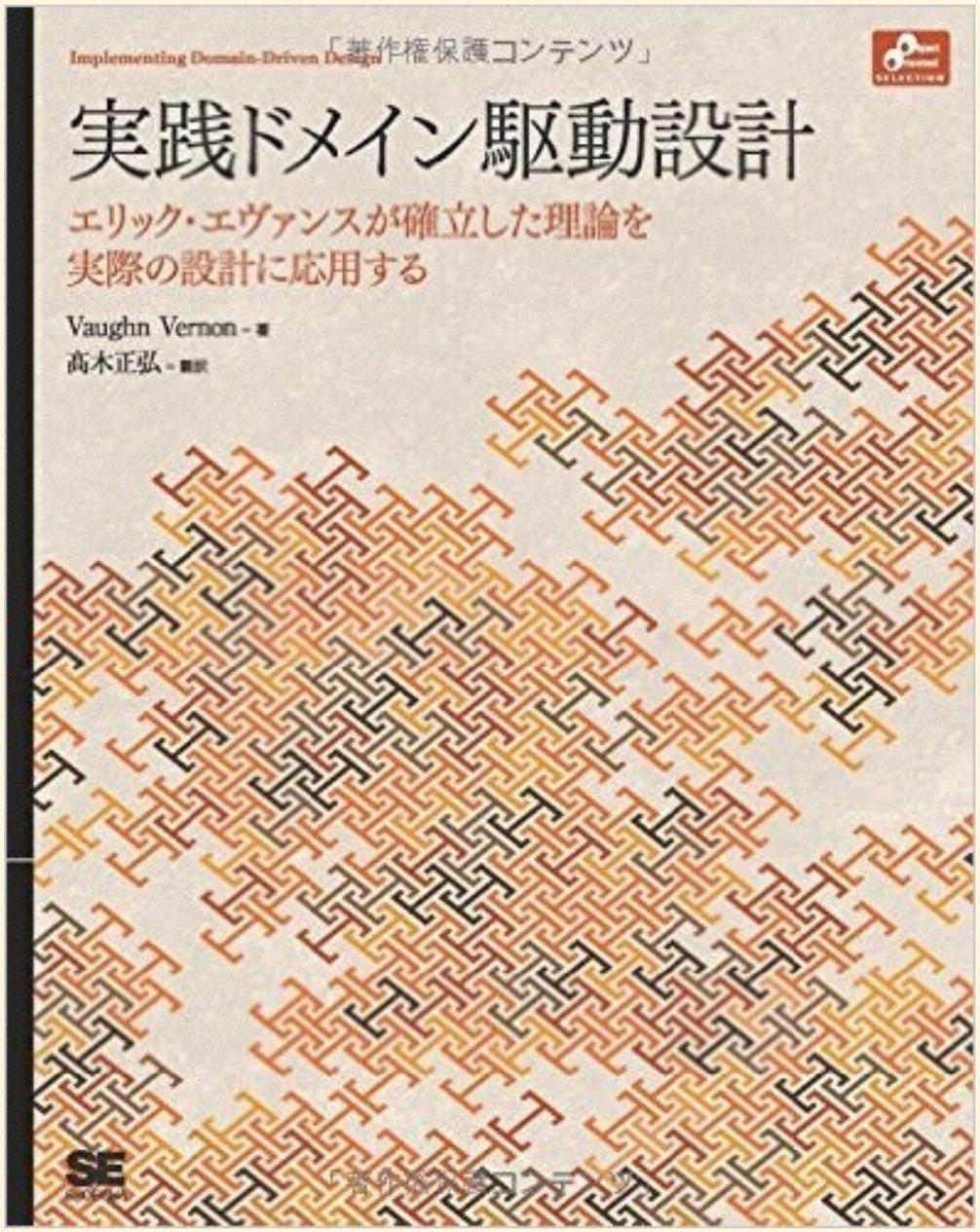
DDD + Flux?

