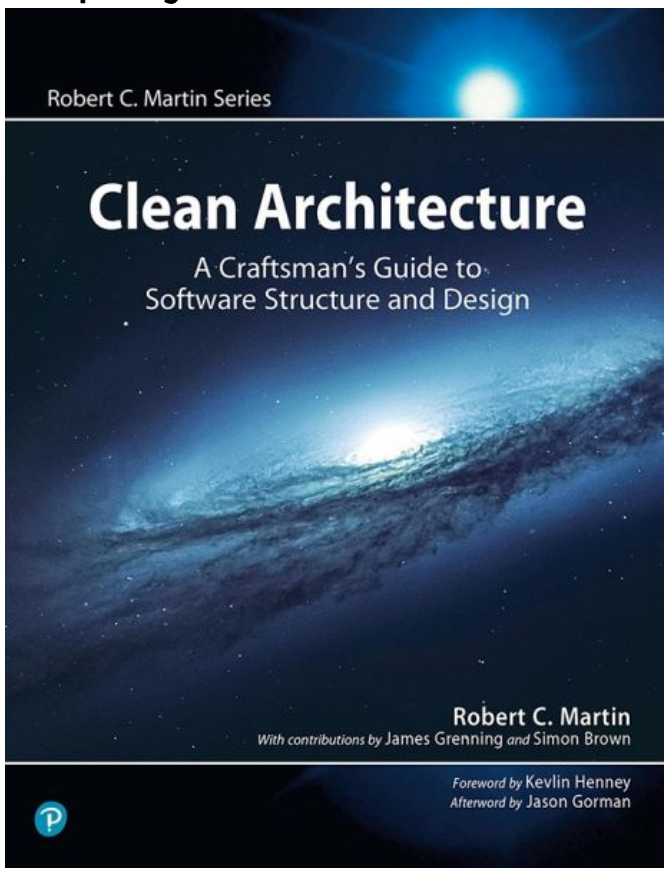
Introduction à l'architecture logicielle

de Astramast

en Mars 2025

0 - Disclaimer

Tout ce que je vais dire et montrer vient de :



Clean Architecture: A Craftsman's Guide to Software Structure and Design: A Craftsman's Guide to Software Structure and Design (Robert C. Martin Series)

1 - L'architecture, c'est quoi, à quoi ça sert ?

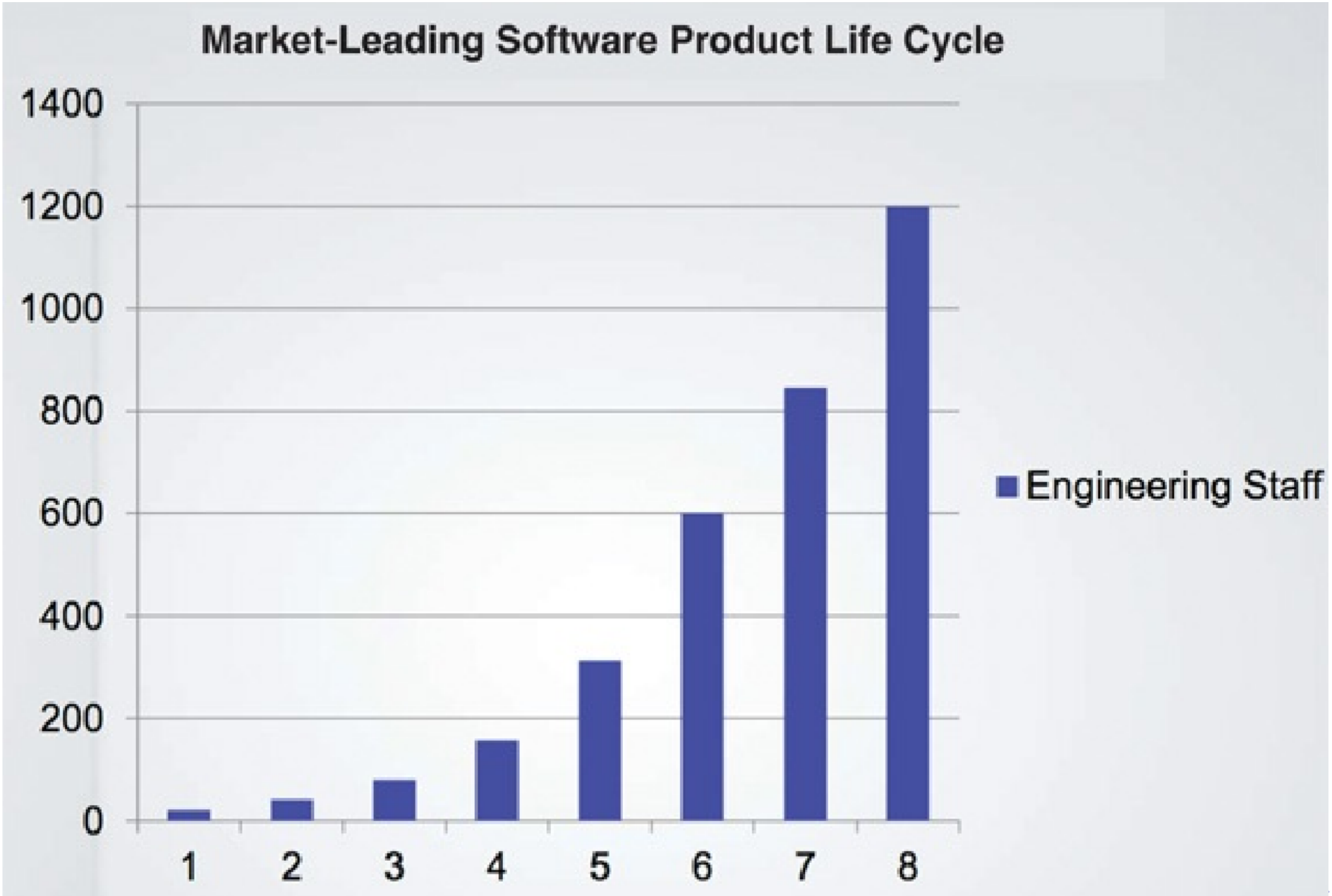
- Écrire un code qui marche c'est FACILE
- Écrire un bon code qui marche c'est DIFFICILE
- C'est quoi un "bon" code ?

