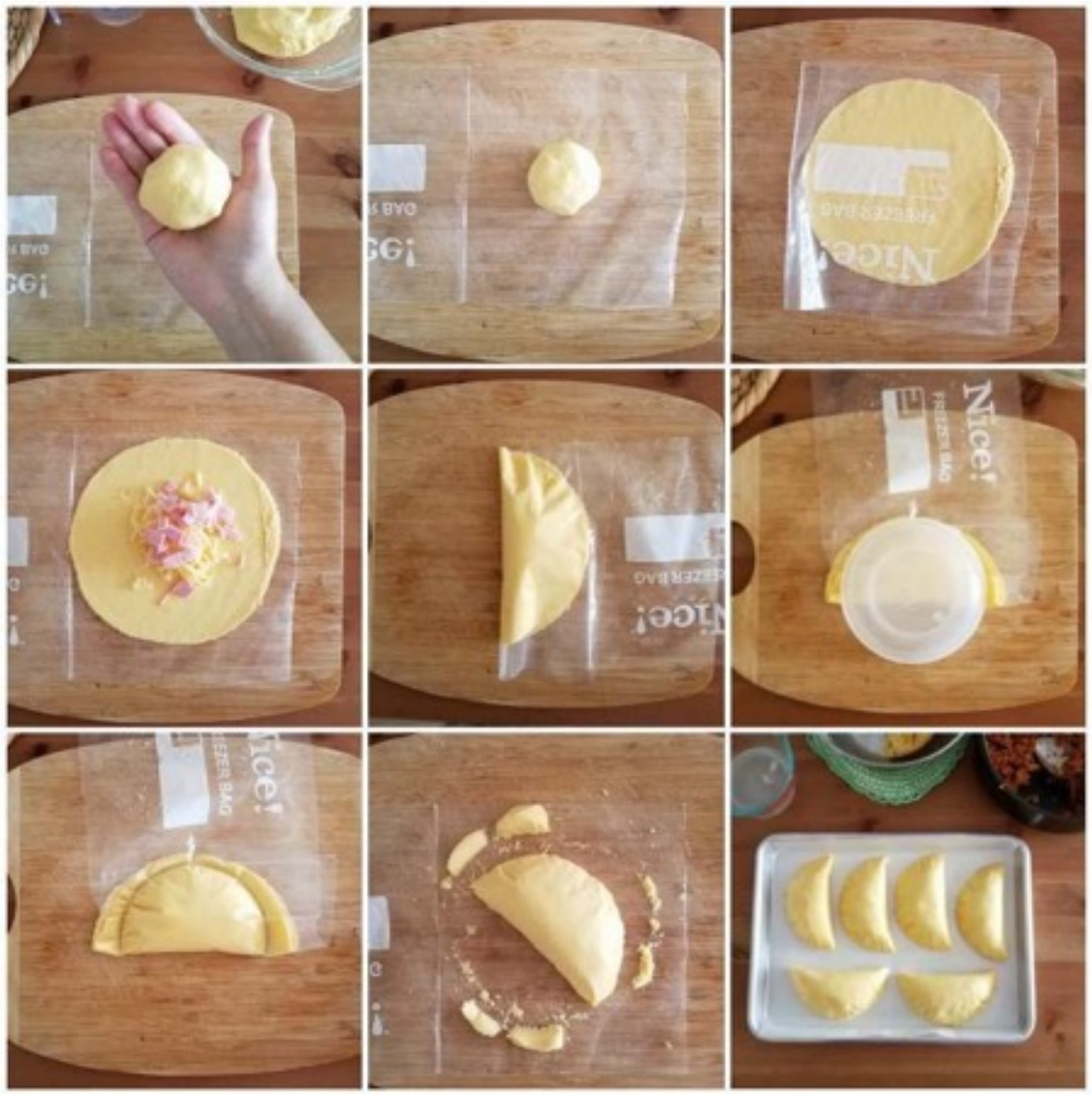
Not so Lego System…

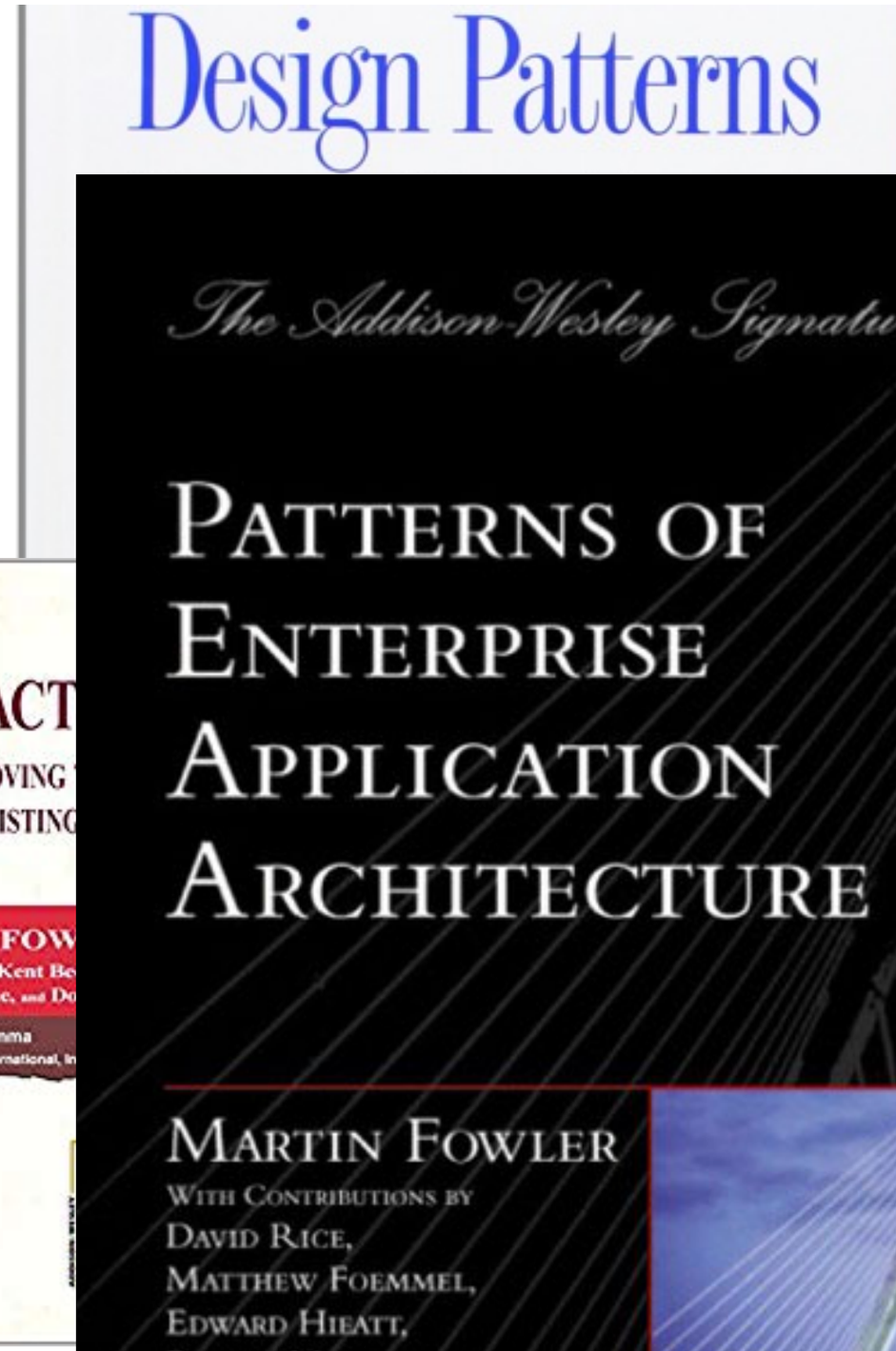
Patterns as small-scale architectures



Patterns as tools

Patterns as recipes



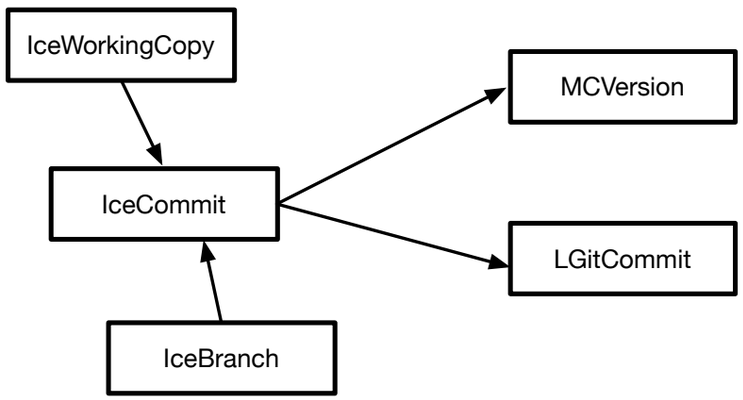


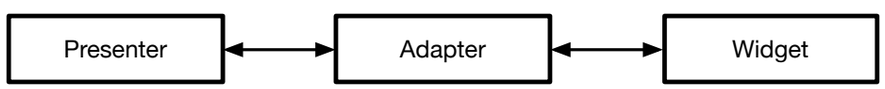
~~Architecture~~ crafting

