



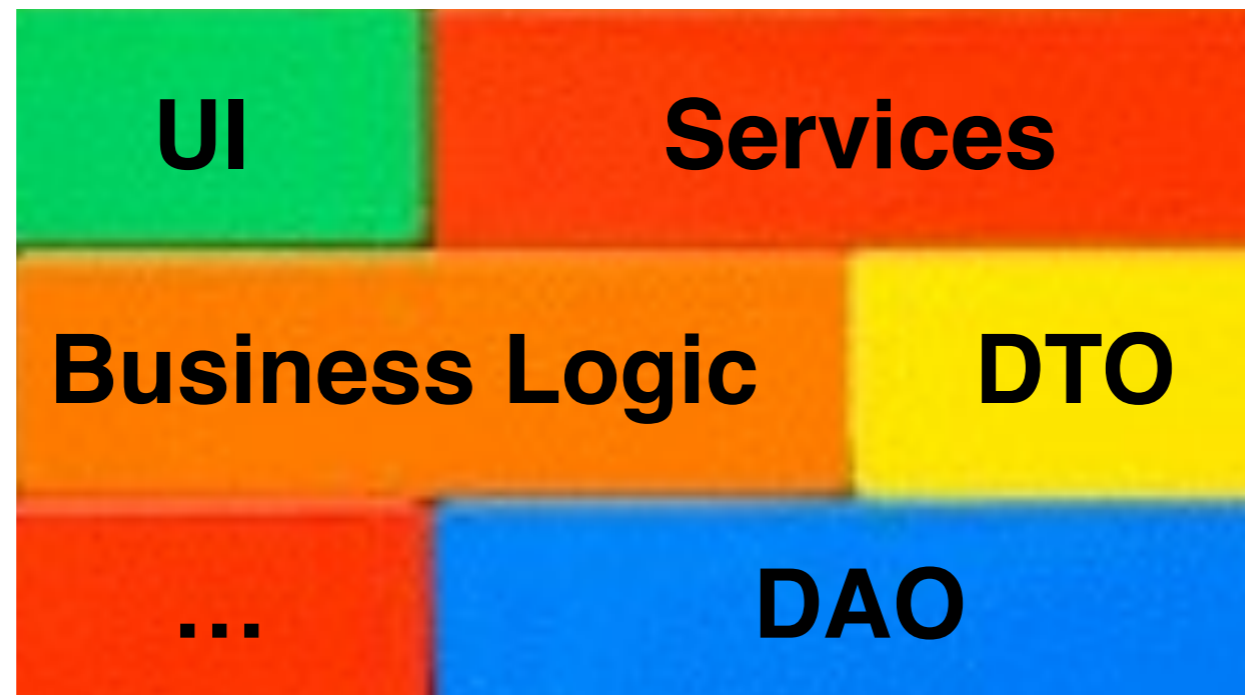
Software Architecture Stories



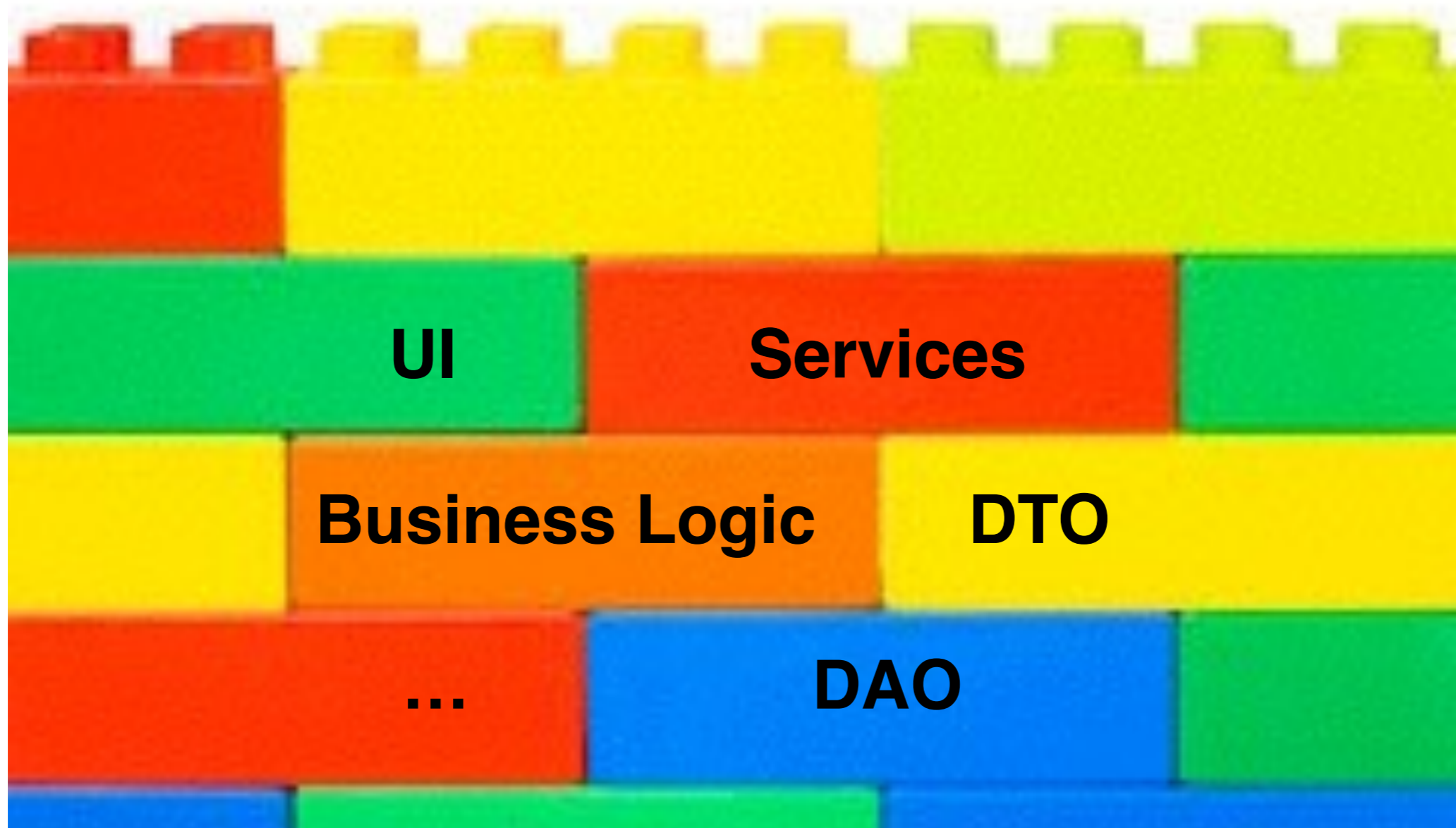
Guille Polito
@guillep

CNRS - UMR9189
CRISAL

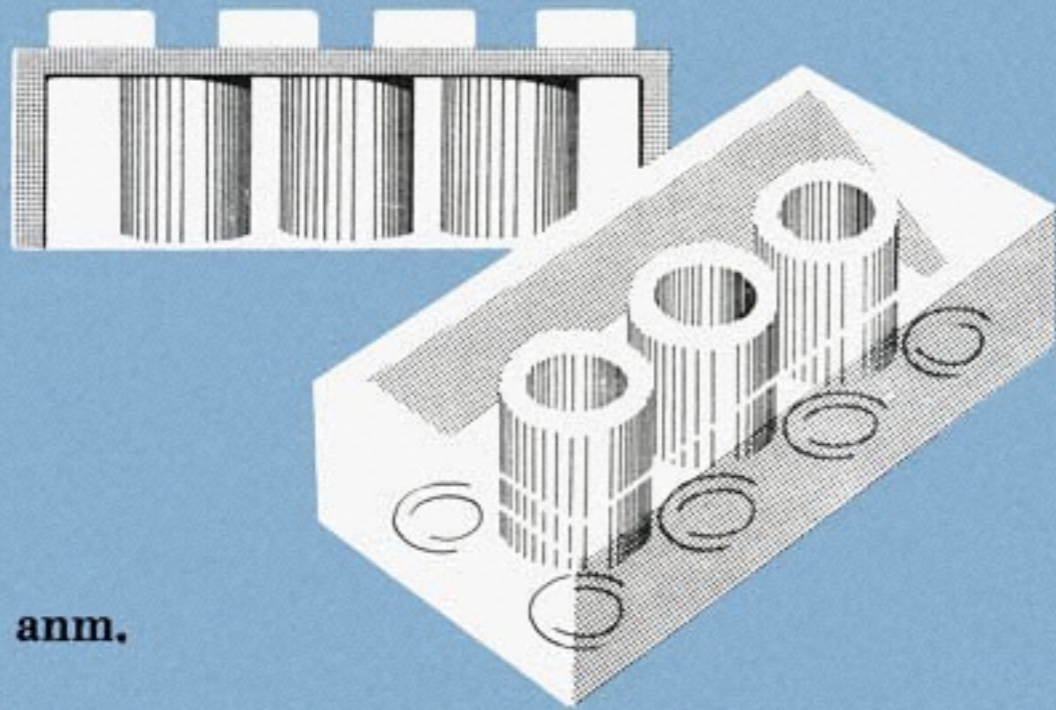




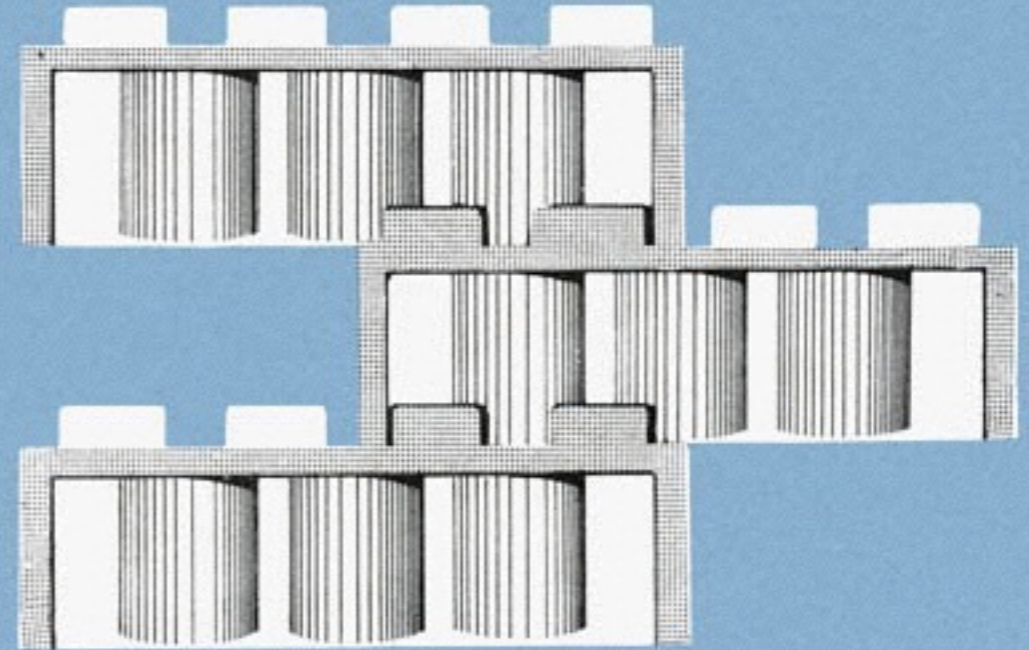
Enterprise Architectures



What I feel about
Enterprise Architectures



