Clean Architecture

Chapter14 元件耦合性

Bear

第14章

元件耦合性

(Component Coupling)

A **無環依賴原則 (ADP)**

在元件的依賴關係圖中不允許出現環。

B **穩定依賴原則 (SDP)**

朝著穩定的方向進行依賴。

C **穩定抽象原則 (SAP)**

元件的抽象程度應該與元件的穩定程度一致。

無環依賴原則 (ADP)

(Acyclic Dependencies Principle)

無環依賴原則 (ADP)

在元件的依賴關係圖中不允許出現環。

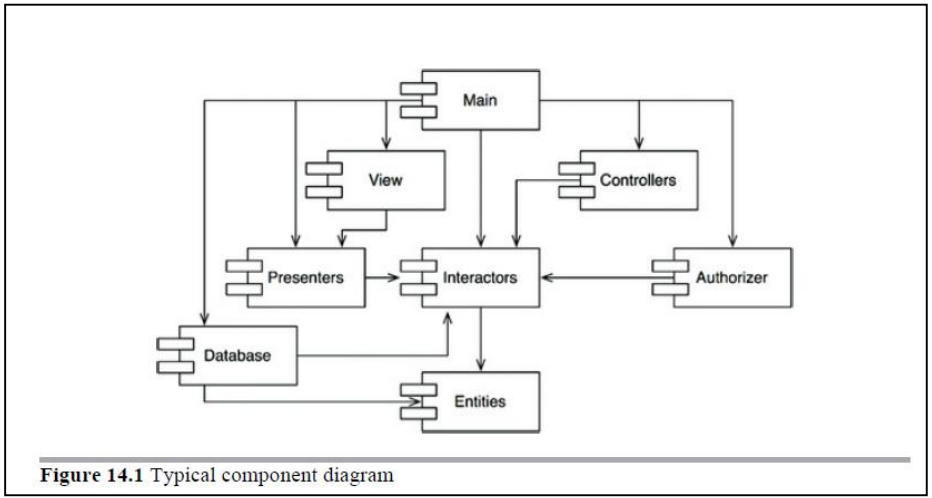


Figure 14.1 Typical component diagram

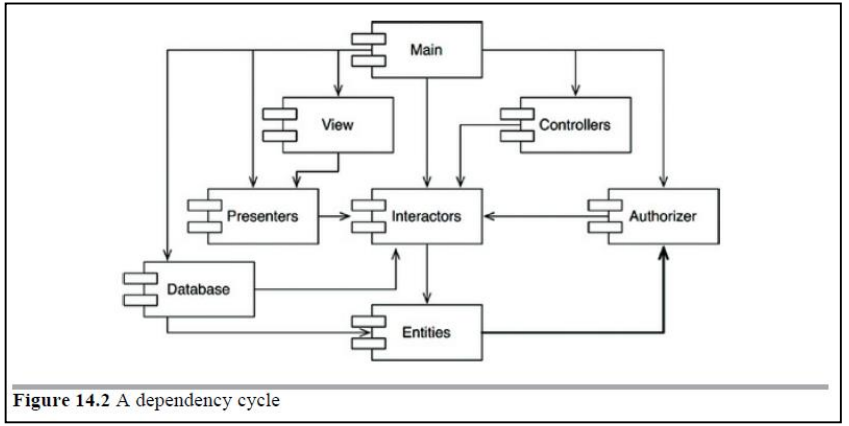


Figure 14.2 A dependency cycle

UML 箭頭說明

無環依賴原則 (ADP)

在元件的依賴關係圖中不允許出現環。

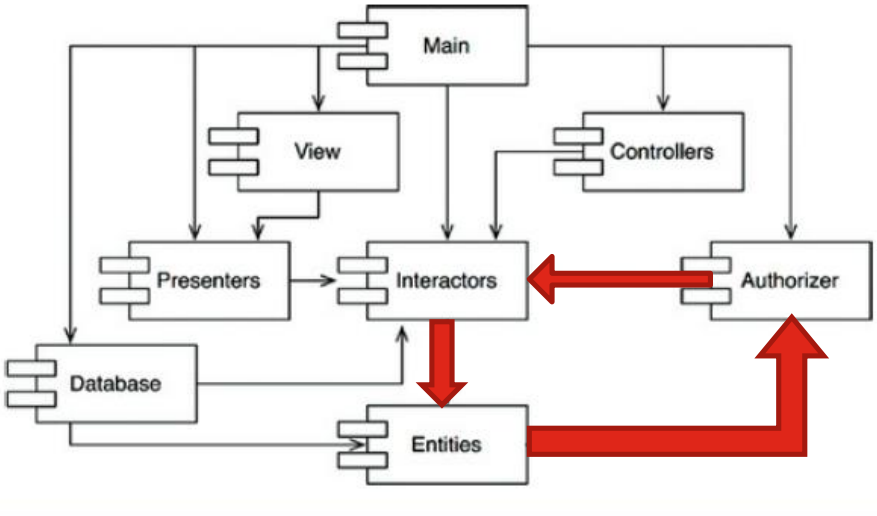


Figure 14.2 A dependency cycle

無環依賴原則 (ADP)

在元件的依賴關係圖中不允許出現環。

Solution.