Pattern cooking

- understand the architecture
- modify the architecture
- make your own architecture

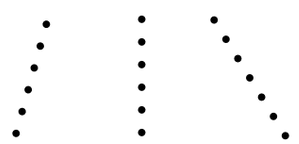
Three stories of Pattern cooking





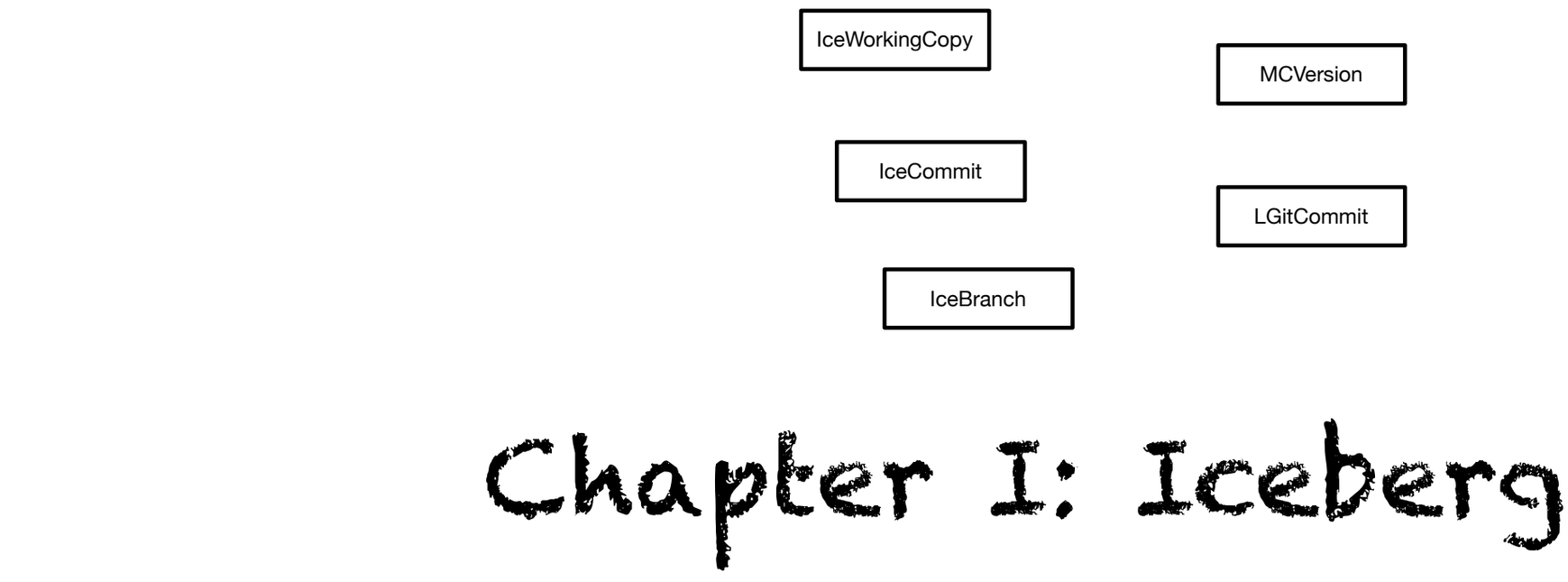

```
[ …a task… ] schedule

runner schedule: […a task…]
```