1 - L'architecture, c'est quoi, à quoi ça sert ?

- Écrire un code qui marche c'est FACILE
- Écrire un bon code qui marche c'est DIFFICILE
- C'est quoi un "bon" code ?

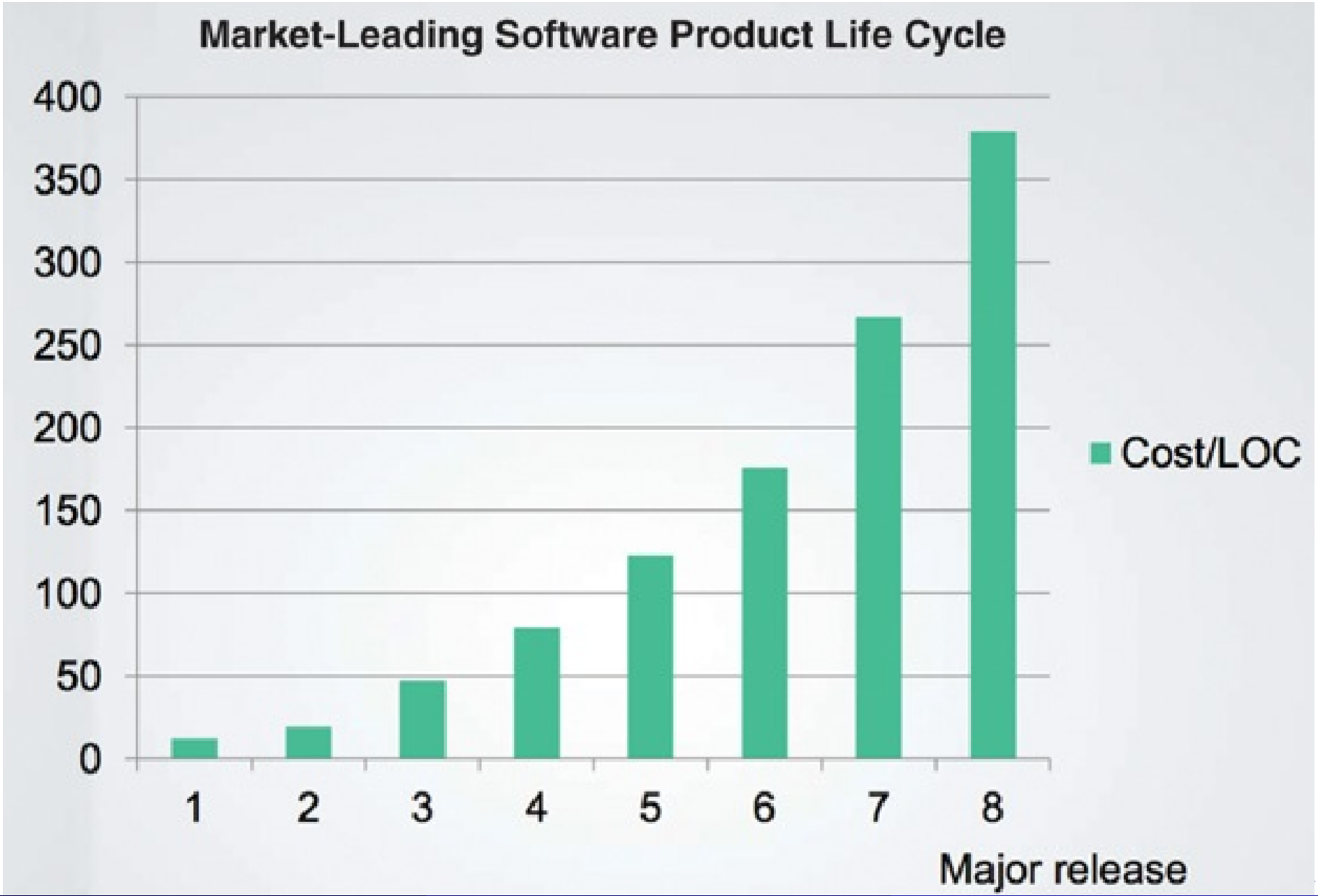
  Un bon code est un code clair, facile à modifier, facile à maintenir.