1. 利用**依賴反向原則 (DIP)**
2. 建立新元件，改變依賴結構

無環依賴原則 (ADP)

在元件的依賴關係圖中不允許出現環。

before.

after.

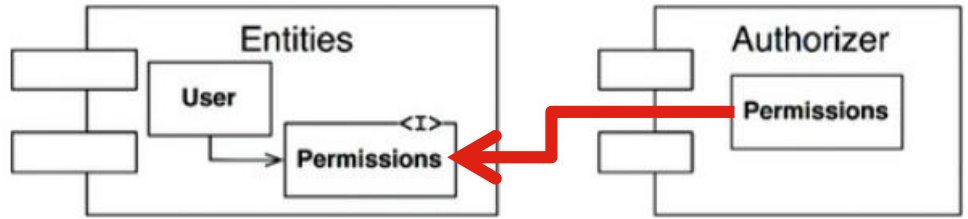


Figure 14.3 Inverting the dependency between `Entities` and `Authorizer`

利用依賴反向原則 (DIP)

無環依賴原則 (ADP)

在元件的依賴關係圖中不允許出現環。

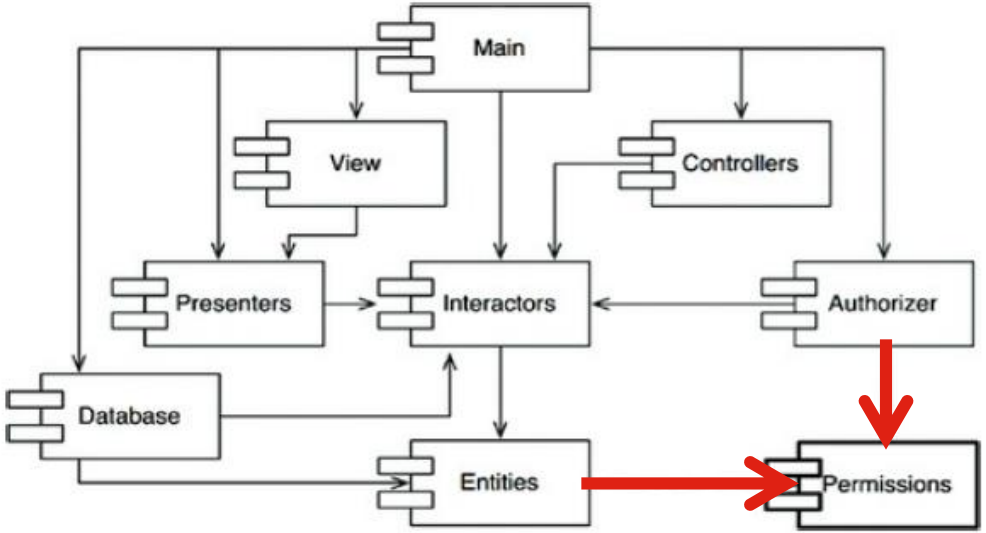


Figure 14.4 The new component that both `Entities` and `Authorizer` depend on

建立新元件，改變依賴結構

無環依賴原則 (ADP)

在元件的依賴關係圖中不允許出現環。

目標：

- 隔離**易變性**。
- 提高**可建置性 (buildability)** 和**可維護性 (maintainability)**。

穩定依賴原則 (SDP)

(Stable Dependencies Principle)

穩定依賴原則 (SDP)