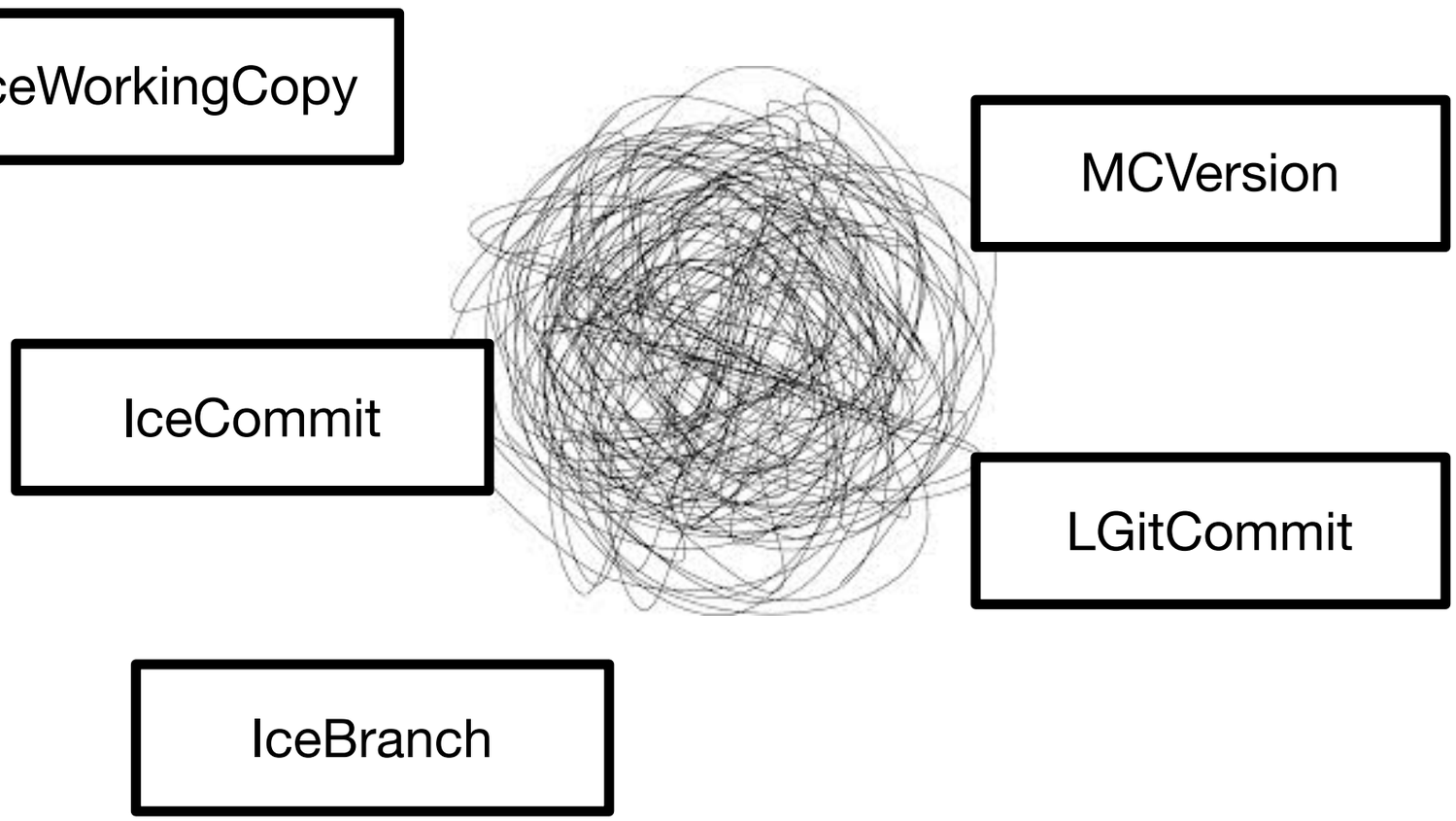


sync async …

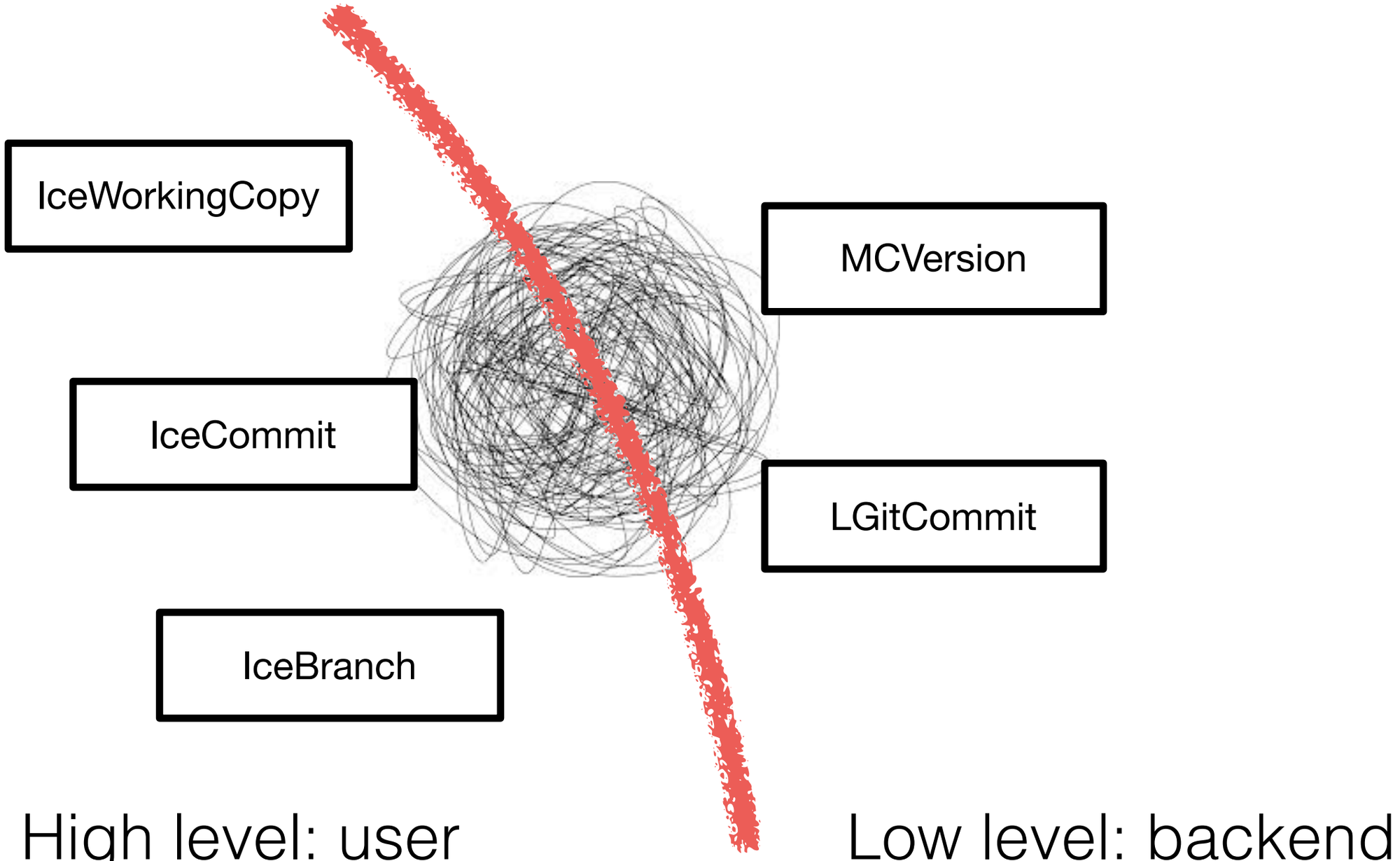
Let’s talk about layers



Chapter I: Iceberg



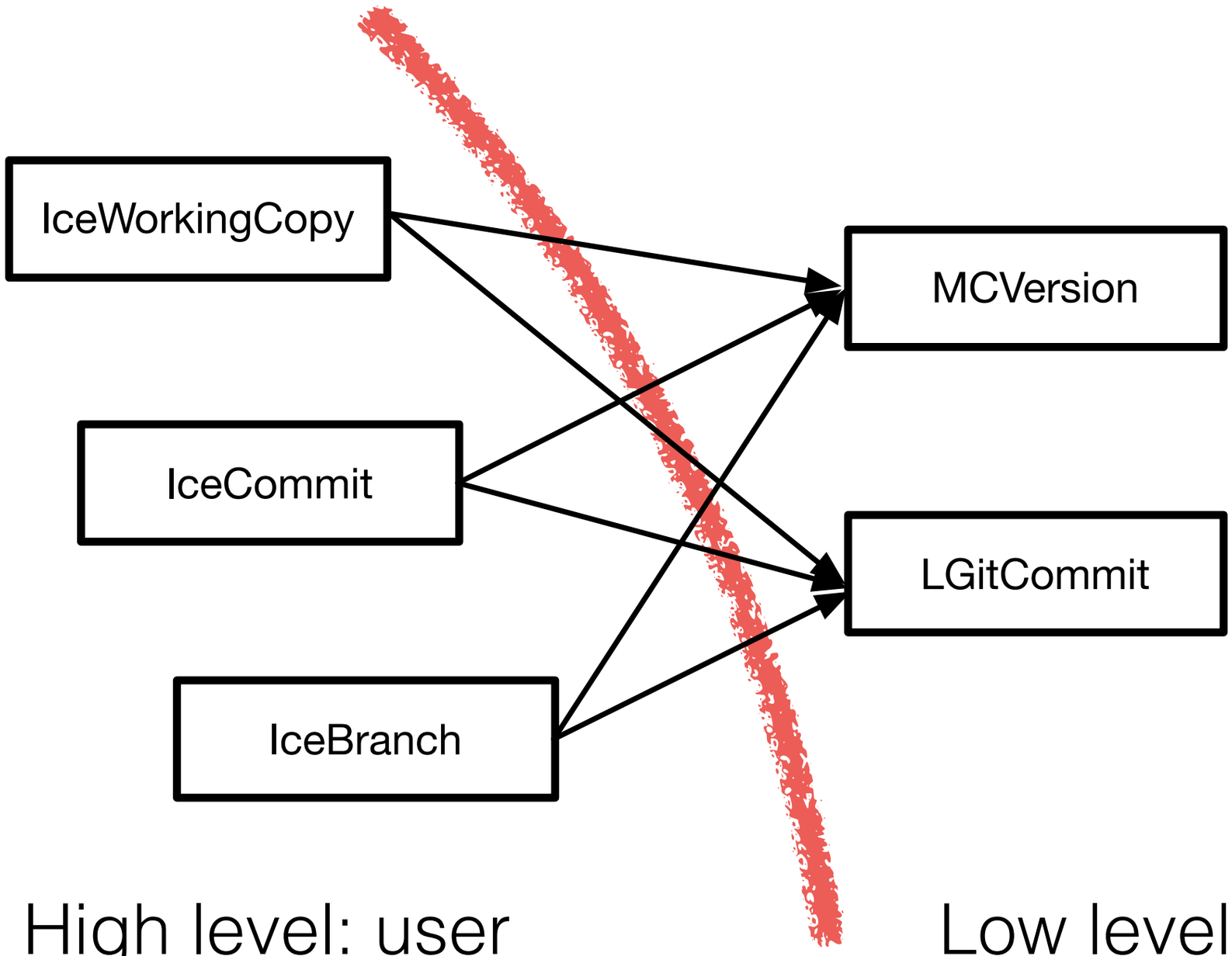
Chapter I: Iceberg



High level: user

Low level: backend

Chapter I: Iceberg

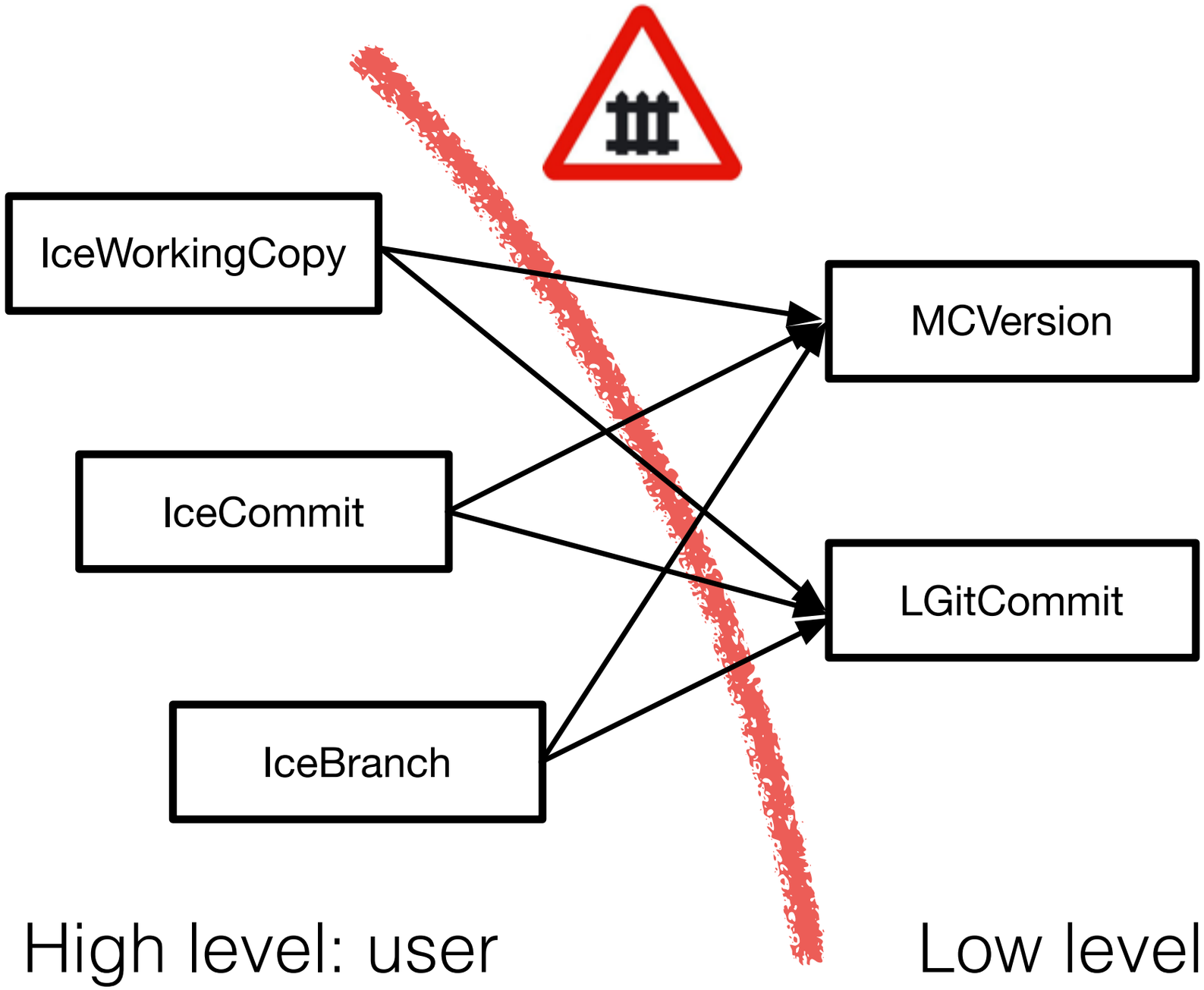