@amagitakayosi



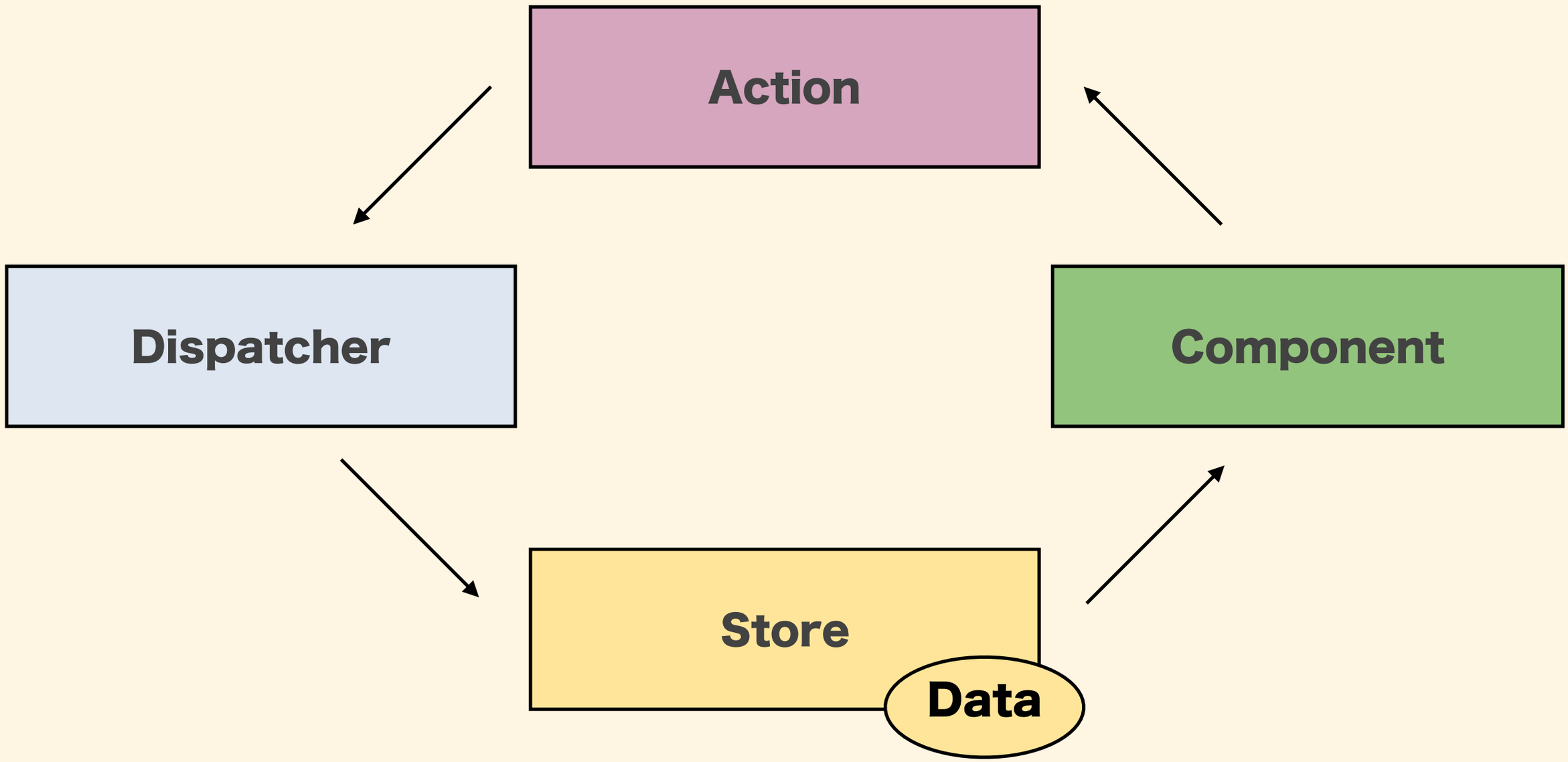
Implementing Domain Driven Design

(˘㊡˘) 。o(Fluxでは？)

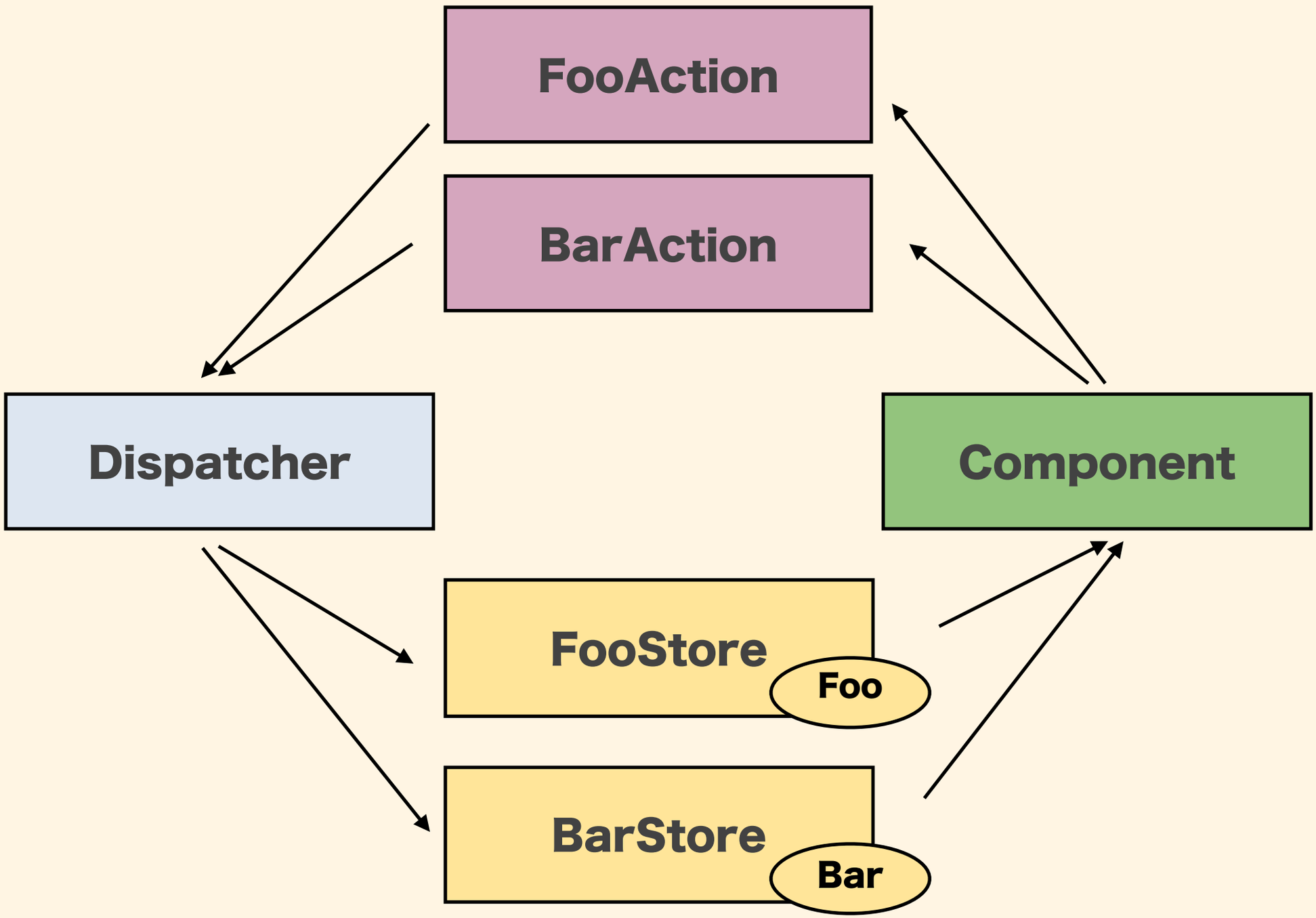
と思ったことを紹介します

Flux

Flux



Storeが増えると...