朝著穩定的方向進行依賴。

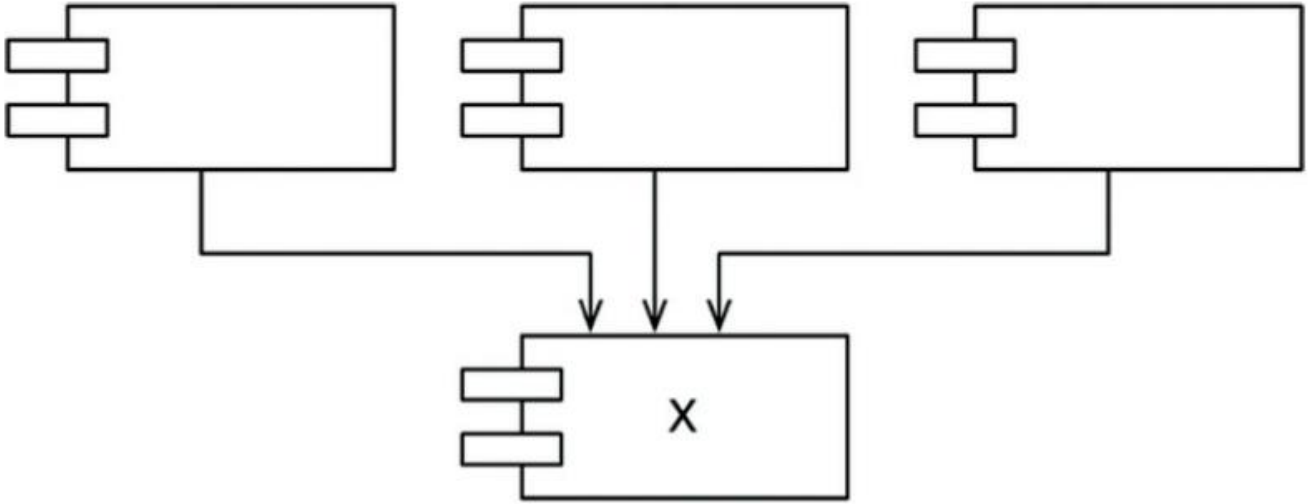


Figure 14.5 x: a stable component

有3個元件依賴X，而X不依賴於任何元件。

UML 箭頭說明

穩定依賴原則 (SDP)

朝著穩定的方向進行依賴。

Support (支援)

我可能會對一資源的資料做簡單的邏輯判斷，不需要依賴任何的 `資料控制結構` ，像是 `Service` 、 `Repository` 或 `Model` ，所以我會分出一層 `Support (常數)` 去做共用的輔助方法。

像是我們可以用 `PostSupport::getAllPostStatus()` 去撈取文章的所有狀態，或者用 `GoogleAnalyticSupport::api()` 去對 GA 的 API 做存取。

> 在 `Support` 的方法皆為 `靜態方法` ，所以可以供任何類別去做存取。

獨立結構

結構名稱	說明
Constant (常數)	共用變數名稱設定
Support (支援)	共用方法
ExceptionCode (例外代碼)	共用例外代碼設定

CharacterSupport.php

```php
<?php
/**
 * 字元處理的Support
 *
 * @author 熊
 * @since 2019/09/15
 */
namespace App\Support;

class CharacterSupport
{
    /**
     * 由轉換規則，取得轉換後的值
     *
     * @param string $rule 使用類型 (star, component_exception)
     * @param string $value 欲轉換的值
     * @return string $result
     */
    public static function getCharacterByRule($rule, $value) {
        switch ($rule) {

            /**
             * 轉換規則: * => X
             */
            case 'star':
                $search = ['*'];
                $replace = ['X'];
                break;
```

CharacterSupport不依賴於任何元件。

架構設計準則

穩定依賴原則 (SDP)