1 - L'architecture, c'est quoi, à quoi ça sert ?

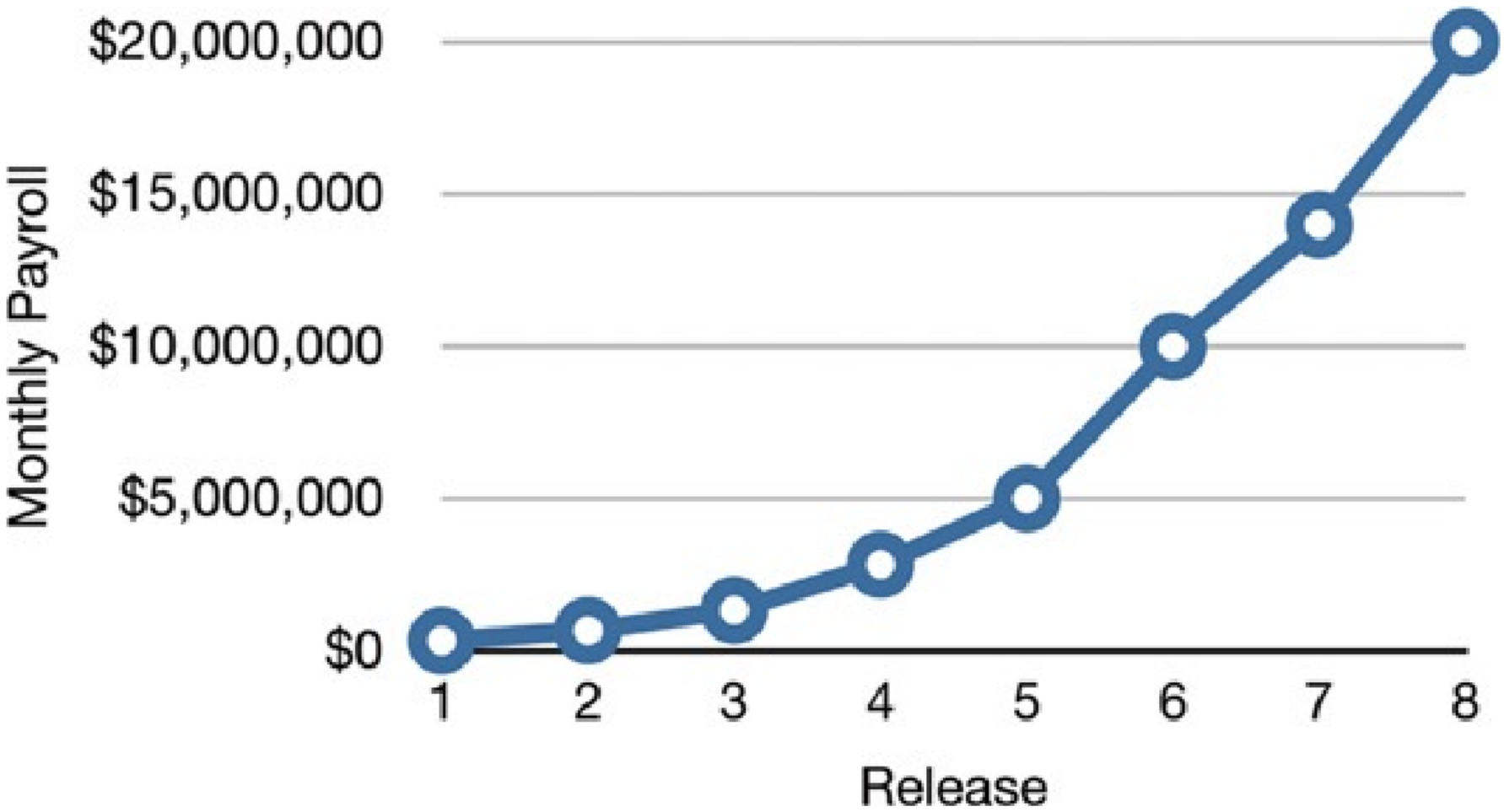


1 - L'architecture, c'est quoi, à quoi ça sert ?

1 - L'architecture, c'est quoi, à quoi ça sert ?



1 - L'architecture, c'est quoi, à quoi ça sert ?



1 - L'architecture, c'est quoi, à quoi ça sert ?

Le but de l'architecture logicielle est de minimiser les efforts humains nécessaires pour construire et maintenir un système.

1 - L'architecture, c'est quoi, à quoi ça sert ?

Une bonne architecture doit fournir des facilités dans :