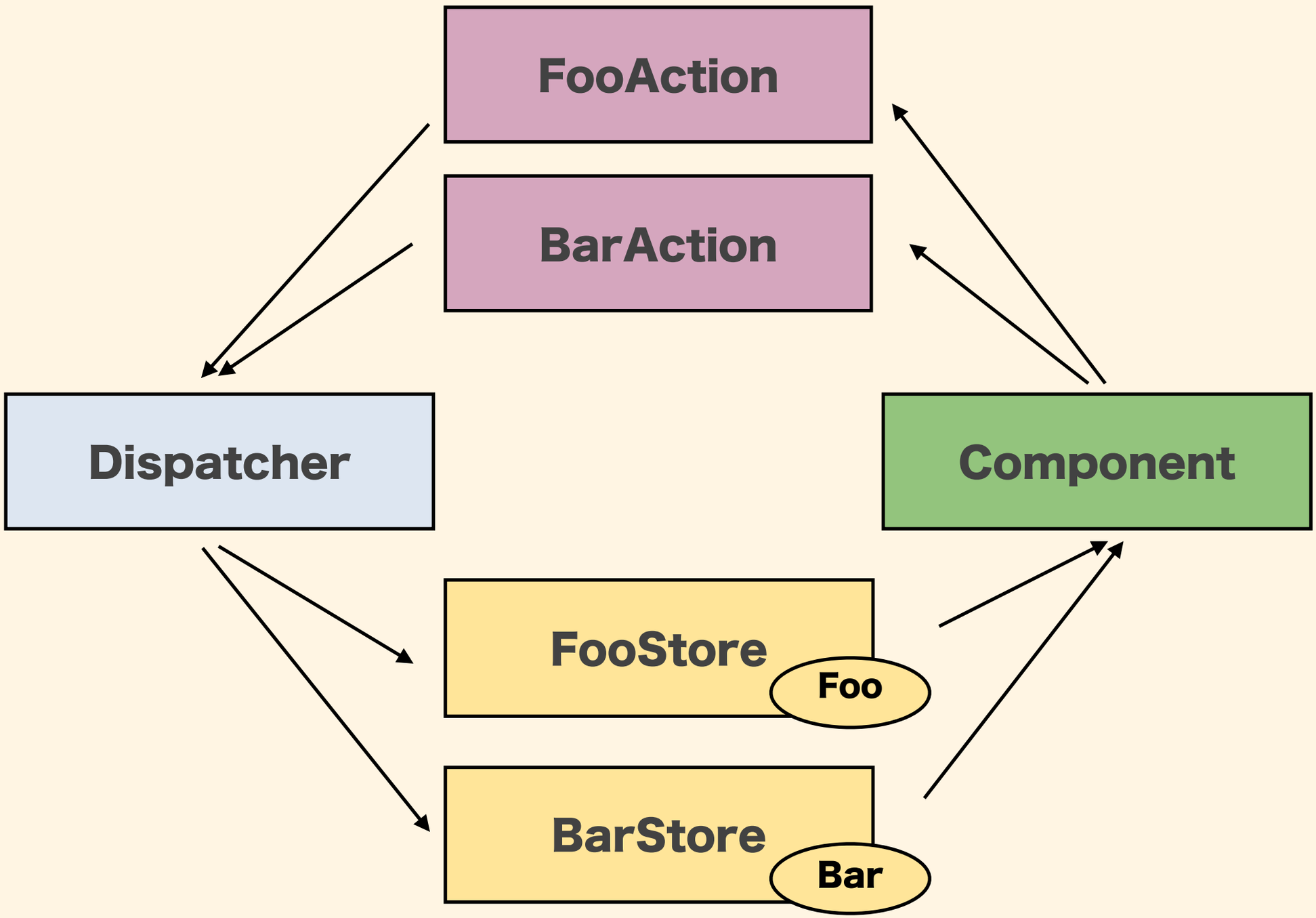

FooAction
BarAction
Dispatcher
Component
FooStore
Foo
BarStore
Bar

DDD

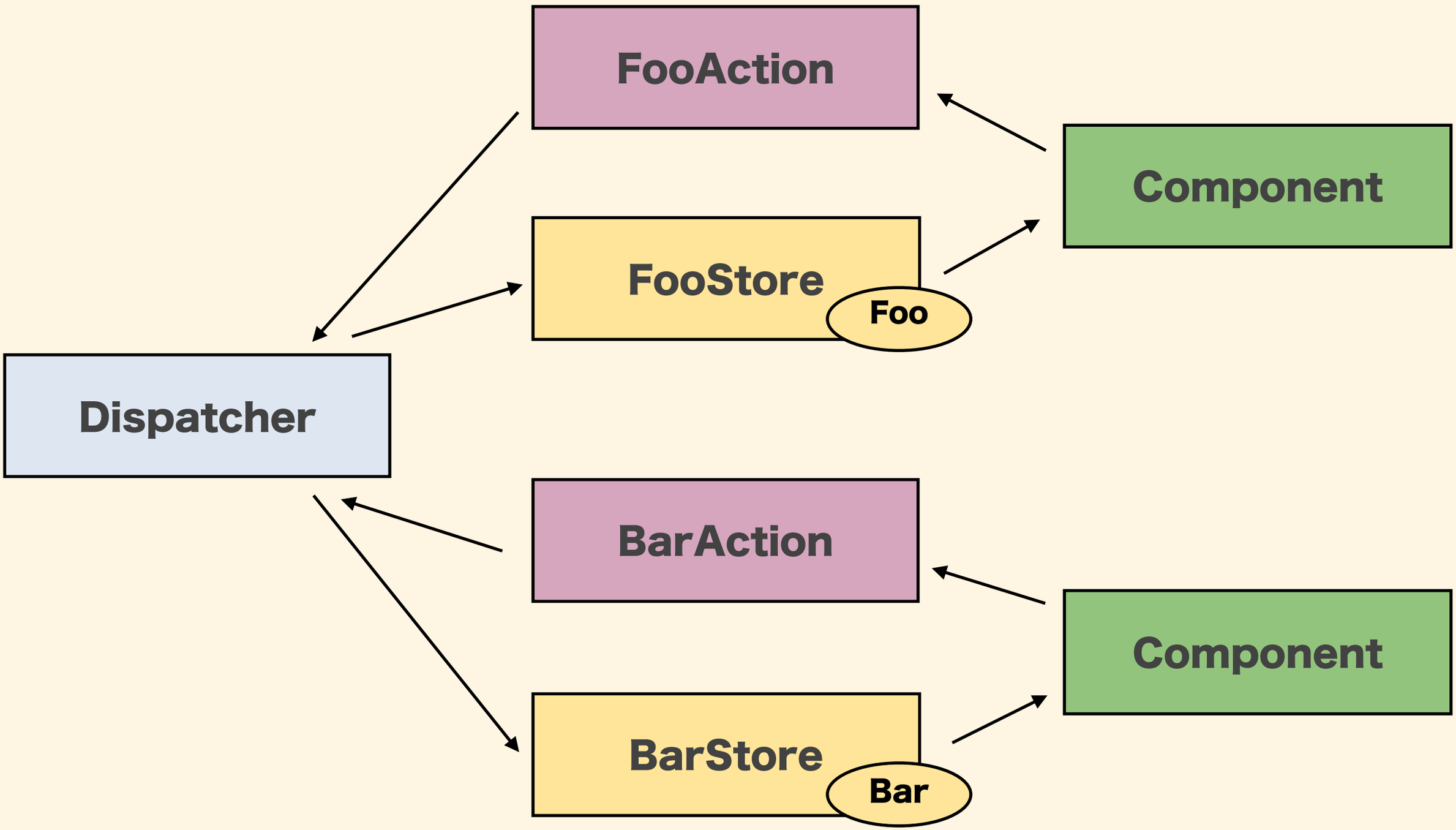
Domain-Driven Design

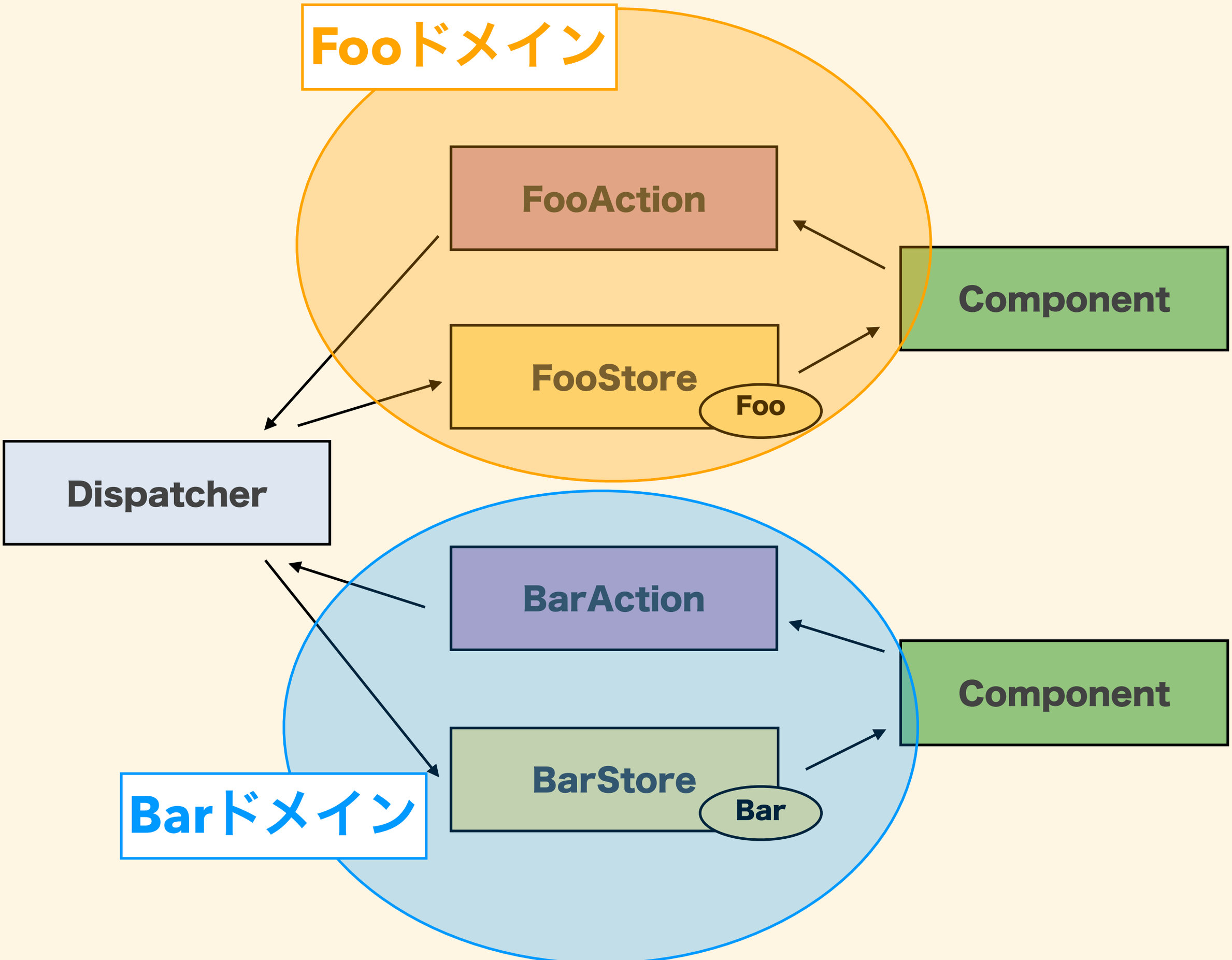
- ビジネスロジックをドメインに分割
  - アプリケーション = ドメインの集合
- ロジックと実装の分離を重視

Storeが増えると...


FooAction
BarAction
Dispatcher
Component
FooStore
Foo
BarStore
Bar

Storeが増えると...


FooAction
Component
FooStore
Foo
Dispatcher
BarAction
Component
BarStore
Bar


Fooドメイン
FooAction
FooStore
Foo
Component
Dispatcher
BarAction
BarStore
Bar
Component
Barドメイン