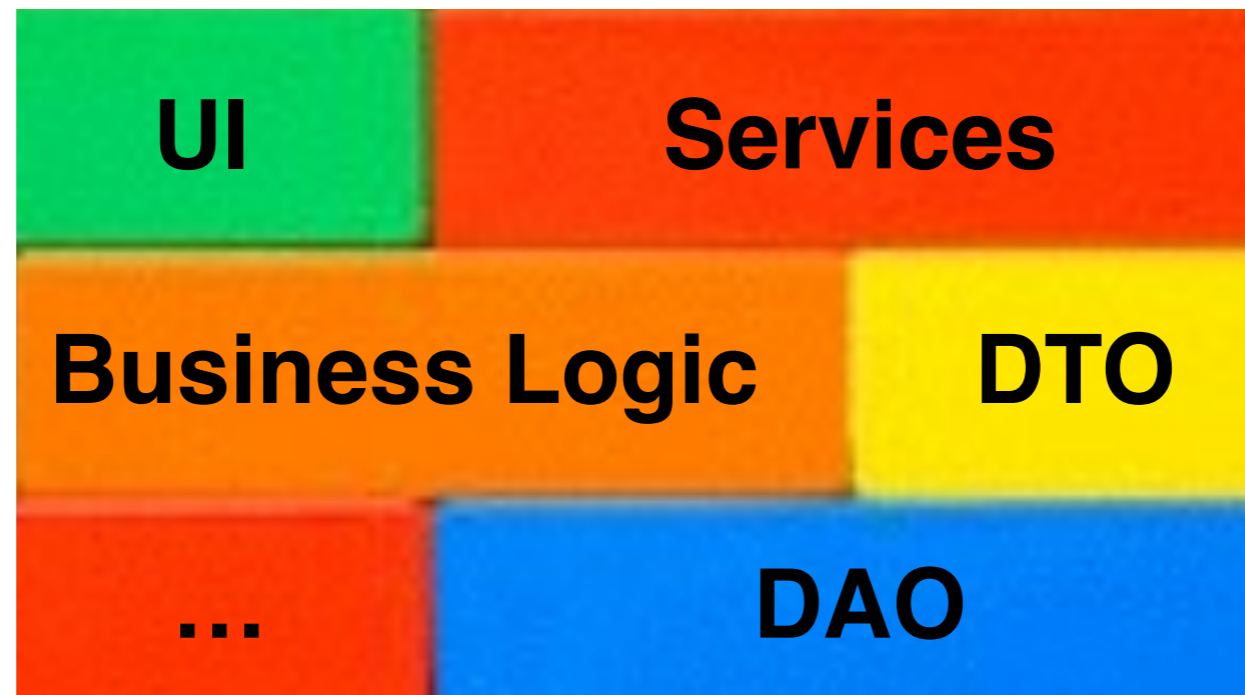
Pat. ann.



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

Design Patterns

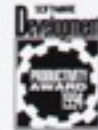
Elements of Reusable
Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides



Cover art © 1994 M.C. Escher / Condon Art - Baarn - Holland. All rights reserved.