Chapter I: Iceberg

High level: user

Low level: backend

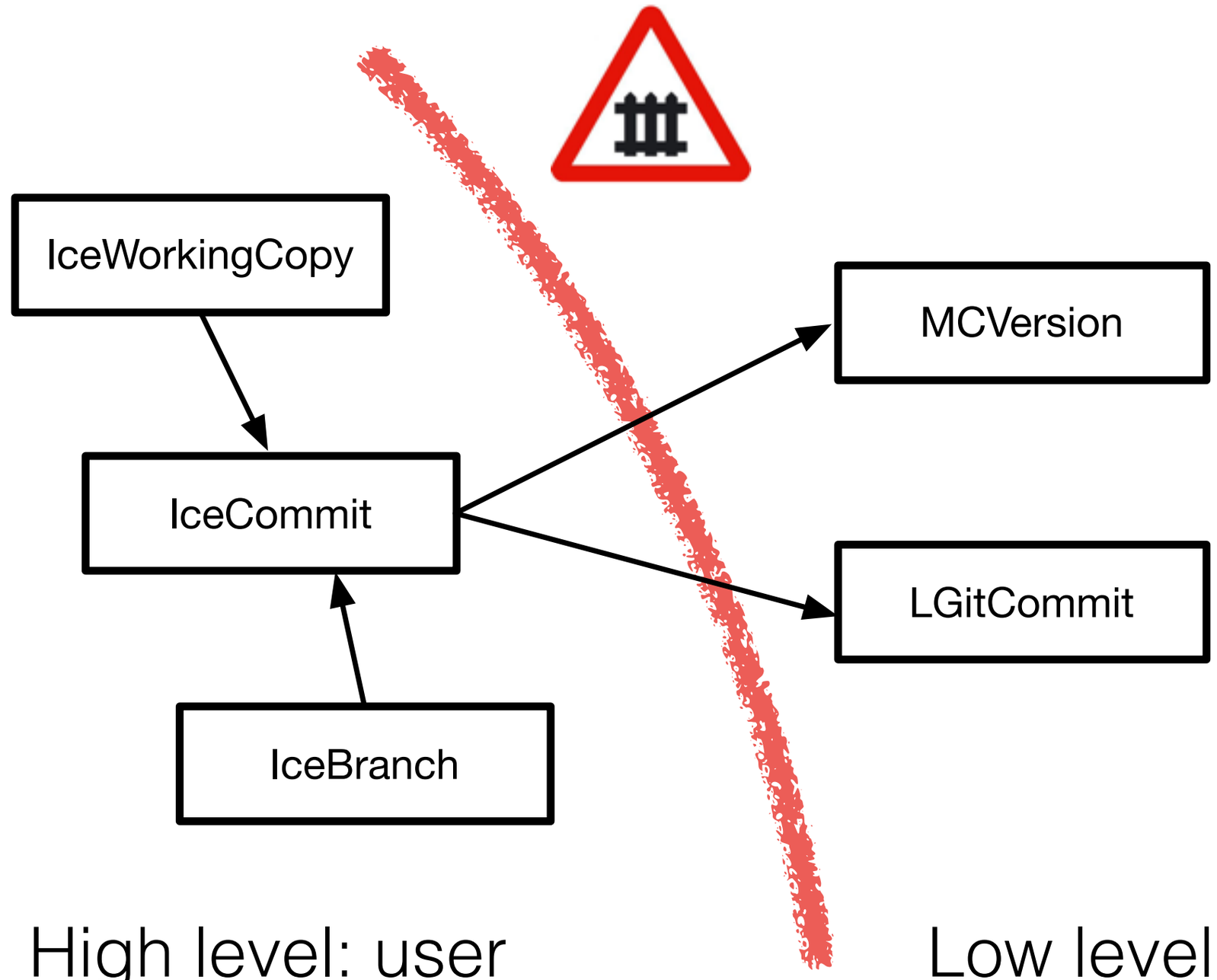
Chapter I: Iceberg



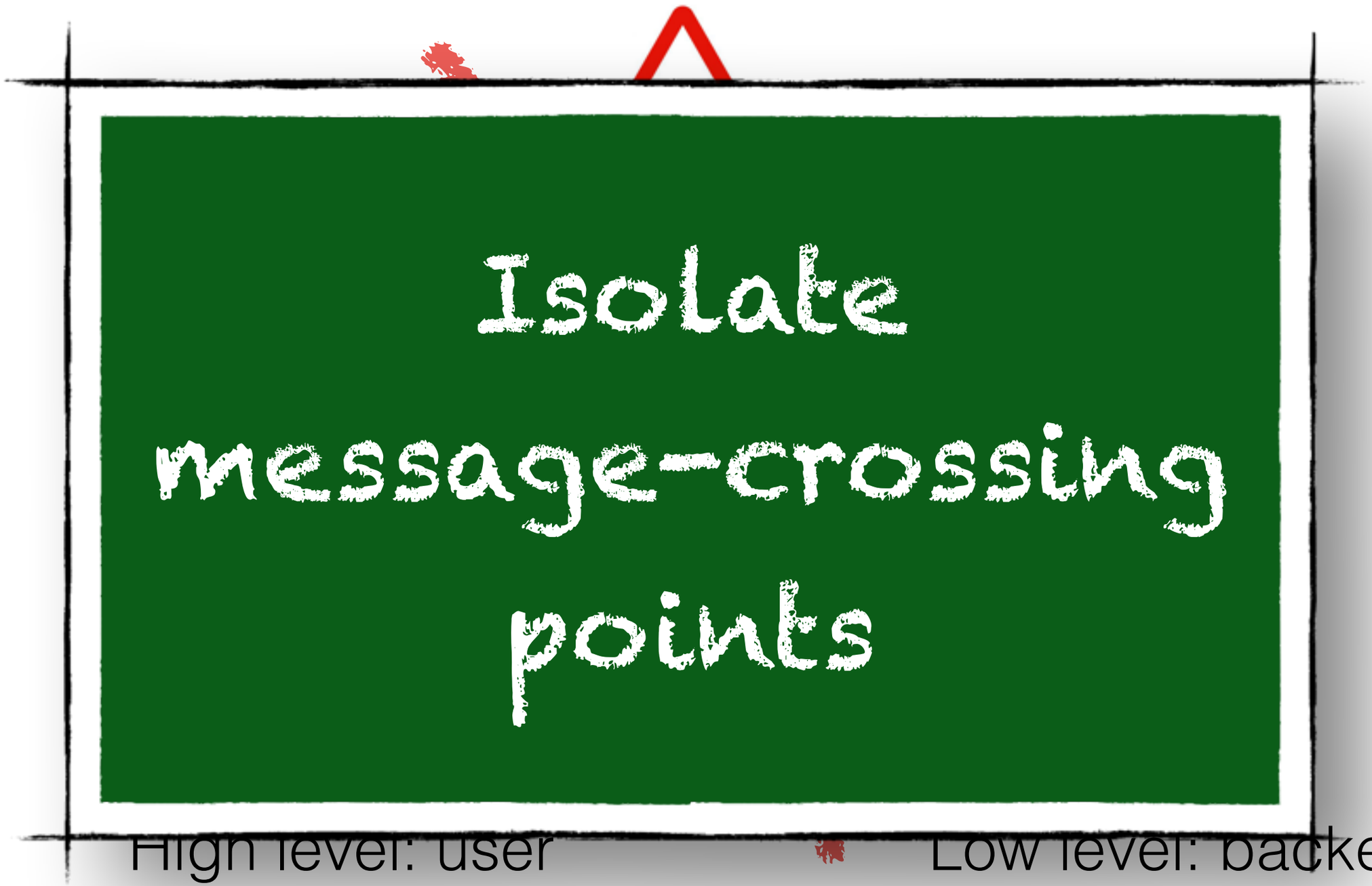
High level: user

Low level: backend

Chapter I: Iceberg



Chapter I: Iceberg



High level: user

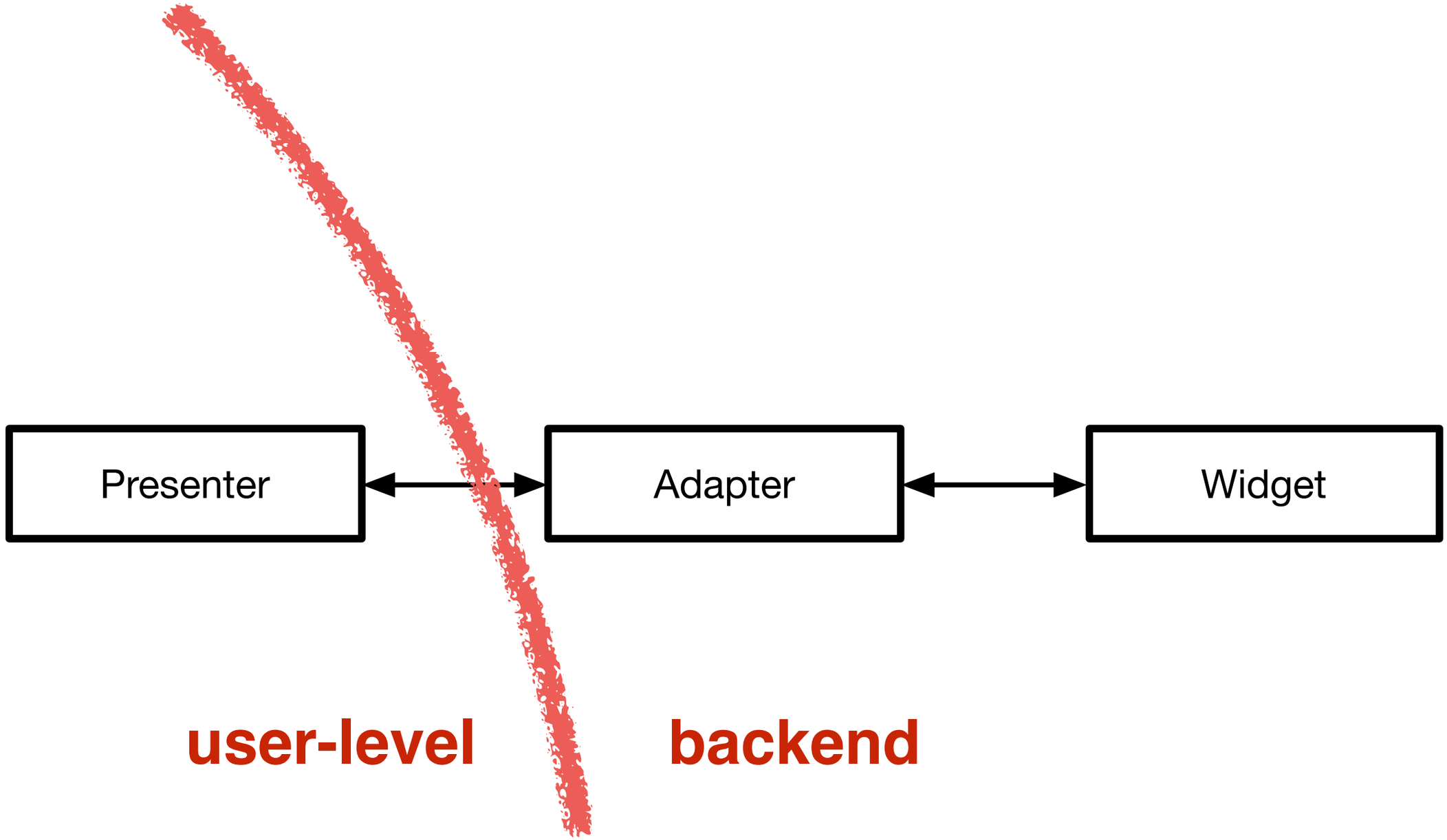
Low level: backend