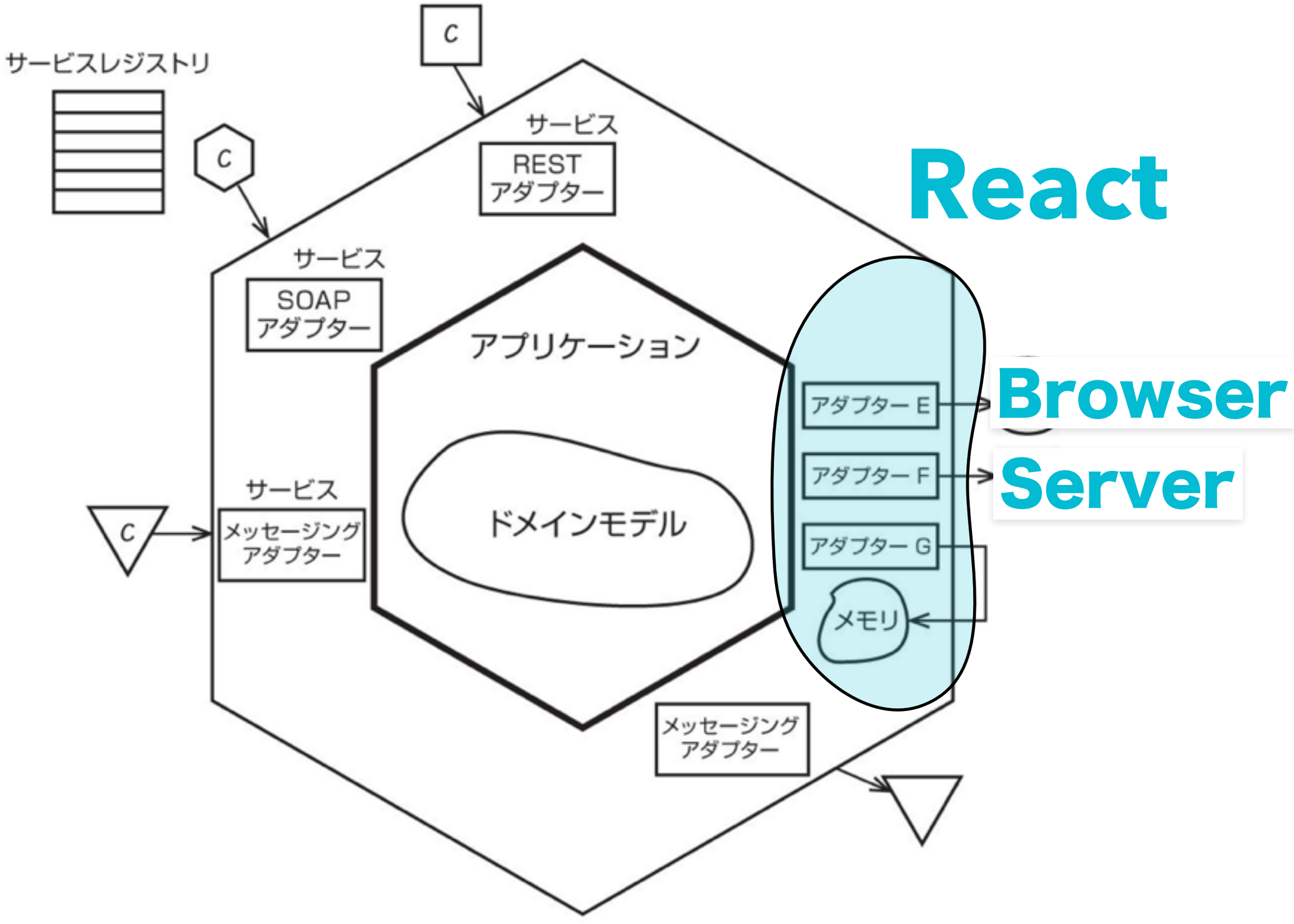


図4-5：SOA やREST、SOAP、メッセージングに対応するヘキサゴナルアーキテクチャ

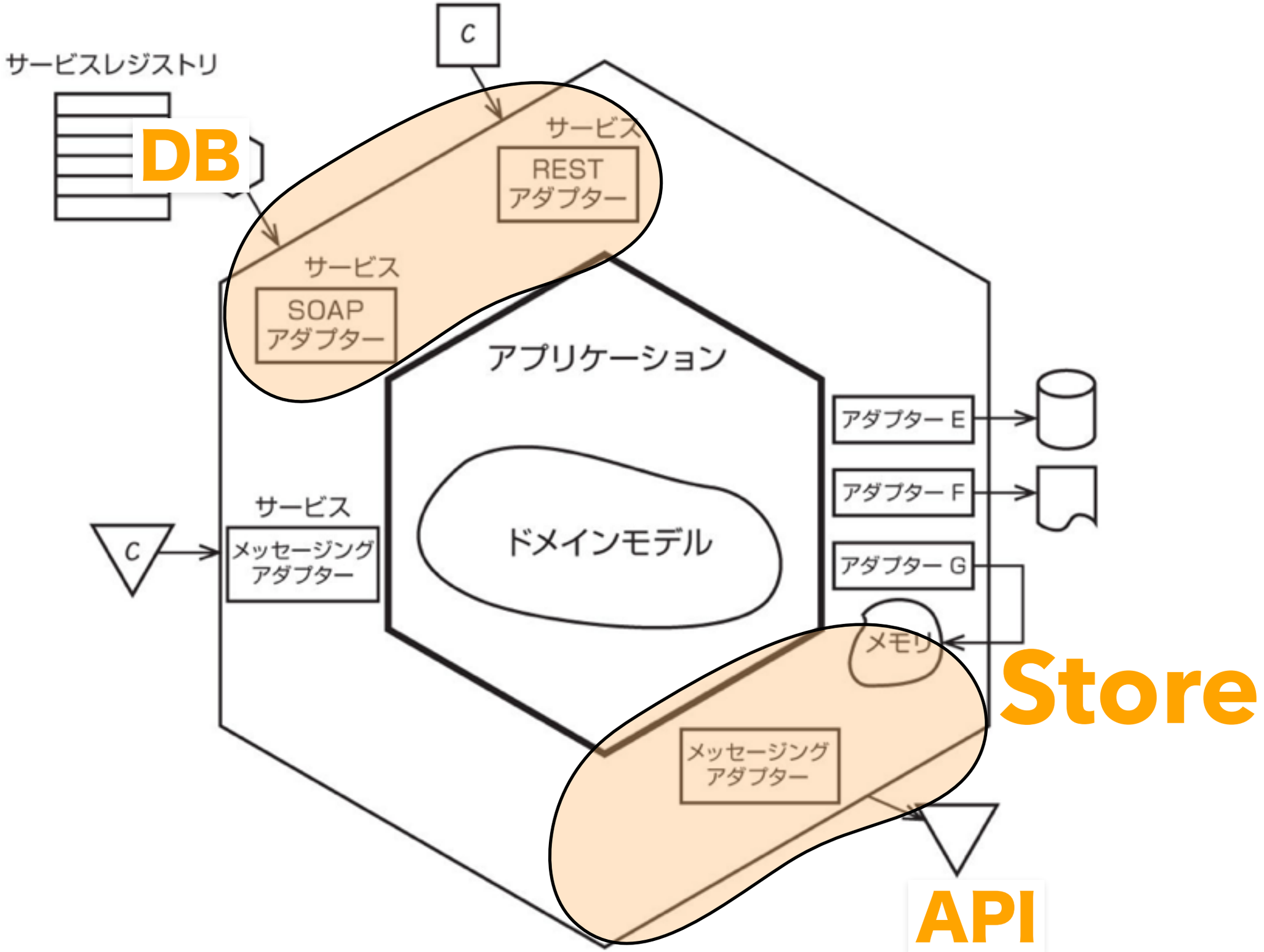
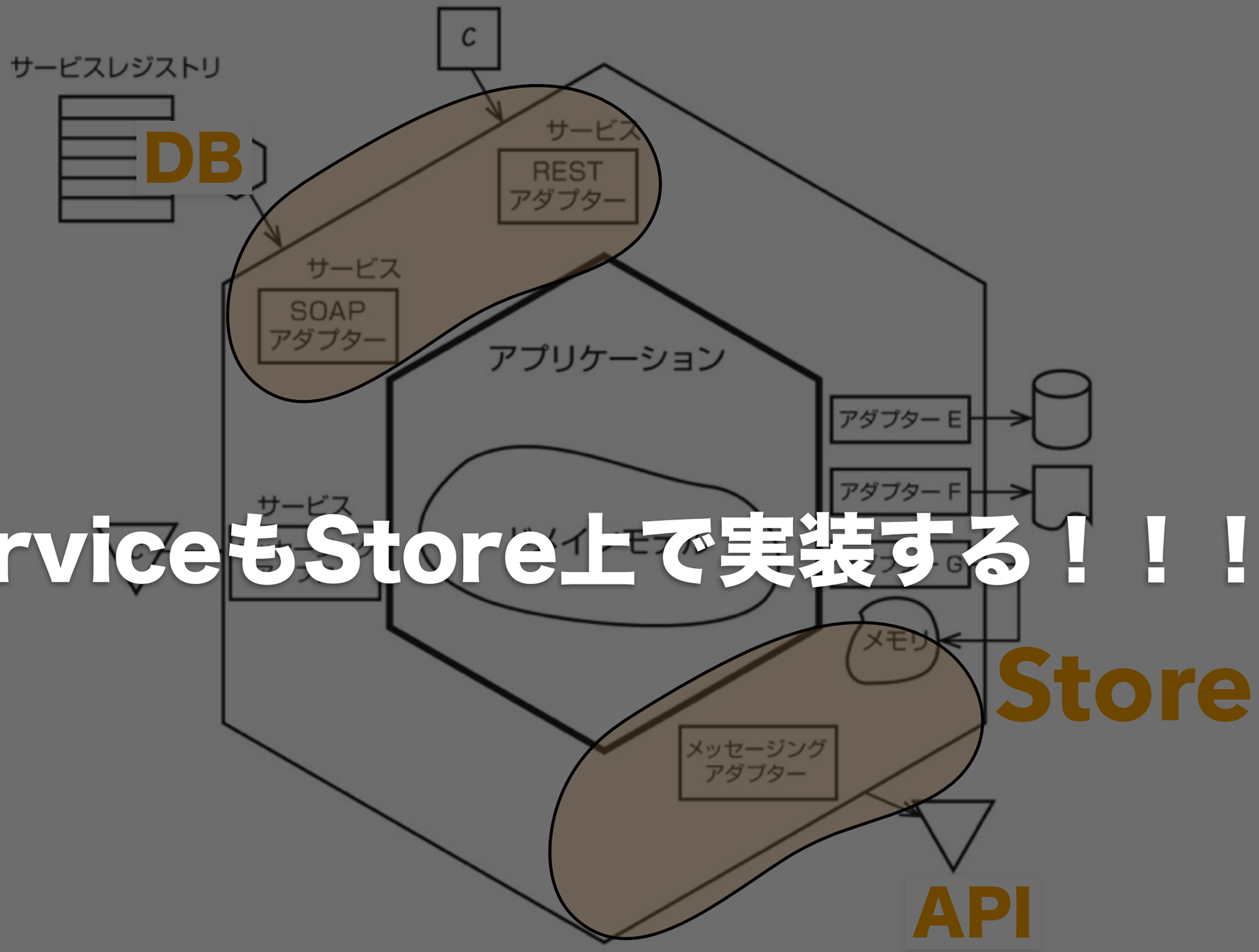