朝著穩定的方向進行依賴。

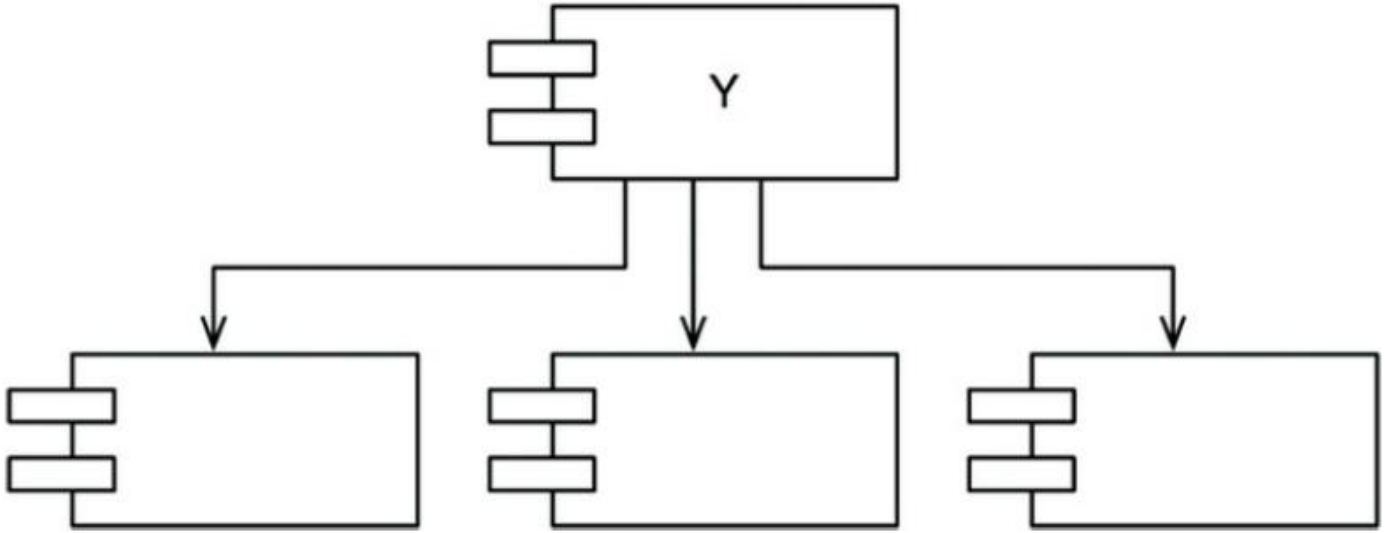


Figure 14.6 Y: a very unstable component

Y依賴3個物件，而Y不被任何元件依賴。

UML 箭頭說明

穩定依賴原則 (SDP)

朝著穩定的方向進行依賴。

Def.

- *Fan-in*：輸入依賴度。元件外部，依賴於該元件的類別個數。
- *Fan-out*：輸出依賴度。元件內部，依賴於元件外的類別個數。
- *Instability*：不穩定性。

$$I = \frac{Fan-out}{(Fan-in + Fan-out)} \quad , I \in [0,1]$$

穩定依賴原則 (SDP)

朝著穩定的方向進行依賴。

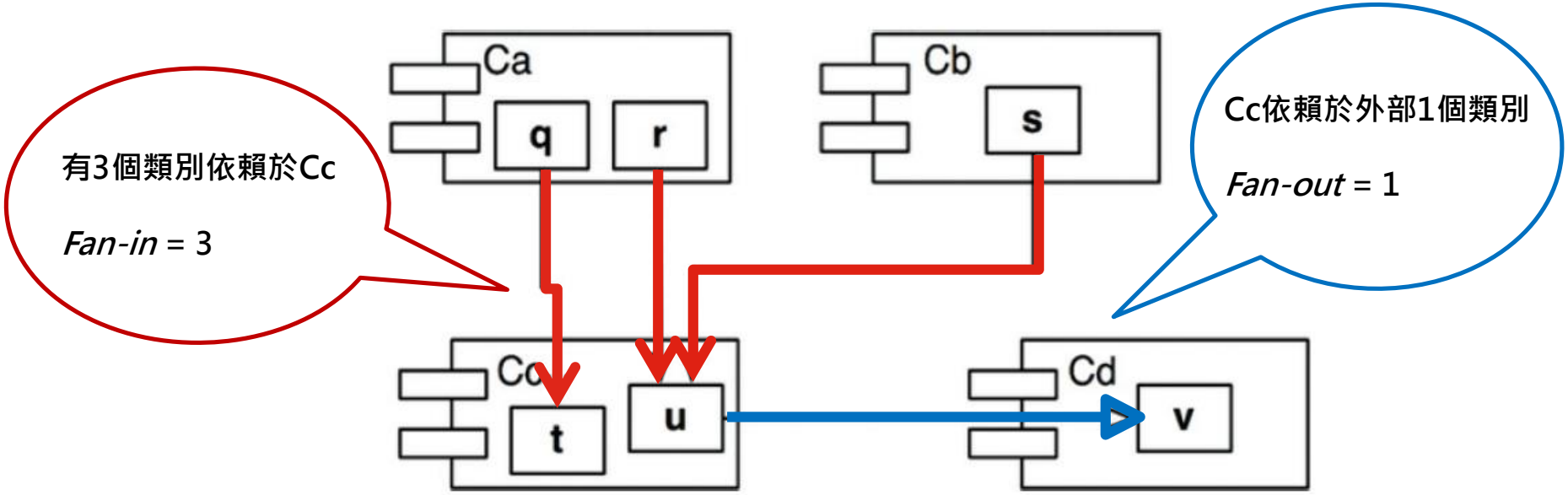


Figure 14.7 Our example

$$\therefore\ I = \frac{1}{3+1} = \frac{1}{4}$$

穩定依賴原則 (SDP)

朝著穩定的方向進行依賴。

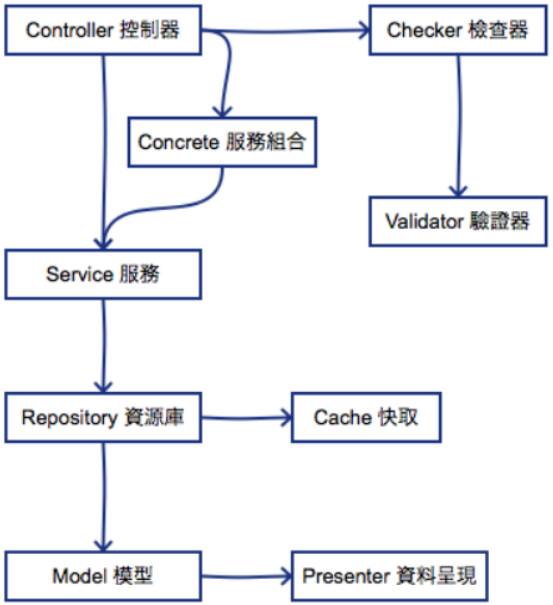
目標：

- **一個元件的 I 值應該大於它所依賴元件的 I 值。**
- **可改變的元件位於頂部並依賴於底部穩定的元件。**

穩定依賴原則 (SDP)