But several layers,
there will be…

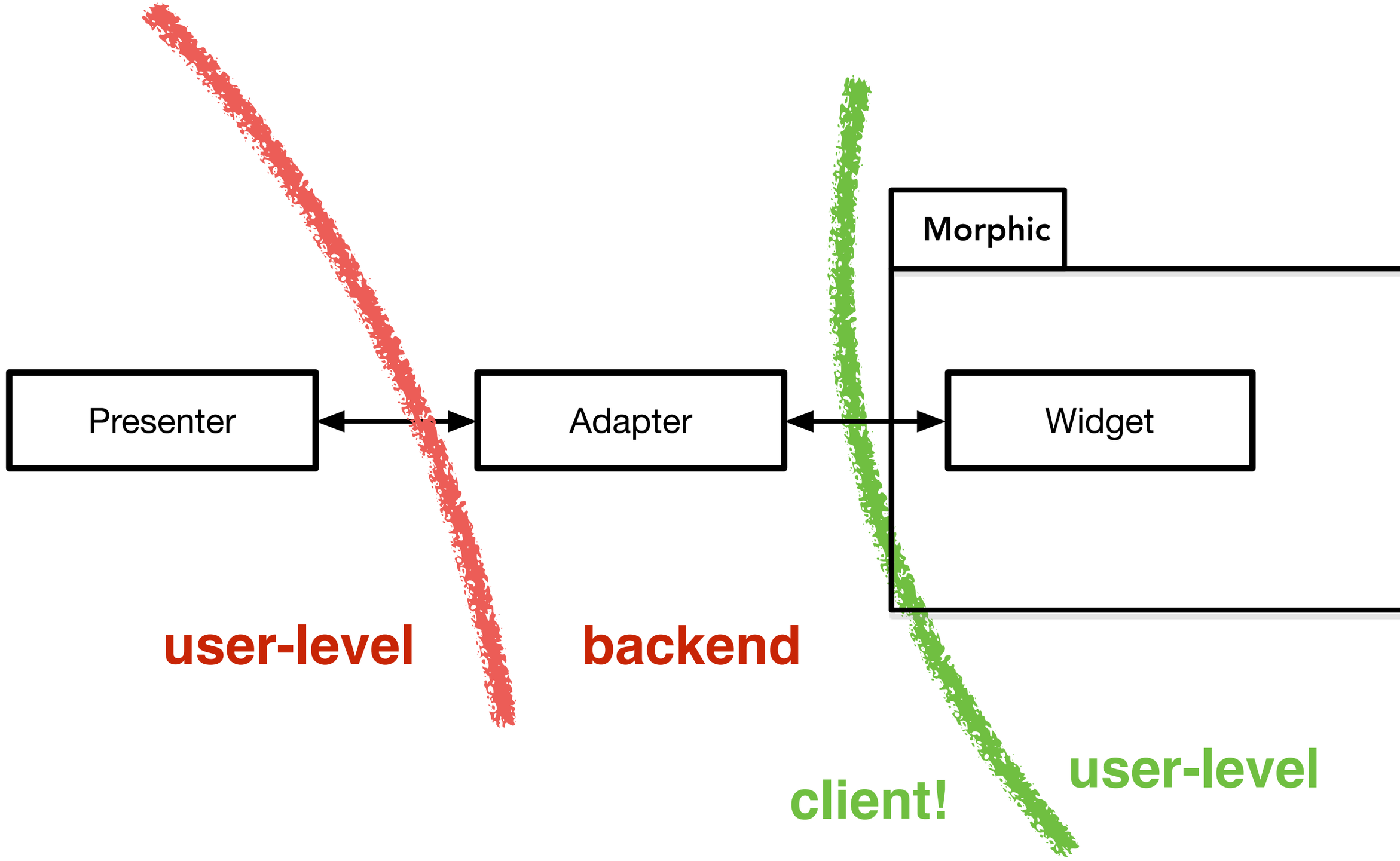
Chapter II: Spec

Chapter II: Spec

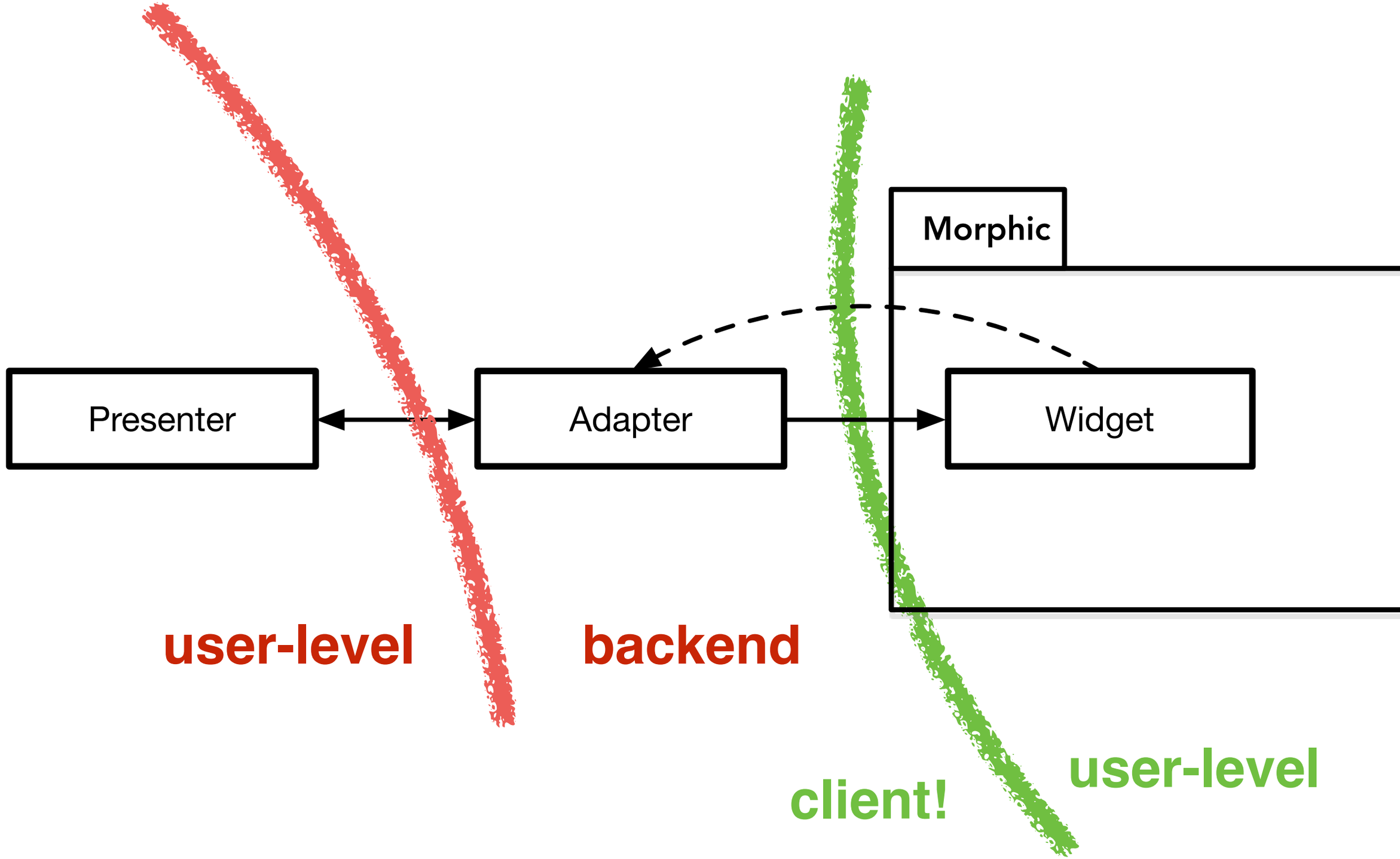
Chapter II: Spec



Chapter II: Spec



Chapter II: Spec

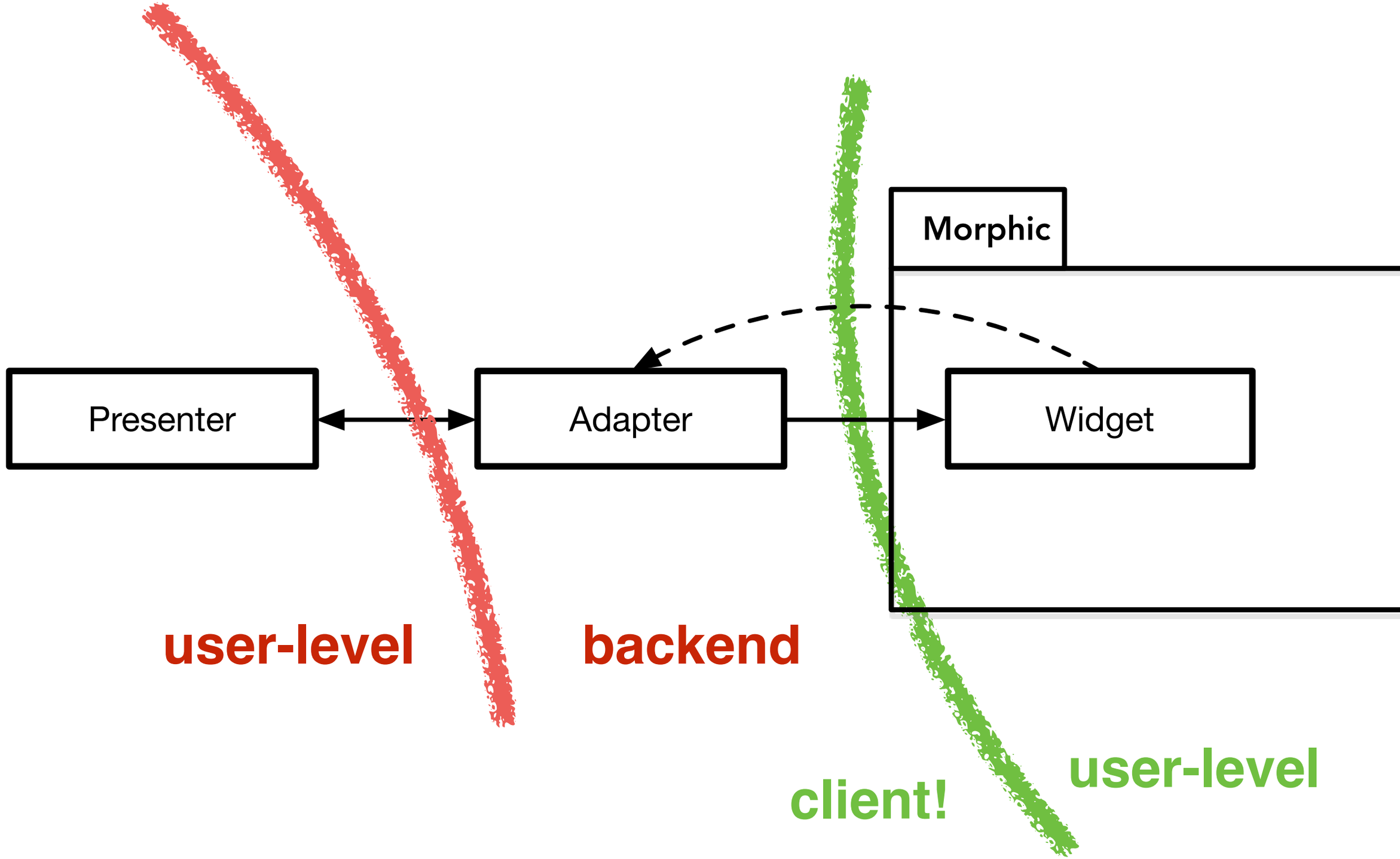


Chapter II: Spec

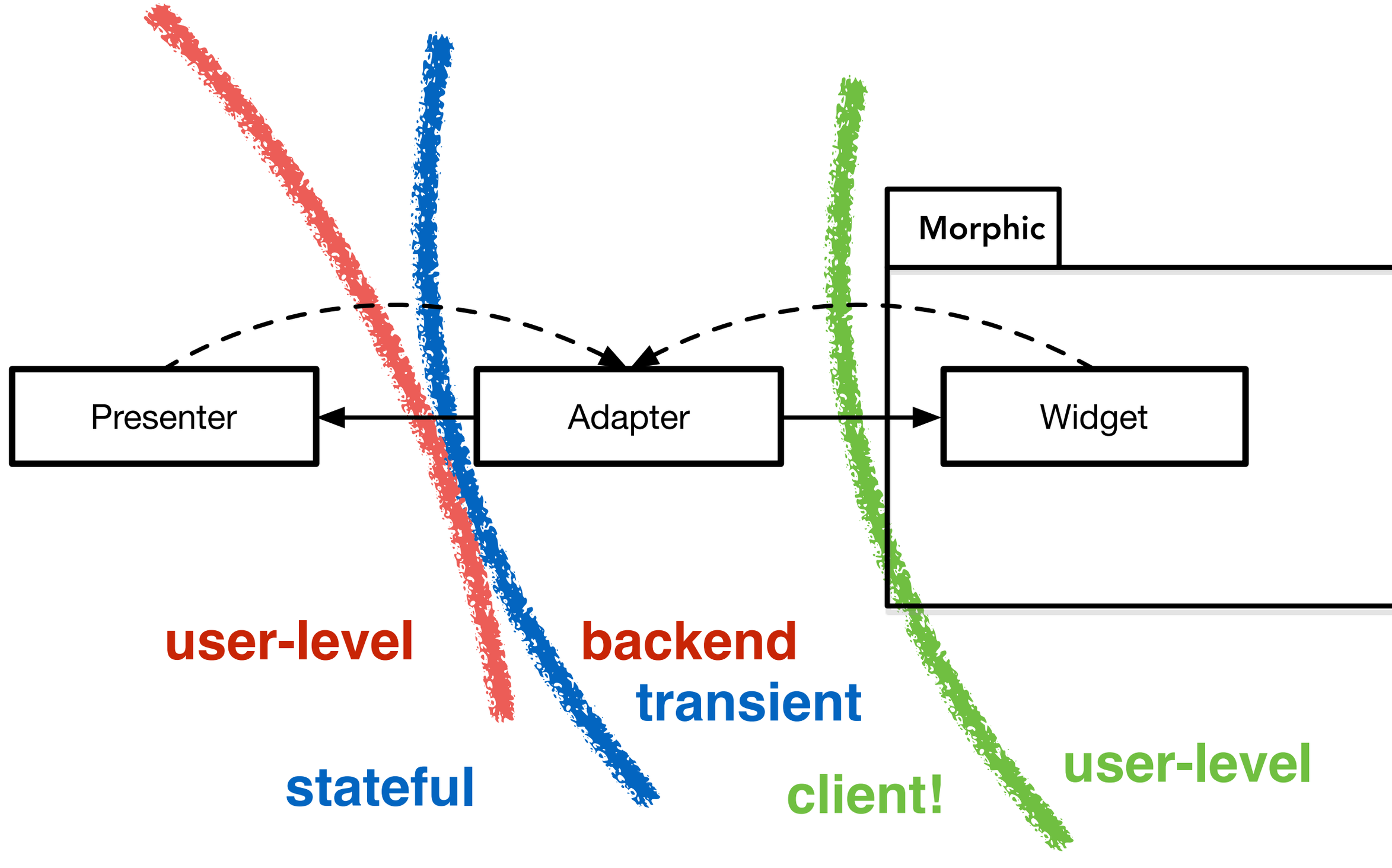