図4-5：SOA やREST、SOAP、メッセージングに対応するヘキサゴナルアーキテクチャ



ServiceもStore上で実装する！！！

図4-5：SOA やREST、SOAP、メッセージングに対応するヘキサゴナルアーキテクチャ

CORS

Command Query Responsibility Segregation

- 直訳すると「コマンドクエリ責任分離」
    - コマンド : 状態を変更する
    - クエリ : 状態を取得する
- 状態の取得方法を制限し、考える事を減らす

FluxとCQRS

- Action ≒ Command ?
  - **Storeは、更新のないActionを無視する**
- Store.getState() ≒ Query ?
  - **Storeを直接変更するのは 🙅‍♀️**

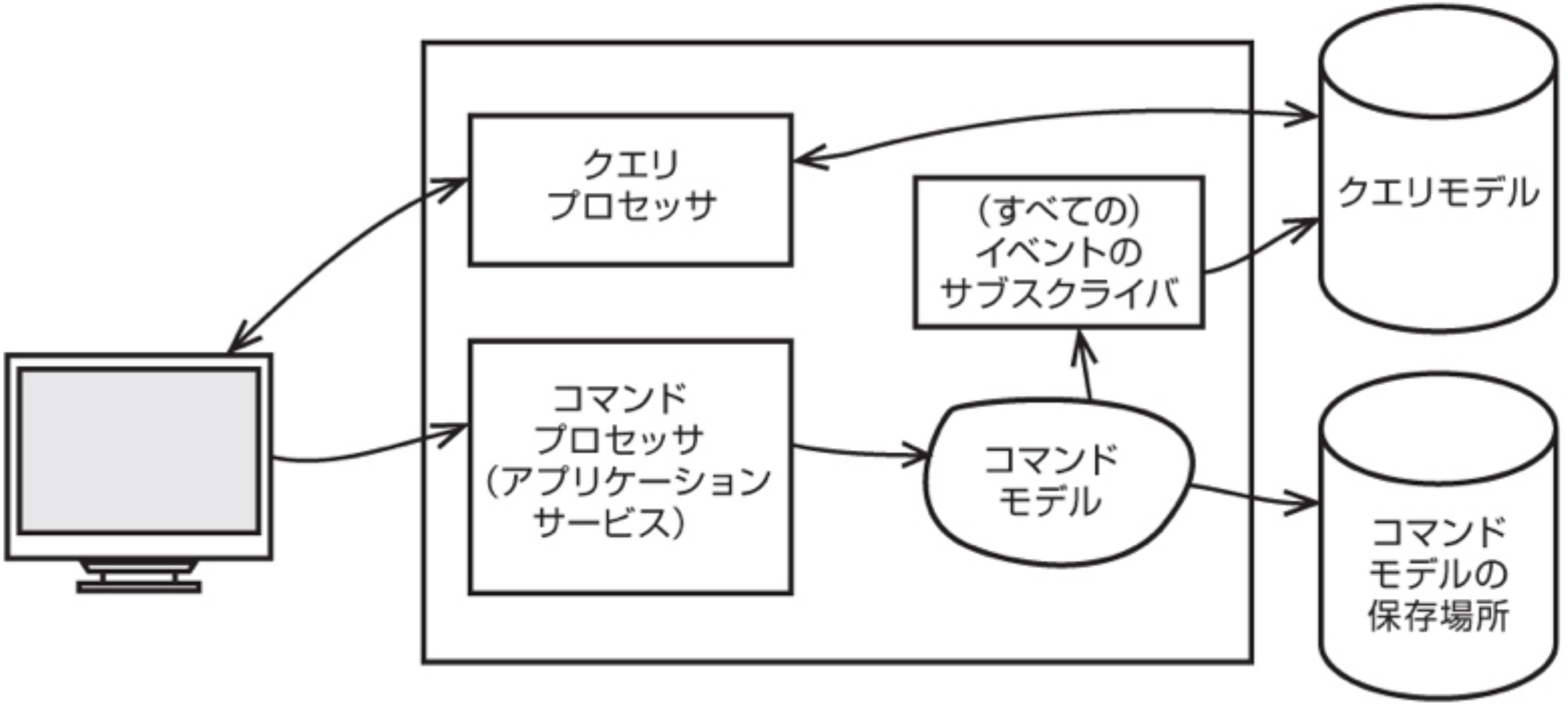


図4-6：CQRS では、クライアントからのコマンドは一方通行でコマンドモデルに届く。クエリは別のデータソースに対して発行する。このデータソースは表示に最適化されており、結果はユーザーインターフェイスや帳票向けに送られる