朝著穩定的方向進行依賴。

資料控制結構



穩定依賴原則 (SDP)

朝著穩定的方向進行依賴。

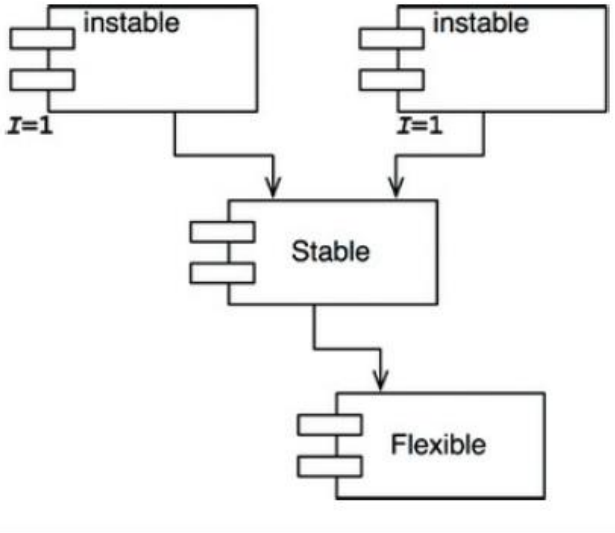


Figure 14.9 SDP violation

穩定依賴原則 (SDP)

朝著穩定的方向進行依賴。

我的 I 比較小
嗚嗚嗚

我的 I 比較大
嘻嘻嘻

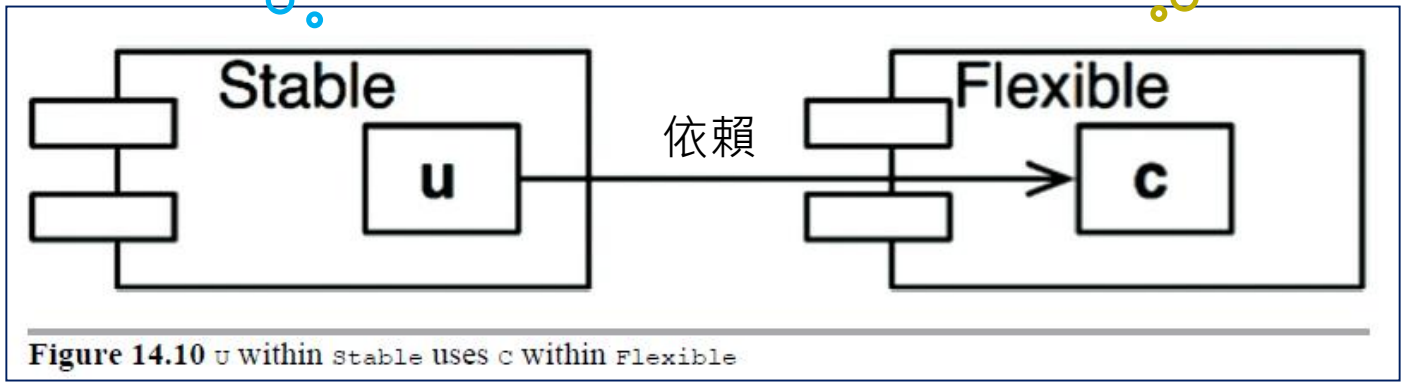


Figure 14.10 `U` within `Stable` uses `C` within `Flexible`

違反: 元件的 I 值應該大於它所依賴元件的 I 值。

穩定依賴原則 (SDP)

朝著穩定的方向進行依賴。

Solution.