- Développement
- Déploiement
- Opérations
- Maintenance
- L'extensibilité
- Indépendance au matériel

2 - Design vs Architecture

L'architecture sans design, c'est construire une maison sans savoir ce qu'est une brique.

3 - Les principes SOLID

- **S**ingle Responsibility Principle (SRP)
- **O**pen-Closed Principle (OCP)
- **L**iskov Substitution Principle (LSP)
- **I**nterface Segregation Principle (ISP)
- **D**ependency Inverstion Principle (DIP)

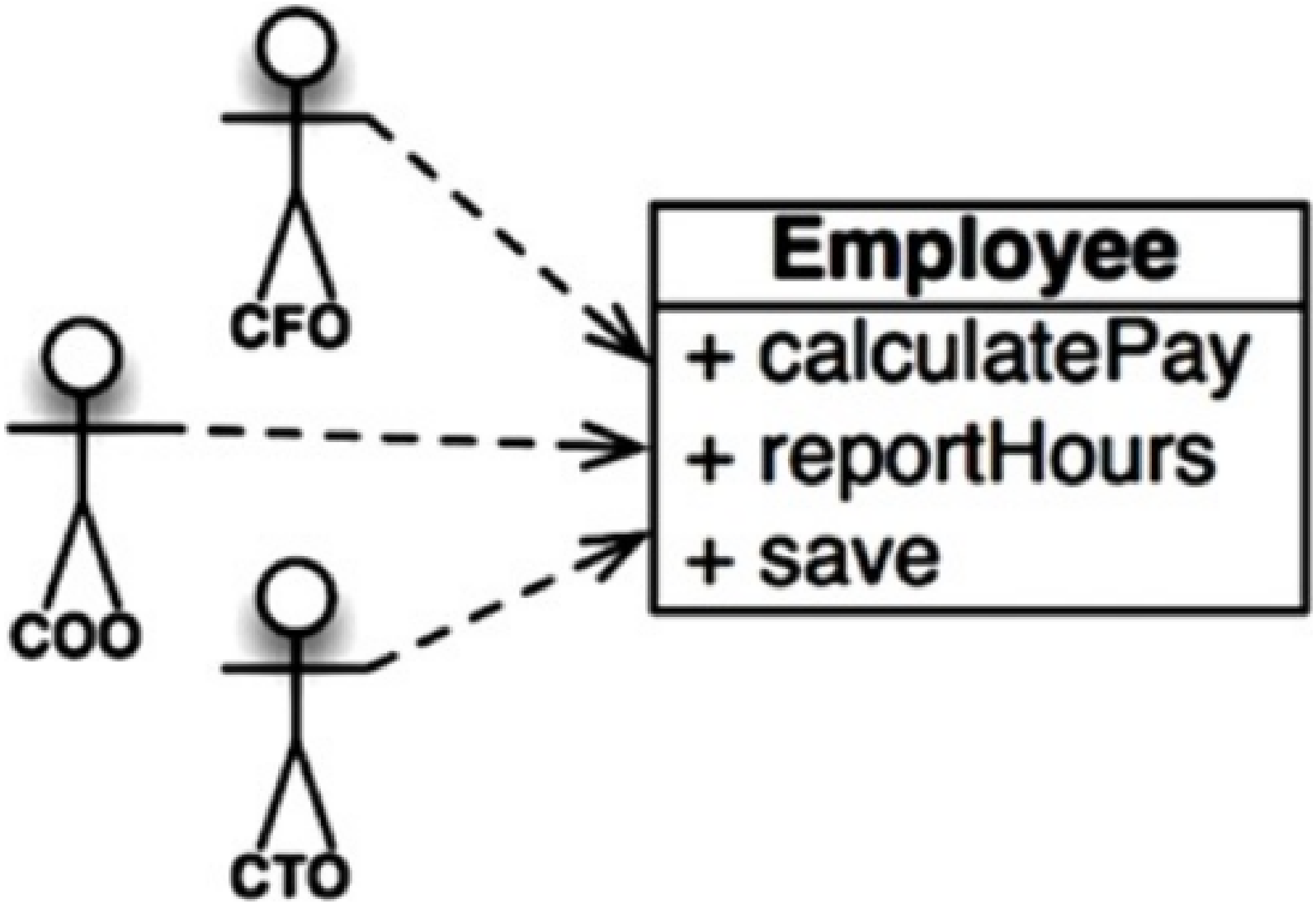
Thus the final version of the SRP is:

A module should be responsible to one, and only one, actor.

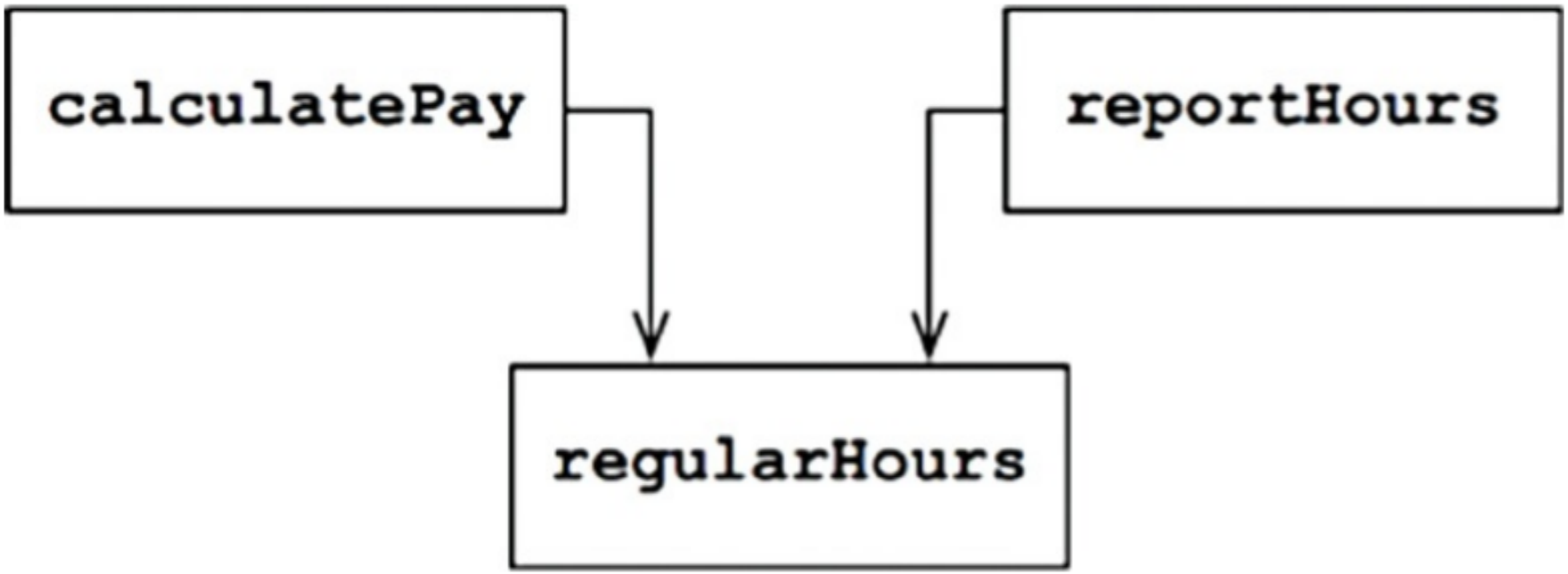
3 - SRP

Un Module est un ensemble de fonctions ou de classes, rendus cohérent par le respect du SRP. (En général, il s'agit d'un fichier de code source.)
Un acteur est une personne ou un groupe de personnes qui demandent des changements du système de la même manière.