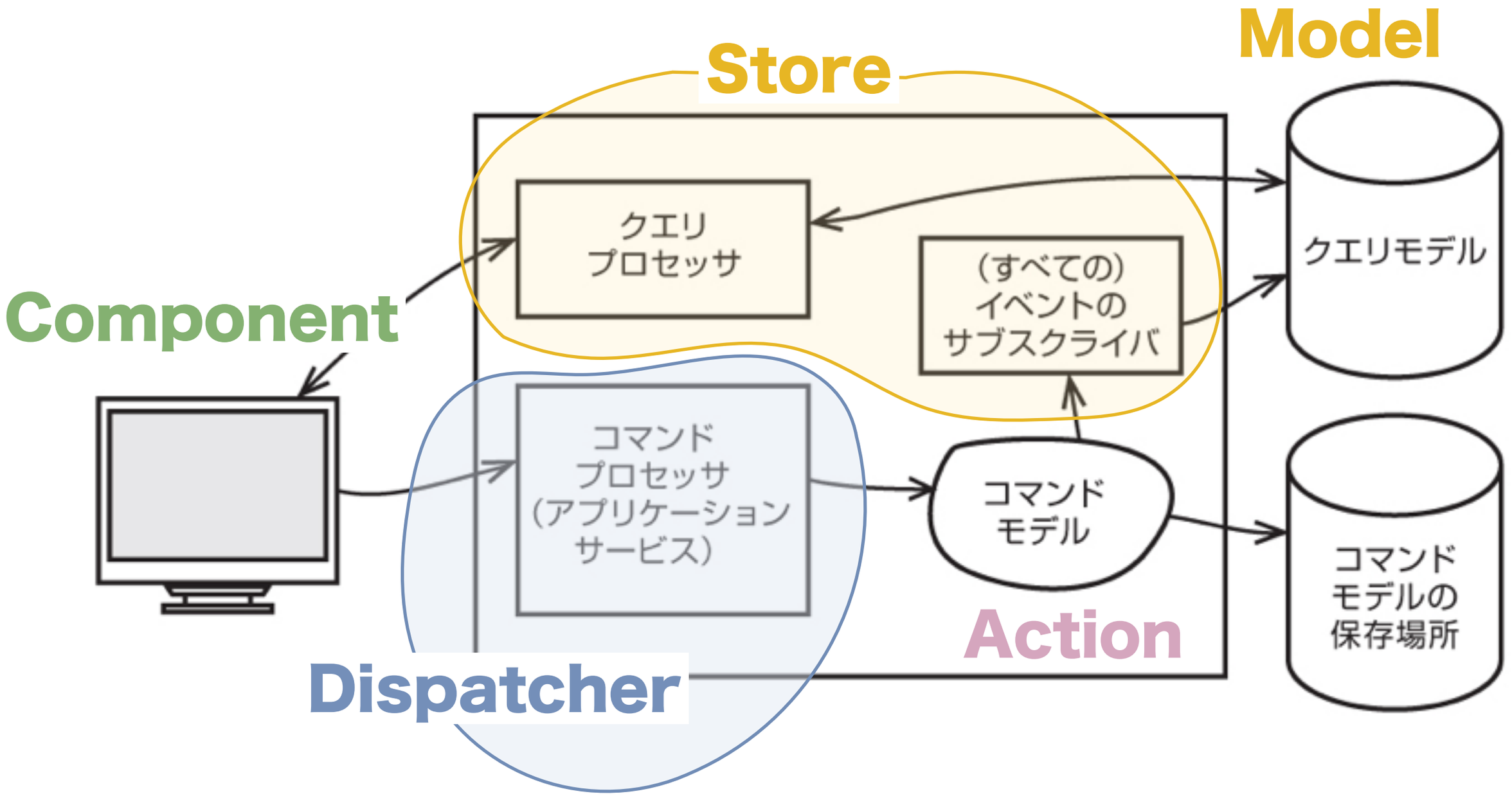


図4-6：CQRS では、クライアントからのコマンドは一方通行でコマンドモデルに届く。クエリは別のデータソースに対して発行する。このデータソースは表示に最適化されており、結果はユーザーインターフェイスや帳票向けに送られる