Foreword by Grady Booch



ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

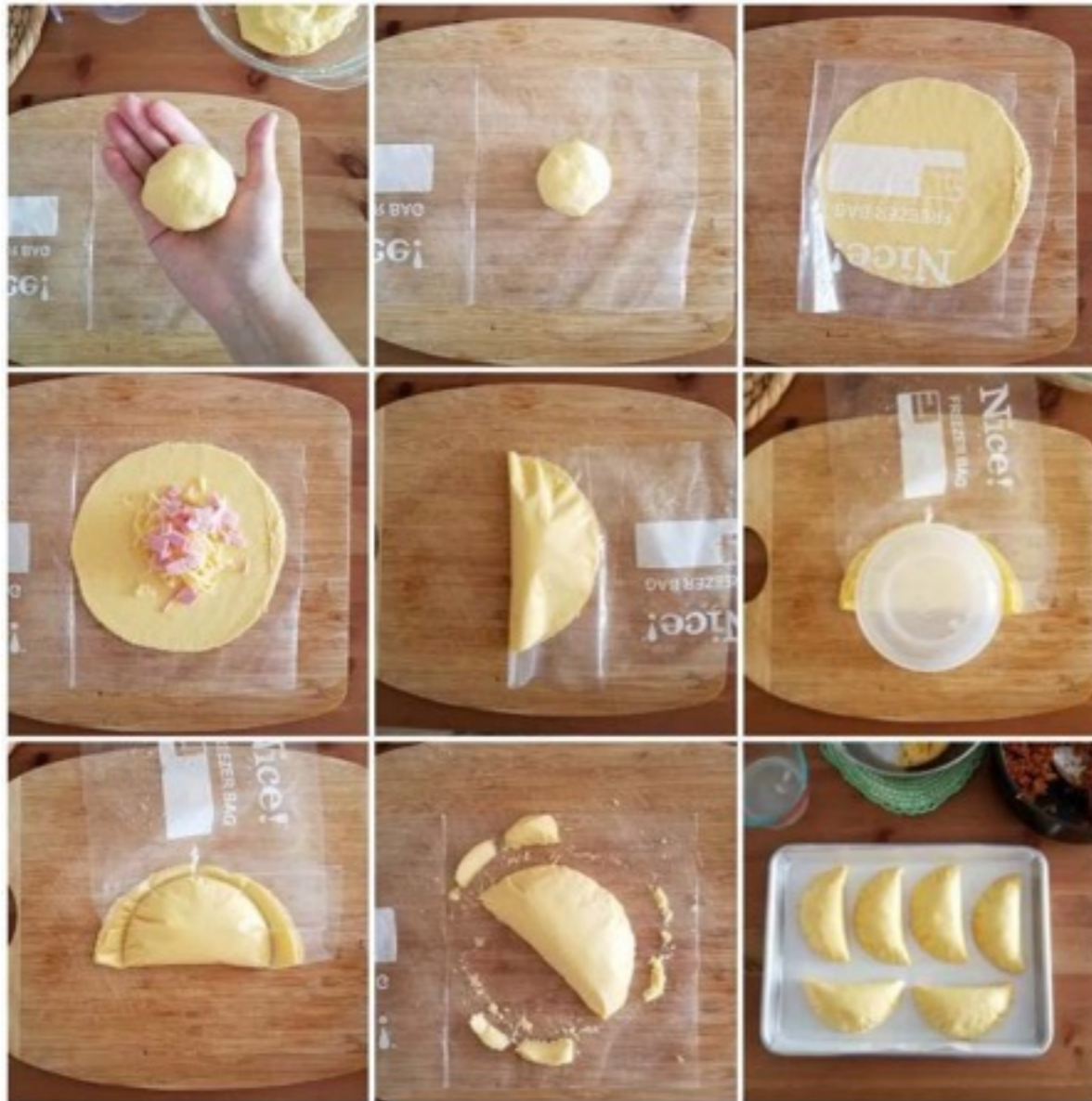


Patterns as tools





Patterns as recipes



Design Patterns

The Addison-Wesley Signature

PATTERNS OF ENTERPRISE APPLICATION ARCHITECTURE

REFACT
IMPROVING
OF EXISTING

MARTIN FOWLER

With contributions by Kent Beck,
William Opdyke, and Doug
Ramage

Foreword by Erich Gamma
Object Technology International, Inc.

MARTIN FOWLER

WITH CONTRIBUTIONS BY
DAVID RICE,
MATTHEW FOEMMEL,
EDWARD HEATT,
ROBERT MEE, AND
RANDY STAFFORD





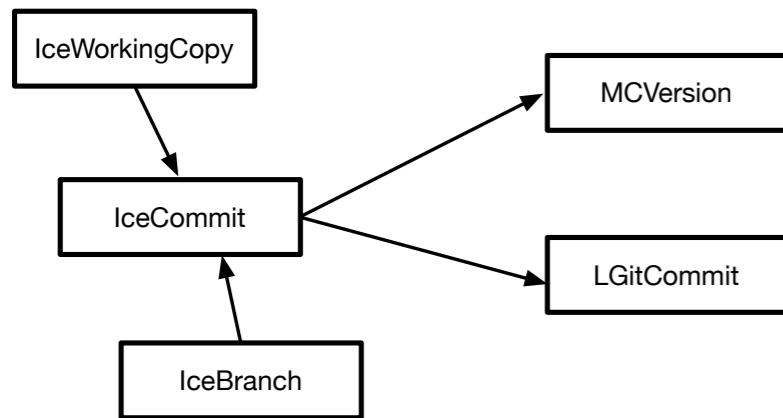
Architecture cratting

Pattern cooking

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

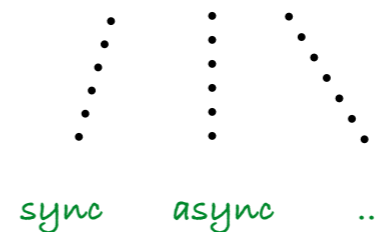
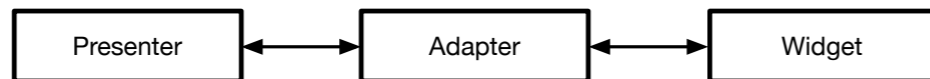


Three stories of Pattern cooking

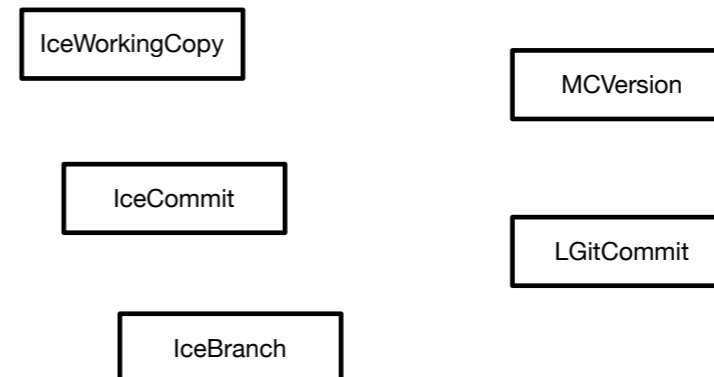


[...a task...] schedule

runner schedule: [...a task...]