- 利用**依賴反向原則 (DIP)**

穩定依賴原則 (SDP)

朝著穩定的方向進行依賴。

Fan-out = 0
Fan-in = 2

$\therefore I = 0$

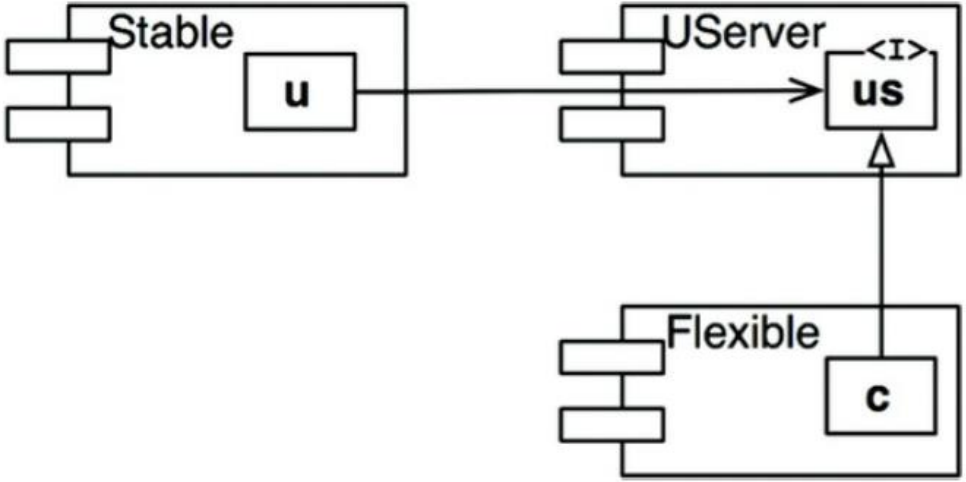


Figure 14.11 c implements the interface class us

現在所有**依賴方向**都順著 **/減少**的方向。

穩定依賴原則 (SDP)

朝著穩定的方向進行依賴。

Note.

- 當高階/低階模組角色錯置問題時，考慮**依賴反向原則 (DIP)**。
- 無環依賴原則 (ADP) 及 穩定依賴原則 (SDP) 是**依賴反向原則 (DIP)** 的延伸。

穩定抽象原則 (SAP)

(Stable Abstractions Principle)

穩定抽象原則 (SAP)

元件的抽象程度應該與元件的穩定程度一致。

- 一個穩定的元件應該是抽象的。
- 一個不穩定的元件應該是具體的。
- 依賴應該朝著抽象的方向。

穩定抽象原則 (SAP)

元件的抽象程度應該與元件的穩定程度一致。

計算抽象性的度量

Def.

N_c—元件中類別的總數。

N_a—元件中抽象類別及介面的總數。

A—抽象性。$A = N_a \div N_c$ ，$A \in [0,1]$

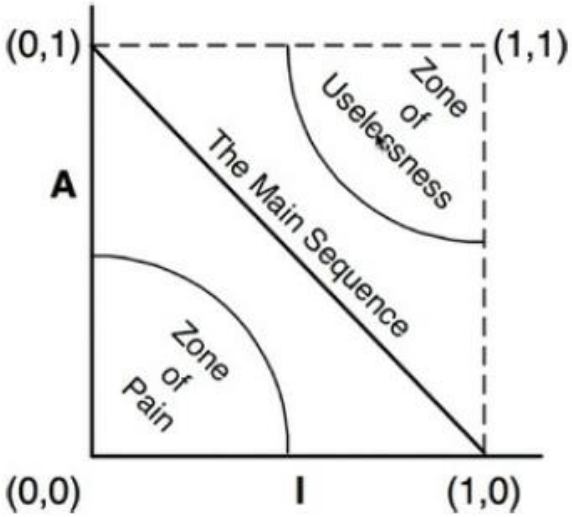
穩定抽象原則 (SAP)

元件的抽象程度應該與元件的穩定程度一致。

抽象且穩定

抽象卻不穩定

$$A = \frac{N_a}{N_c}$$



具體卻穩定

具體且不穩定

Figure 14.13 Zones of exclusion

$$I = \frac{Fan-out}{(Fan-in + Fan-out)}$$

穩定抽象原則 (SAP)

元件的抽象程度應該與元件的穩定程度一致。

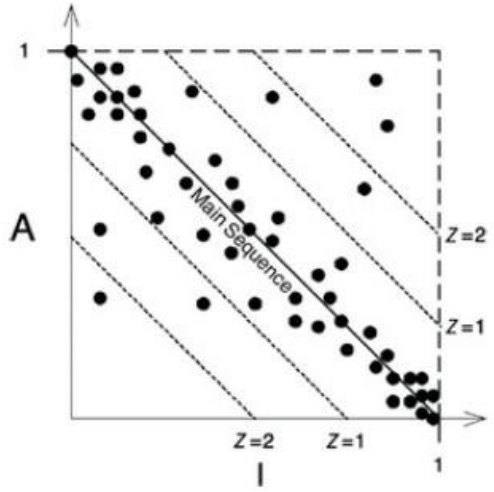
到主序列的距離 (點到線距離公式簡化)

Def.

$$D = |A + I - 1| \quad , D \in [0,1]$$

穩定抽象原則 (SAP)