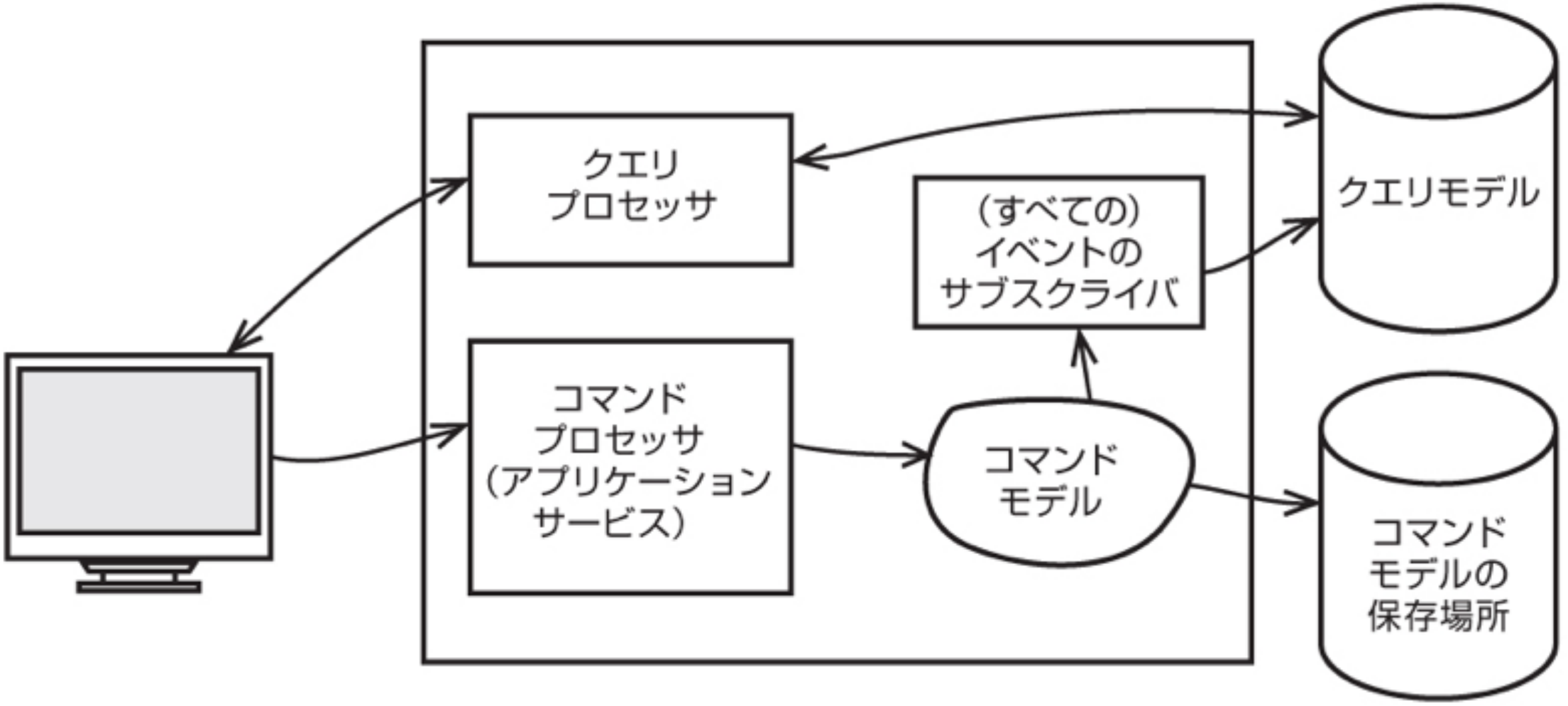
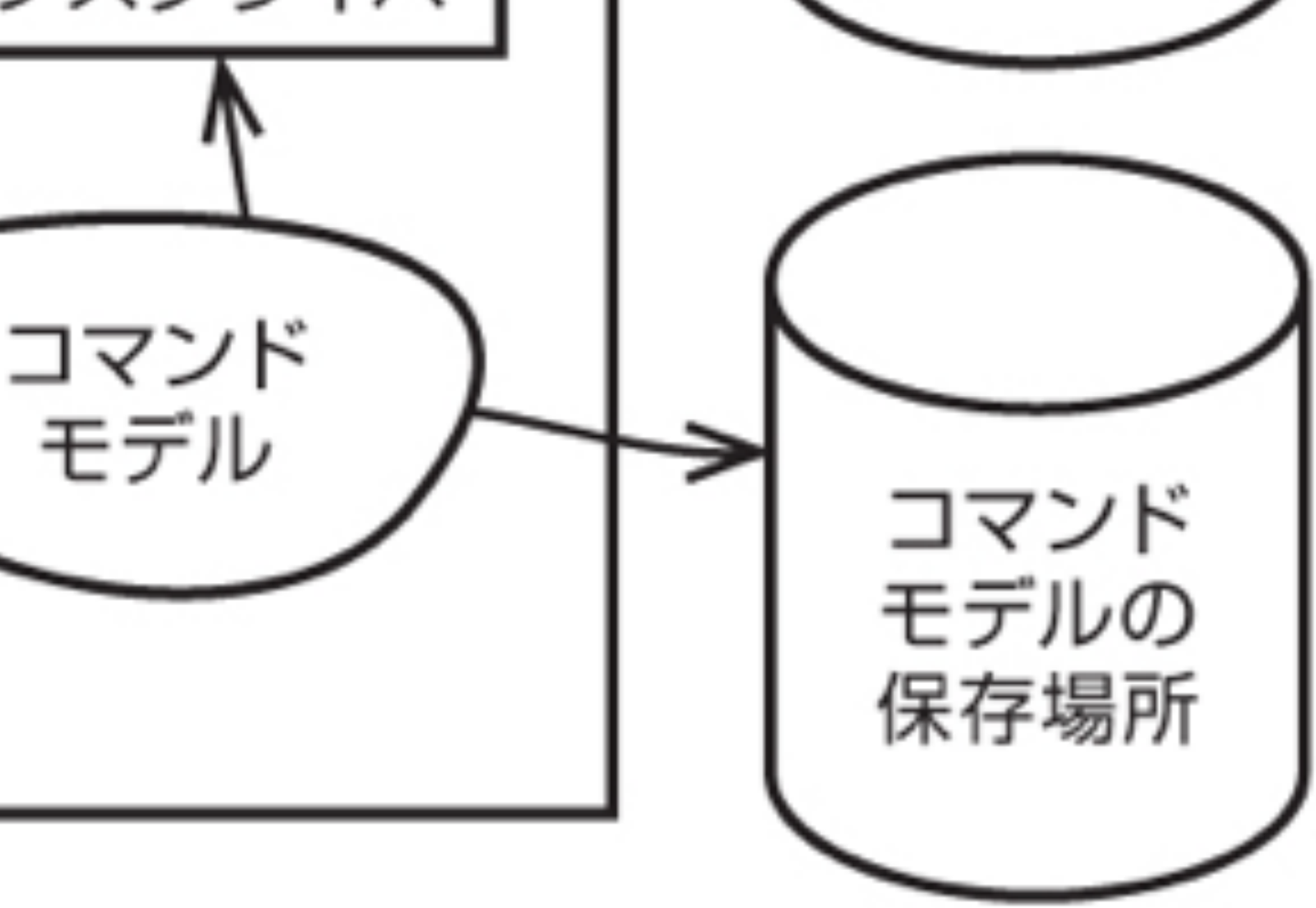


図4-6：CQRS では、クライアントからのコマンドは一方通行でコマンドモデルに届く。クエリは別のデータソースに対して発行する。このデータソースは表示に最適化されており、結果はユーザーインターフェイスや帳票向けに送られる



は一方通行でコマンドモデルに

Event Sourcing

EventSourcing

- 集約をイベントのシーケンスから作る
  - 現在の状態 = 初期状態 + イベント
- Commandのみ保存すれば良い
  - Actionを記録していればおｋ

Dispatcherで記録してみる

- facebook/flux の Dispatcher を拡張
- イベントと時刻を記録
- replay機能

```
class LoggedDispatcher extends Dispatcher {

  constructor () {
    super();
    this.eventStore = [];
  }

  dispatch (event) {
    this.eventStore.push({
      time  : Date.now(),  // 時刻を記録
      event : event,
    });
    super.dispatch(event);
  }

```

```
replay (time) {

  // 初期状態に戻す
  super.dispatch({ type: 'INIT' });

  // 指定時刻以前のイベントを発行
  this.eventStore.forEach((e) => {
    if (e.time < time) {
      super.dispatch(e.event);
    }
  });
```

デモ

http://gmork.in/rec-act/

まと

まとめ

- Store = ドメインと実装をつなぐポート
- EntityはStoreに置く
  - Redux等だとEntity持ちにくい
- **みんなでStore構造化してこ！！！！**