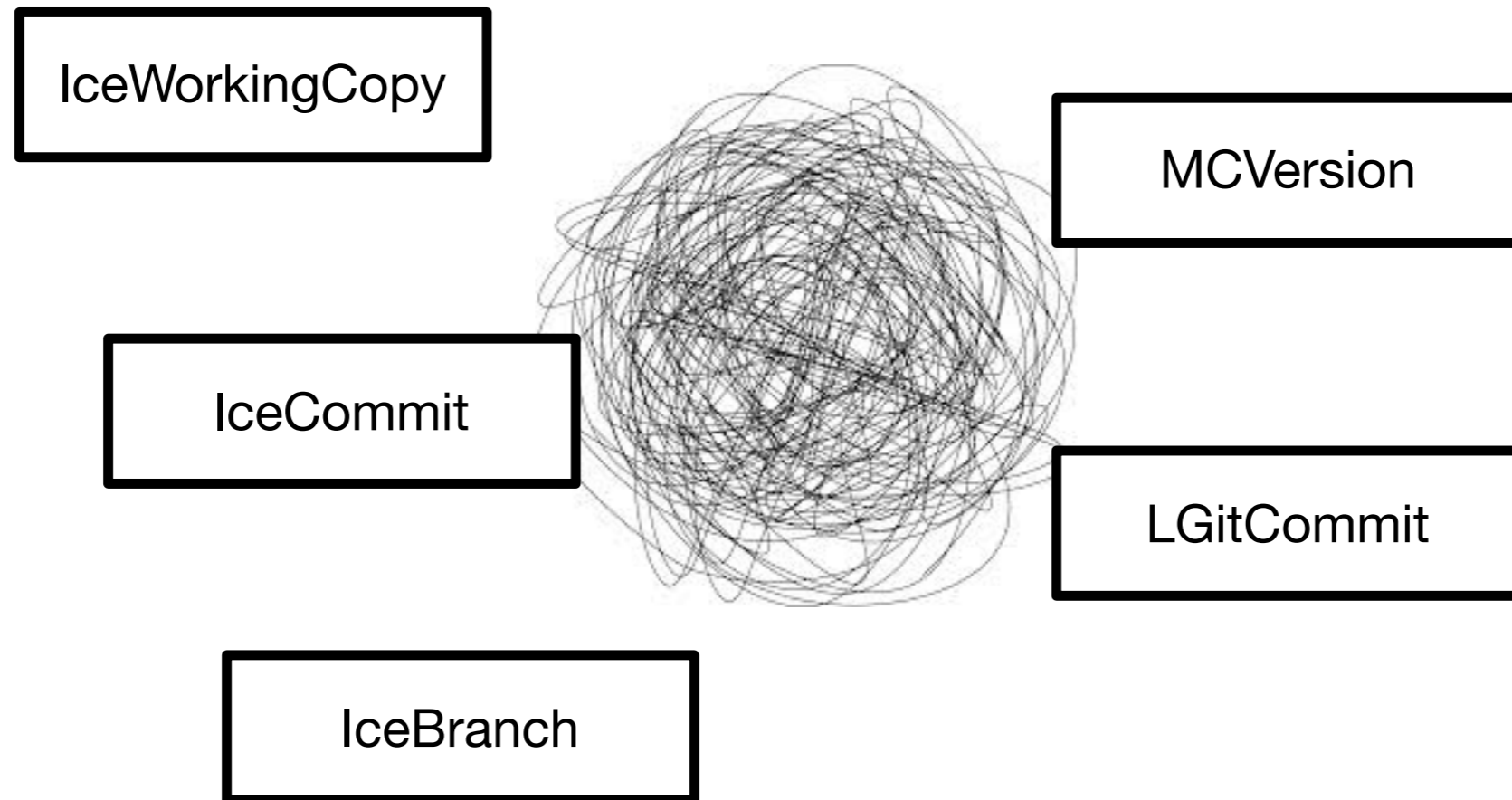
Let's talk about layers



Chapter I: Iceberg

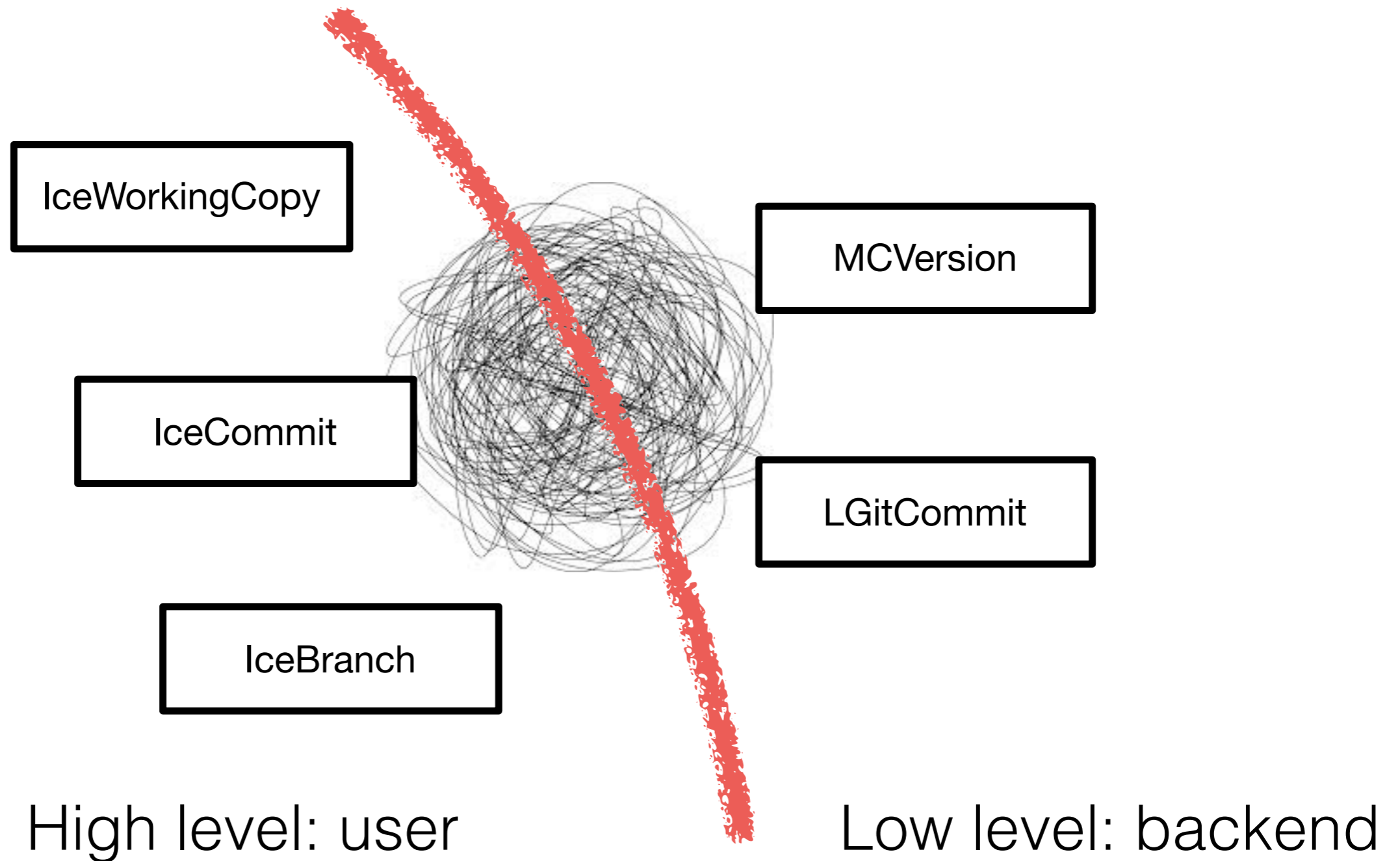


Chapter I: Iceberg



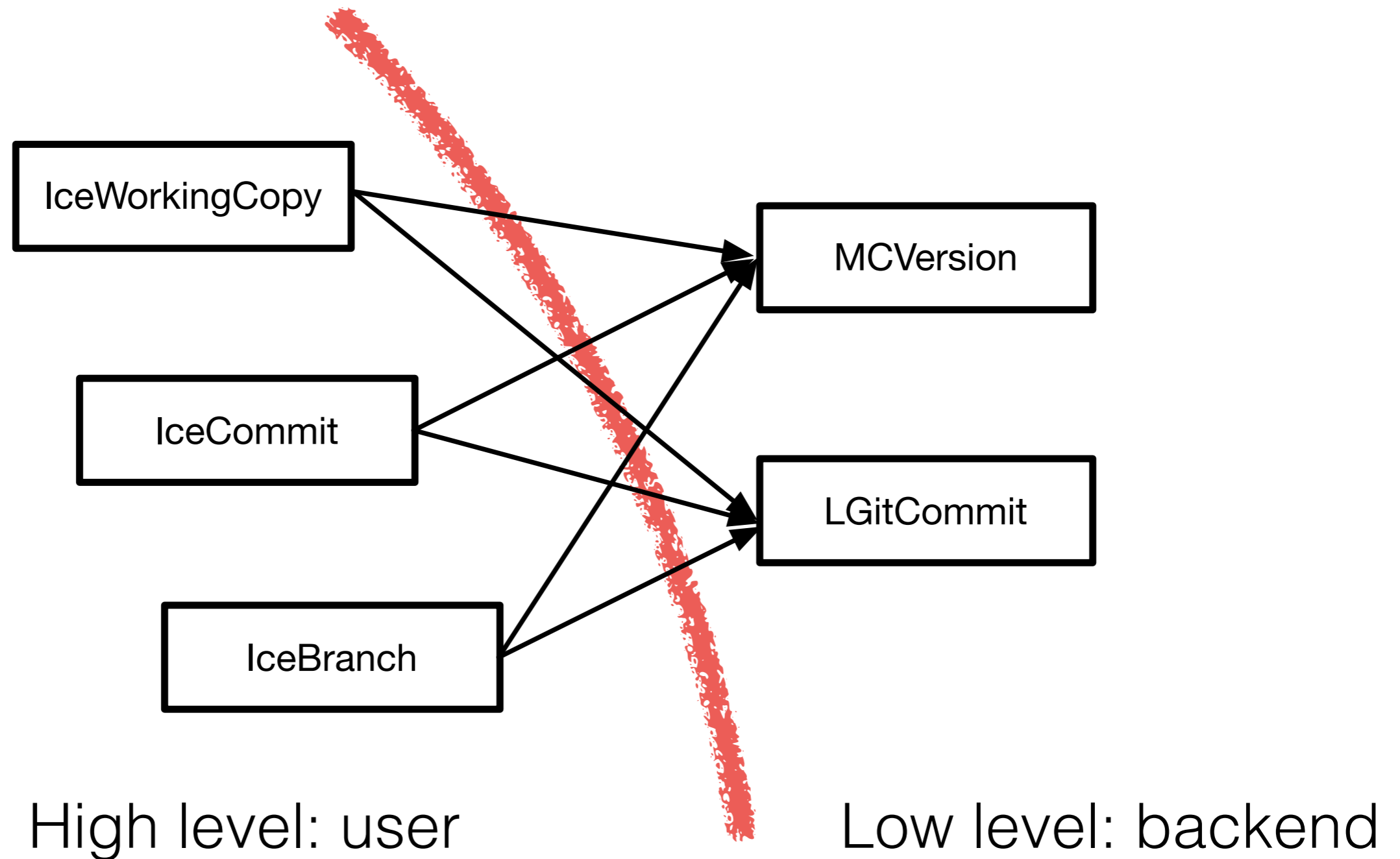