元件的抽象程度應該與元件的穩定程度一致。



註：Z為標準化後的標準差

Figure 14.14 Scatterplot of the components

穩定抽象原則 (SAP)

元件的抽象程度應該與元件的穩定程度一致。

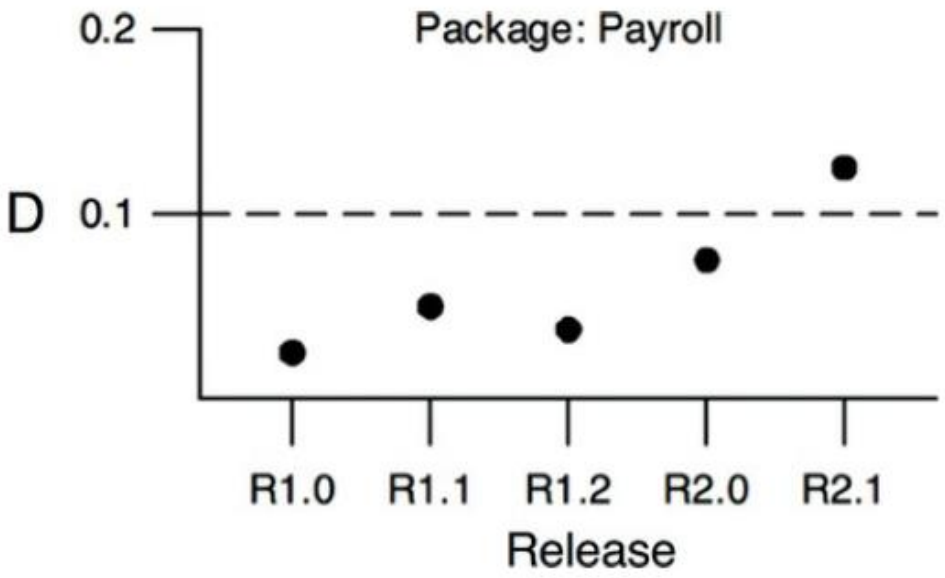


Figure 14.15 Plot of D for a single component over time

穩定抽象原則 (SAP)