Loose coupling
via
event-messages

user-level
client!

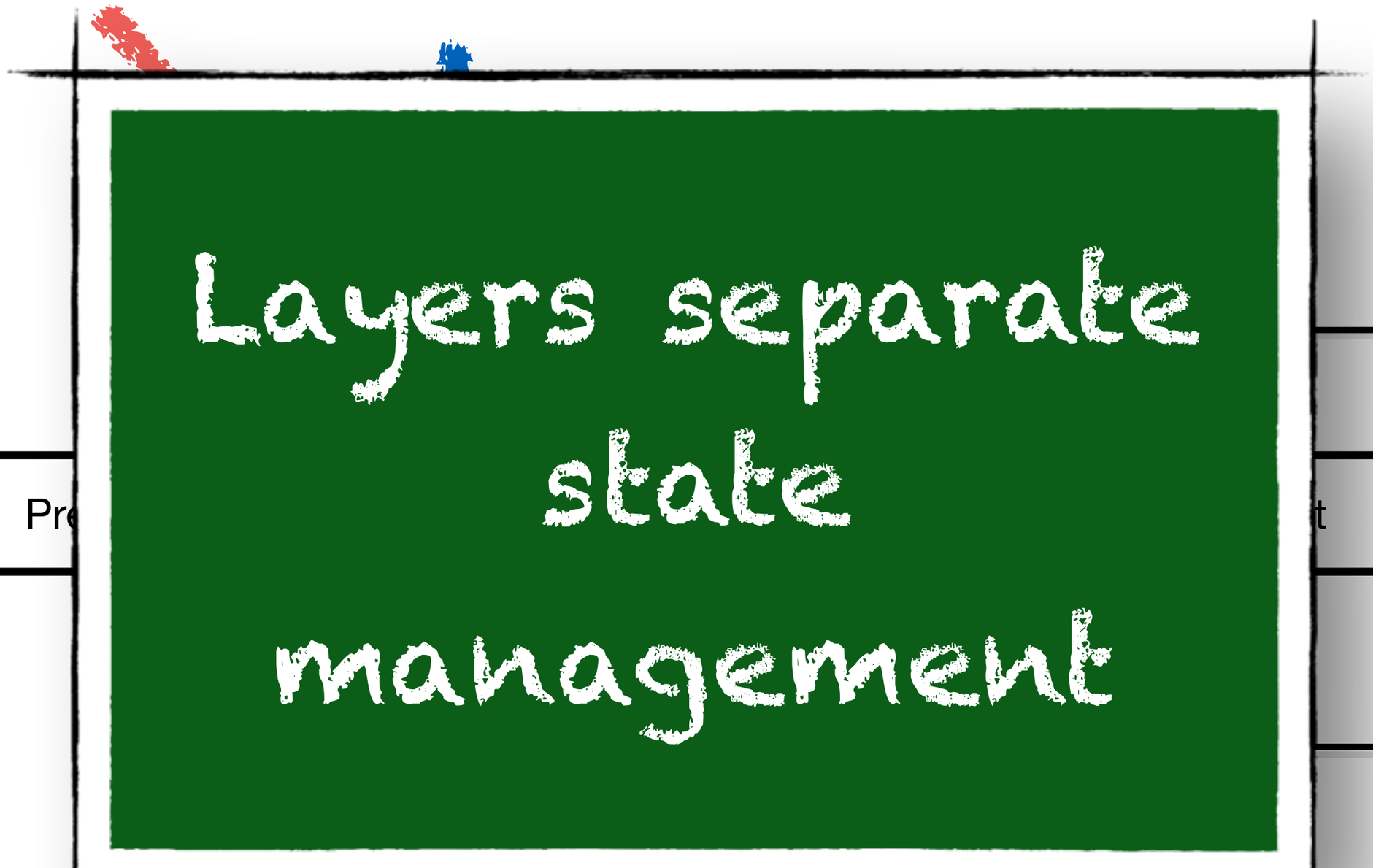
Chapter II: Spec



Chapter II: Spec



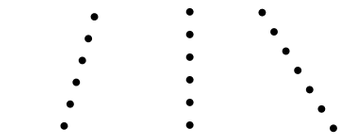
Chapter II: Spec



Separate components only they shall not…

```
[ …a task… ] schedule

runner schedule: […a task…]
```