Chapter I: Iceberg





Chapter I: Iceberg





Chapter I: Iceberg

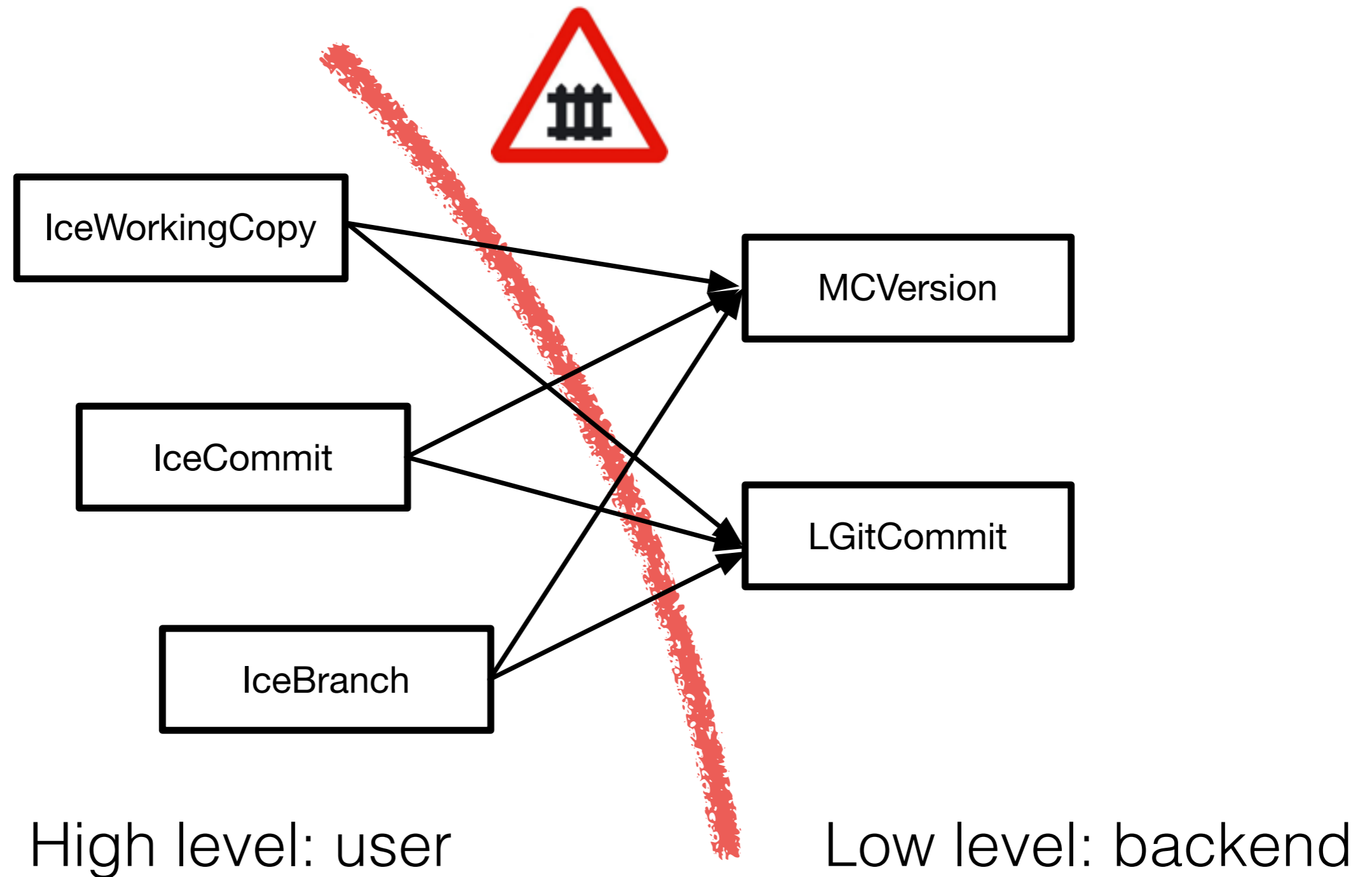
Layers separate
levels of
abstraction

High level: user

Low level: backend

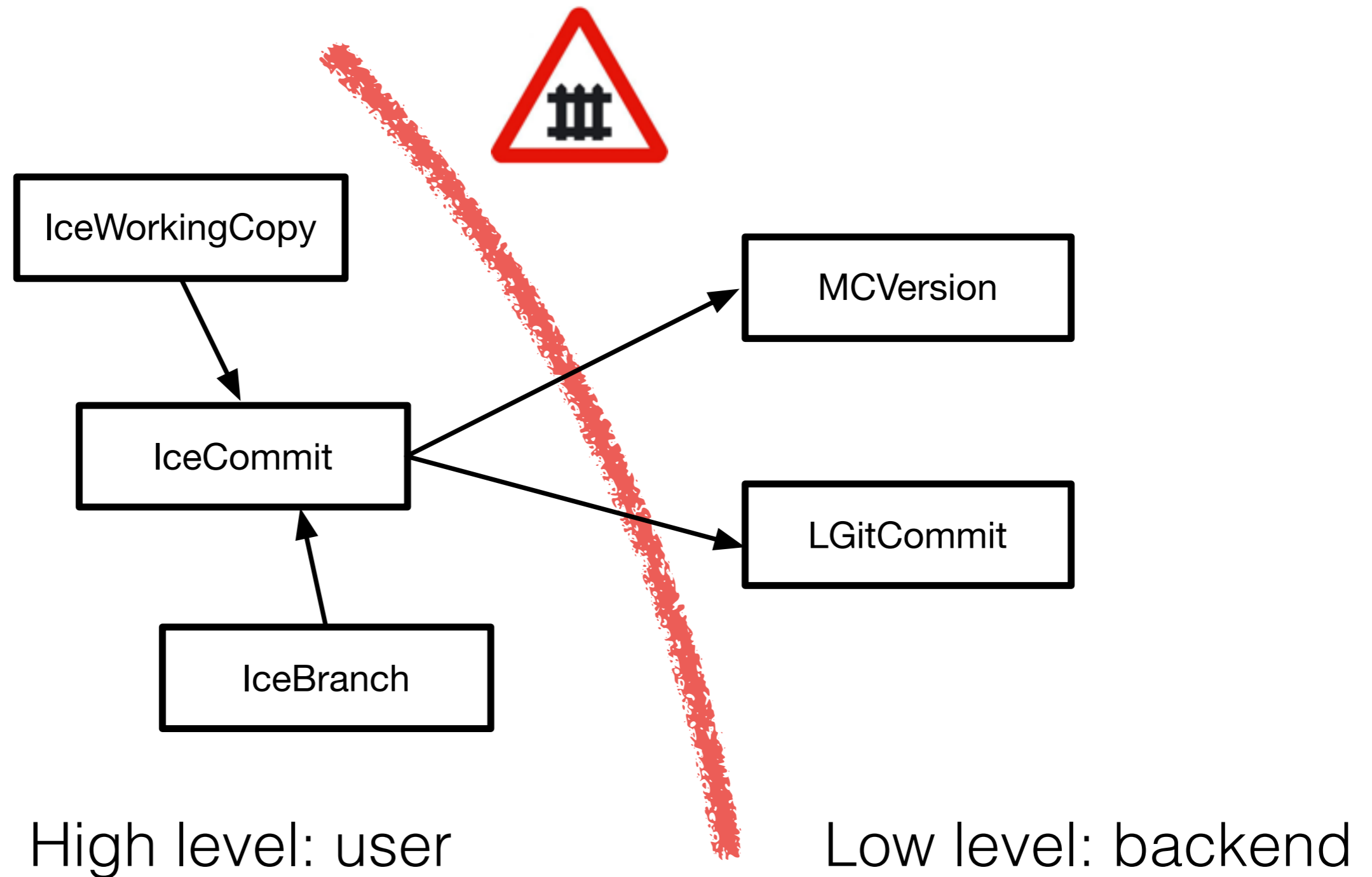