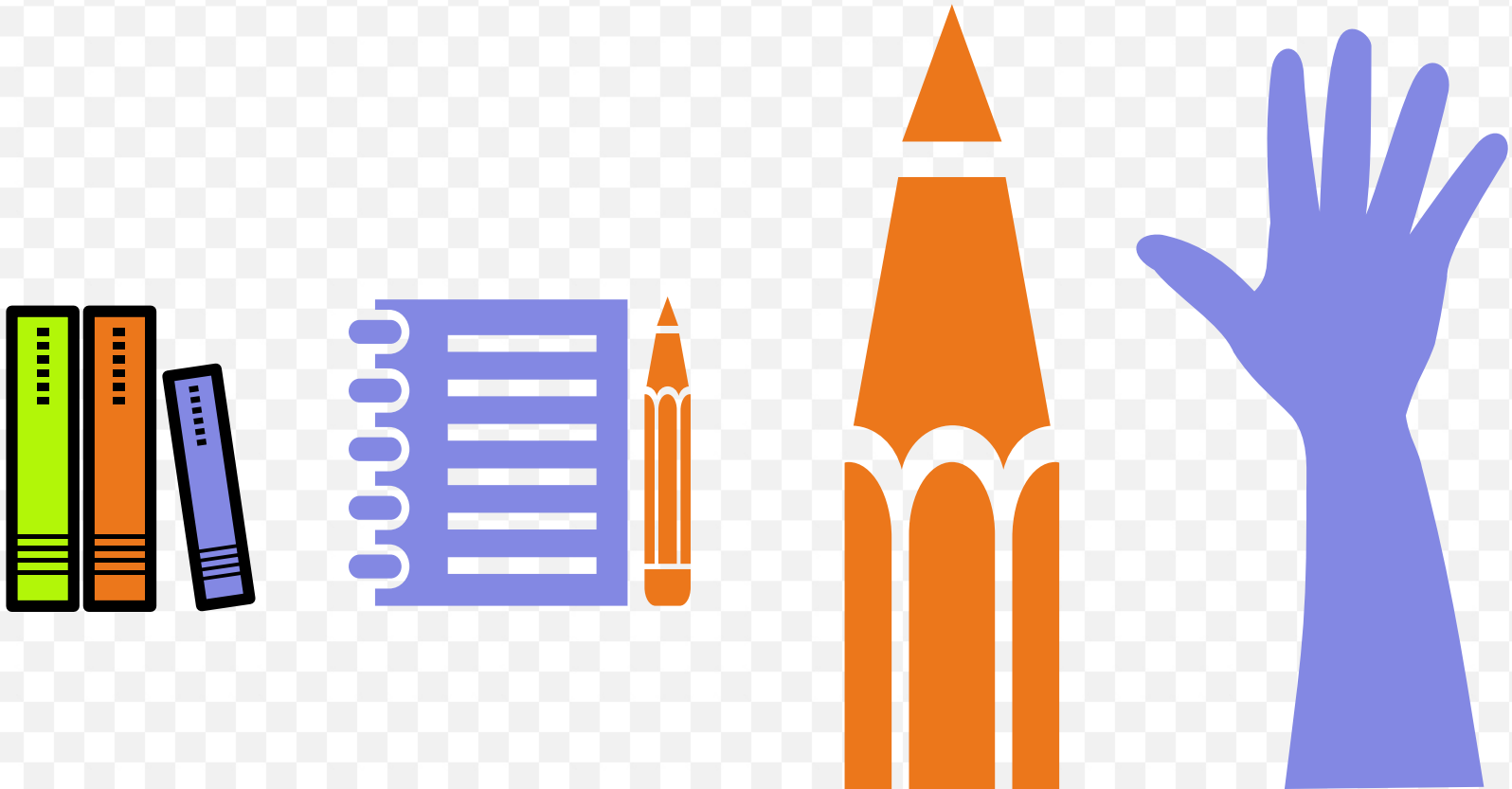
元件的抽象程度應該與元件的穩定程度一致。

- 本章描述的依賴性管理度量，僅為作者的量測方式。
- 目的為找到「好的依賴及抽象」方式。

Thank you

感謝你的聆聽

補充資料



單一職責原則 (SRP)

(Single Responsibility Principle)

- 一個模組應該只有一個，且只有一個理由會使其改變。
- 一個模組應該只對唯一一個角色負責。

開放封閉原則 (OCP)

(Open-Closed Principle)

- 一個軟體製品應該對於擴展是開放的，但對於修改是關閉的。

Liskov替換原則 (LSP)

(Liskov Substitution Principle)

- 若對型態S的每一個物件o1，都存在一個型態為T的物件o2，使得在所有針對T編寫的程式P中，用o1替換o2後，程式P的行為功能不變，則S是T的子型態。

介面隔離原則 (ISP)

(Interface Segregation Principle)

- 客戶不應該被強迫依賴他們不使用的方法。

依賴反向原則 (DIP)

(Dependency Inversion Principle)

- 原始碼的依賴關係只涉及抽象，不涉及具體。
- 高階模組不應該依賴於低階模組，兩者都該依賴抽象介面。
- 抽象界面不應該依賴具體實現，而具體實現則應該依賴抽象界面。

發佈等價原則 (REP)

(Reuse/Release Equivalence Principle)

- 再使用性的細微度就是發佈的細微度。

共同封閉原則 (CCP)

(Common Closure Principle)

- 將那些會因著相同理由、在相同時間發生變化的類別收集到相同的元件之中。將那些在不同時間、因著不同理由發生變化的類別分割到不同元件之中。
- 即單一職責原則 (SRP) 的元件版本。

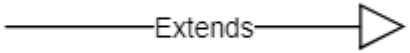
共同重複使用原則 (CRP)

(Common Reuse Principle)

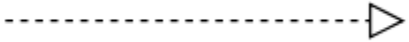
- 不要強迫元件的使用者依賴他們不需要的東西。
- 即介面隔離原則 (ISP) 的元件版本。

UML 箭頭說明

1. **空心三角箭頭的實線，箭頭指向父類** = extends

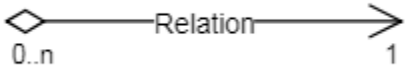


2. **空心三角箭頭的虛線，箭頭指向接口** = implements



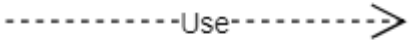
3. **普通箭頭的實線，指向被擁有者** = 關聯關係 (Association)

4. **空心菱形的實線，菱形指向整體，箭頭指向部分** = 聚合關係 (Aggregation)



5. **實心菱形的實線，菱形指向整體，箭頭指向部分** = 組合關係 (Composition)
(與聚合關係差異在: **<u>組合關係的部分不能離開整體</u>**)

6. **普通箭頭的虛線，箭頭指向被使用者** = 依賴 (Dependency)



架構設計準則

結構名稱	可存取	可被存取
Controller (控制器)	Checker、Concrete、Service、DB transaction	-
Concrete (服務組合)	Service	Controller
Service (服務)	Repository、Validator	Controller、Concrete
Checker (檢查器)	Validator	Controller
Validator (驗證器)	-	Checker
Repository (資源庫)	自己的 Model、自己的 Cache	Service
Cache (服務)	-	Repository
Model (模型)	自己的 Presenter	Repository
Presenter (資料呈現)	-	Model

- 參考自架構設計準則 · Laravel 5學習筆記。