Chapter III: TaskIt

Chapter III: TaskIt

```
[ …a task… ] schedule
```

and eventually execute

Chapter III: TaskIt

`[ …a task… ] schedule`

and eventually execute

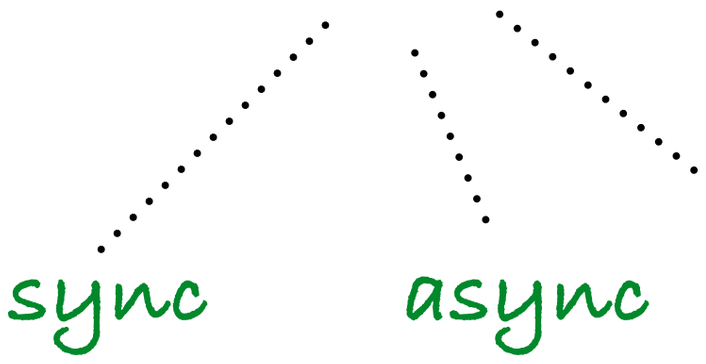
but how?

Chapter III: TaskIt

Convenience

`[ …a task… ] schedule`

`runner schedule: […a task…]`



Explicit

Chapter III: TaskIt

Convenience

Thin convenience layers

on top of explicit layers

syn

Three lessons of Architecture crafting

Guille Polito
@guillep

CNRS - UMR9189
CRIStAL

separate by levels of abstraction

isolate points of failure

separate stateful from stateless

be explicit, as less magic as possible