Chapter I: Iceberg



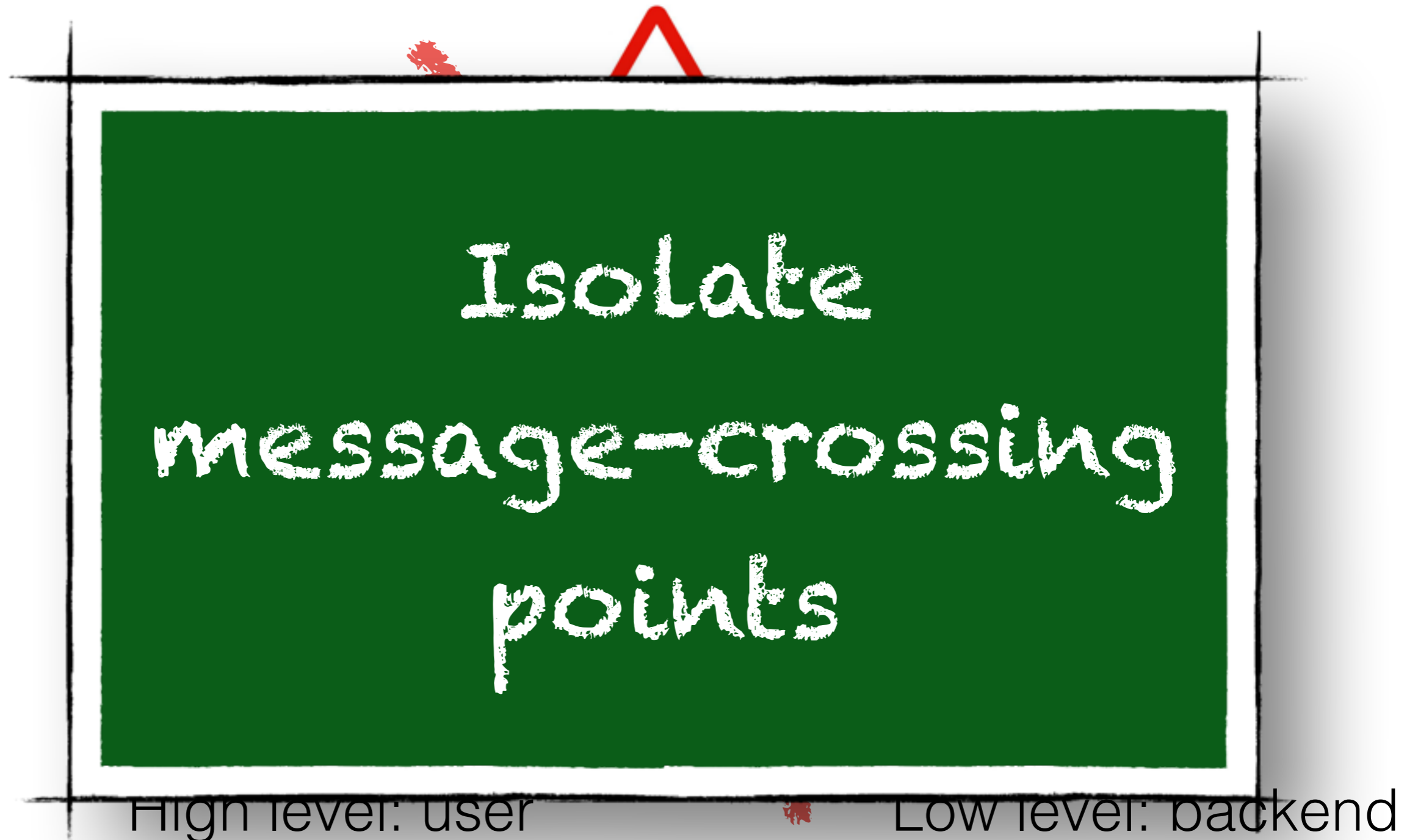


Chapter I: Iceberg

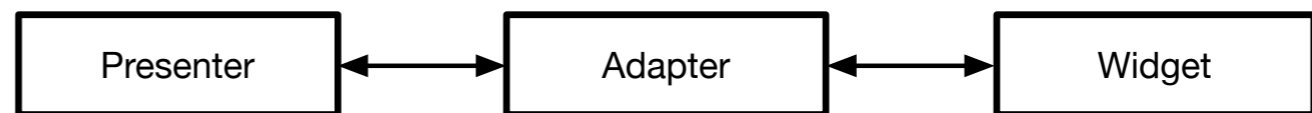




Chapter I: Iceberg



But several layers,
there will be...



Chapter II: Spec

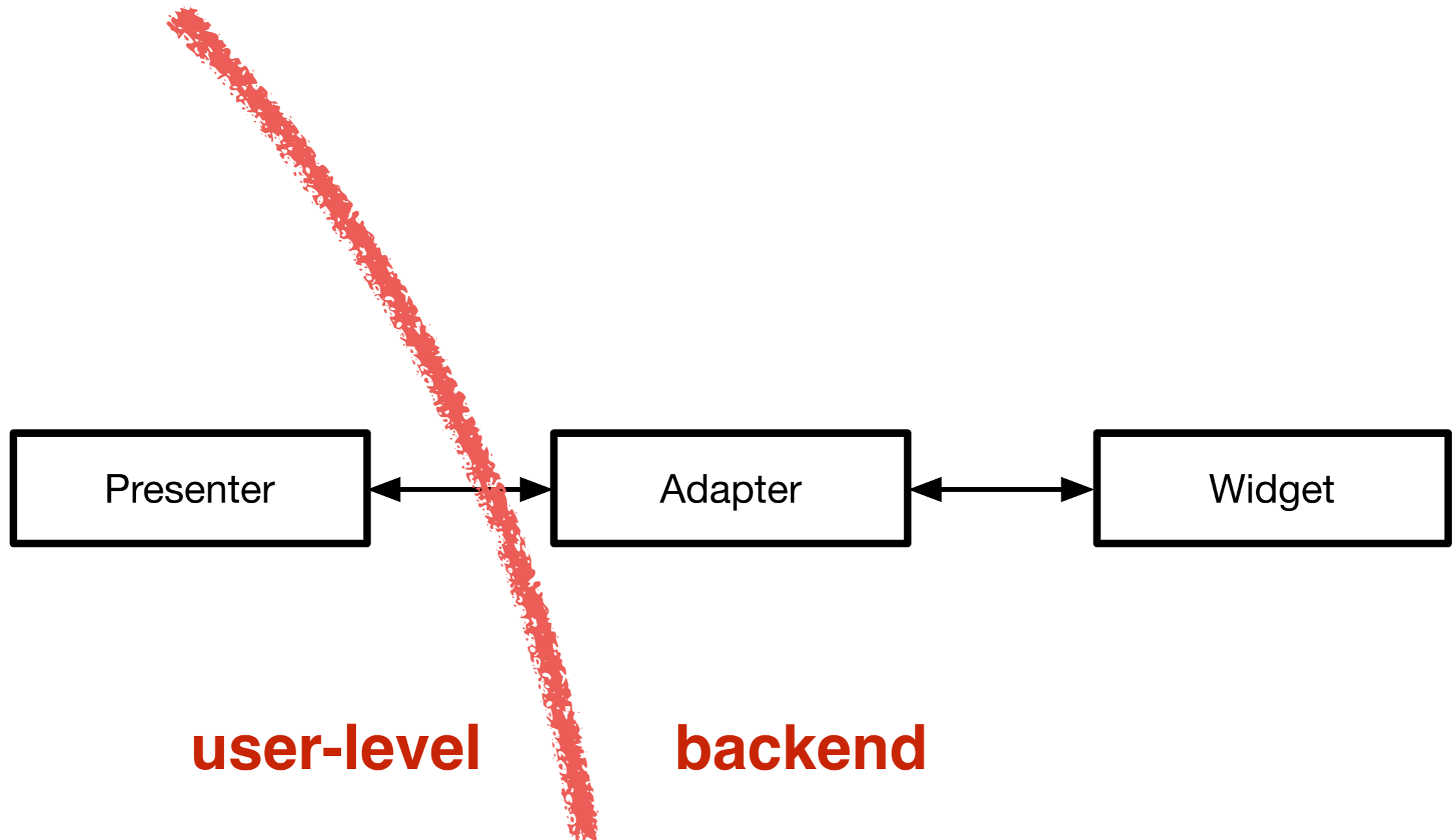


Chapter II: Spec



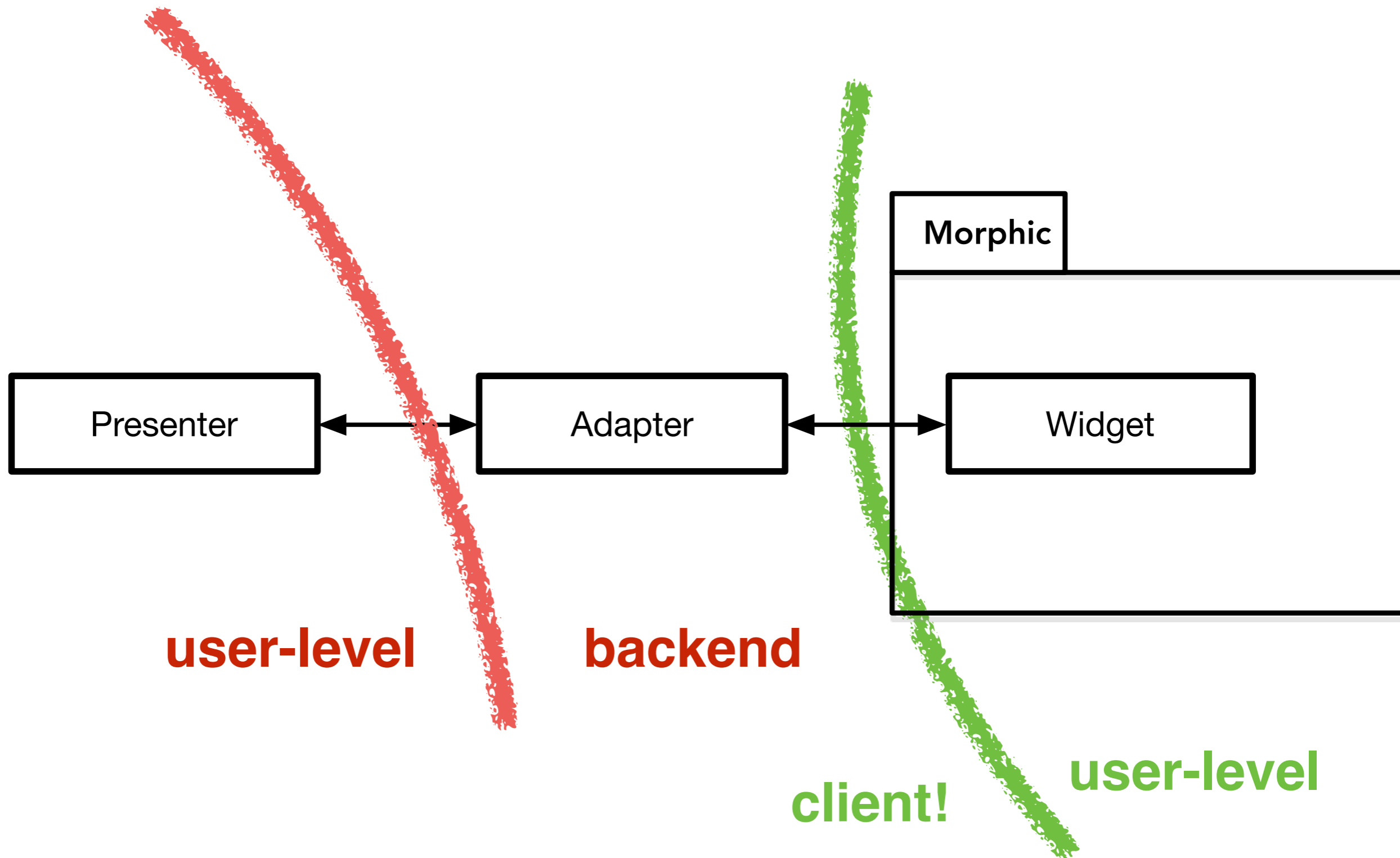


Chapter II: Spec



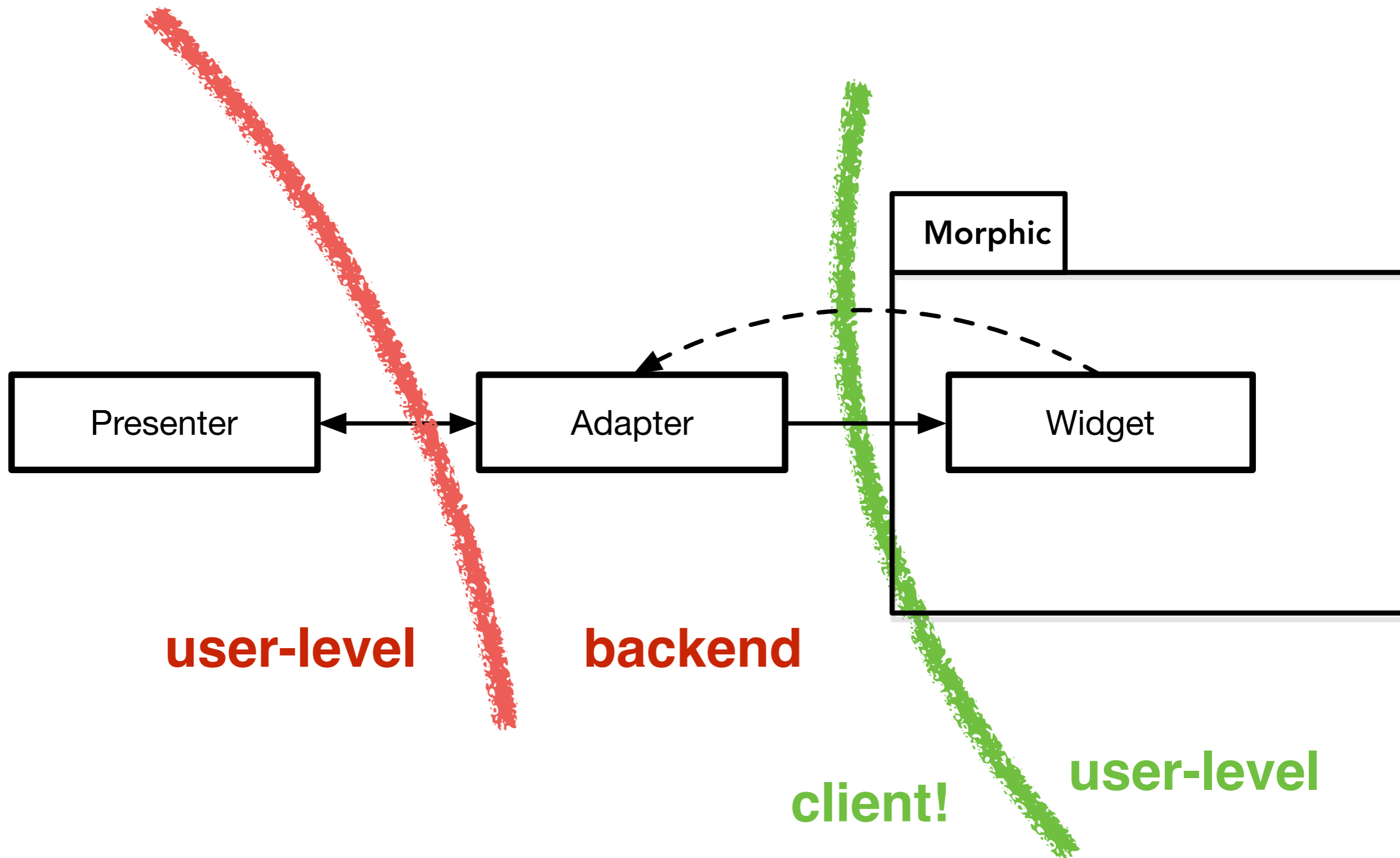


Chapter II: Spec





Chapter II: Spec





Chapter II: Spec

Loose coupling
via
event-messages

Pre

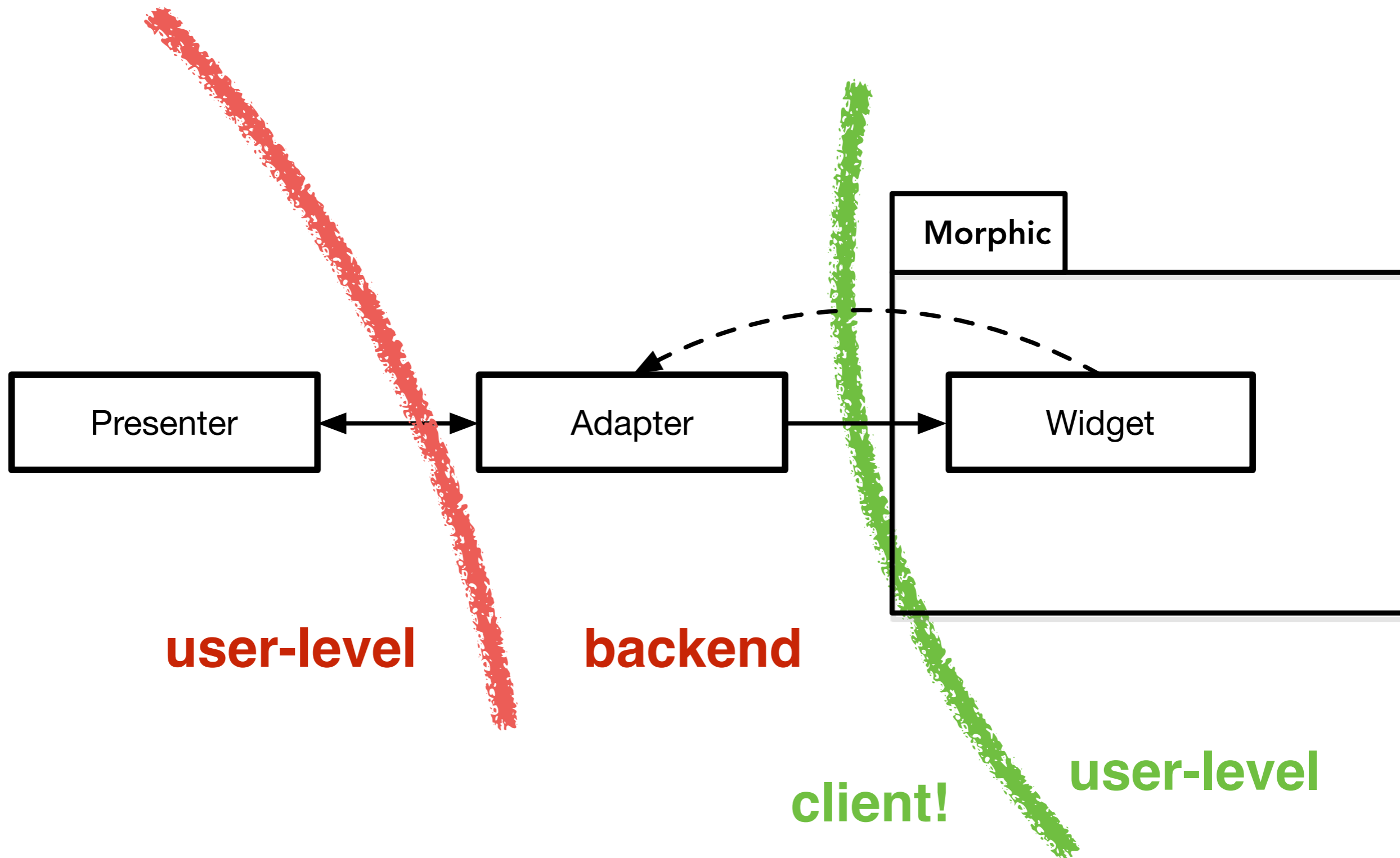
t

client!

user-level



Chapter II: Spec





Chapter II: Spec

Layers separate
state
management

Pre

t

stateful

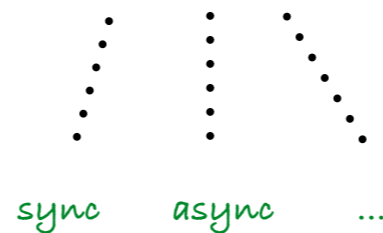
user!

user-level

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...]
```

sync

async

...

Explicit



Chapter III: TaskIt

Convenience

****Thin**** convenience layers

on top of explicit layers

syn



Three lessons of Architecture crafting

Guille Polito
@guillep

CNRS - UMR9189
CRISAL



separate by levels of abstraction

isolate points of failure

separate stateful from stateless

be explicit, as less magic as possible