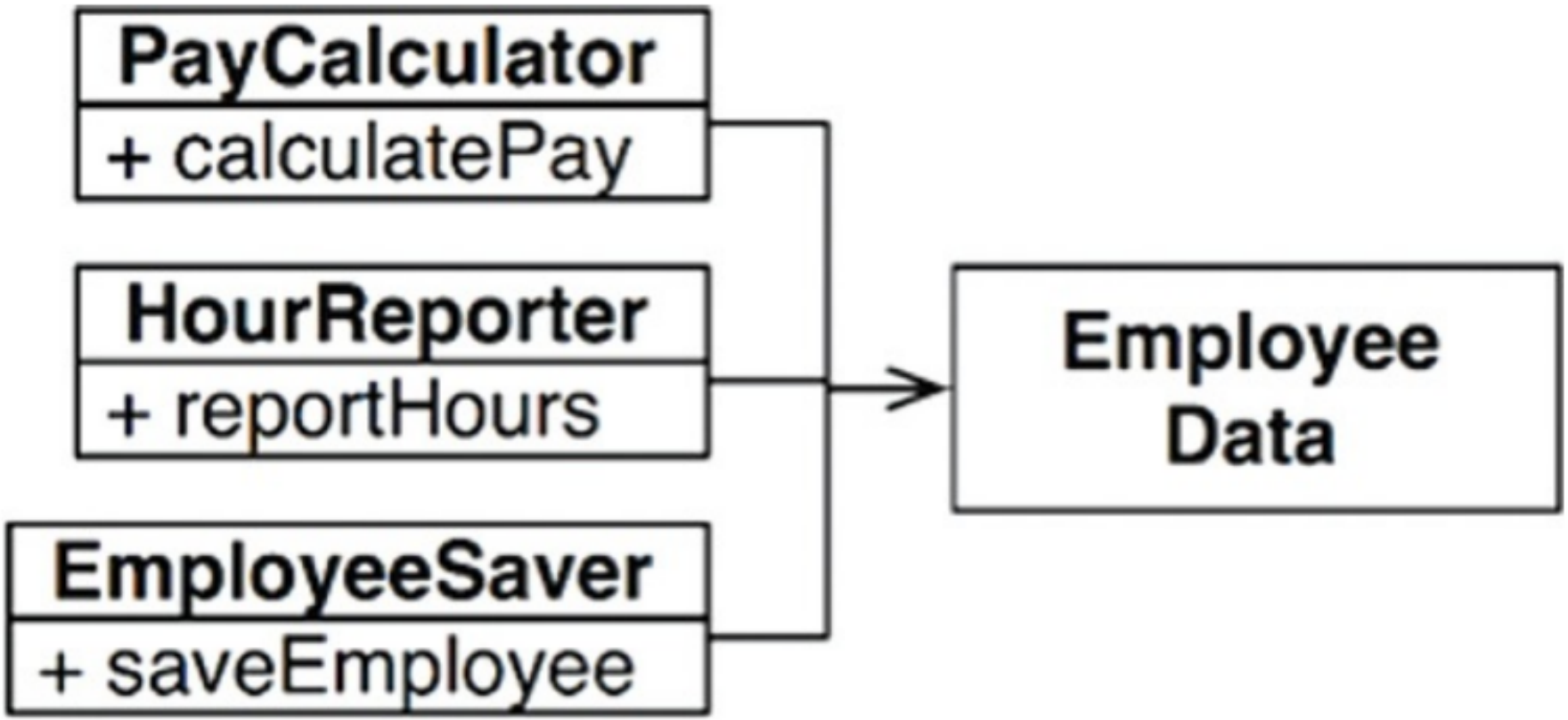
3 - SRP



3 - SRP



3 - SRP


PayCalculator
+ calculatePay
HourReporter
+ reportHours
EmployeeSaver
+ saveEmployee
Employee Data

3 - SRP

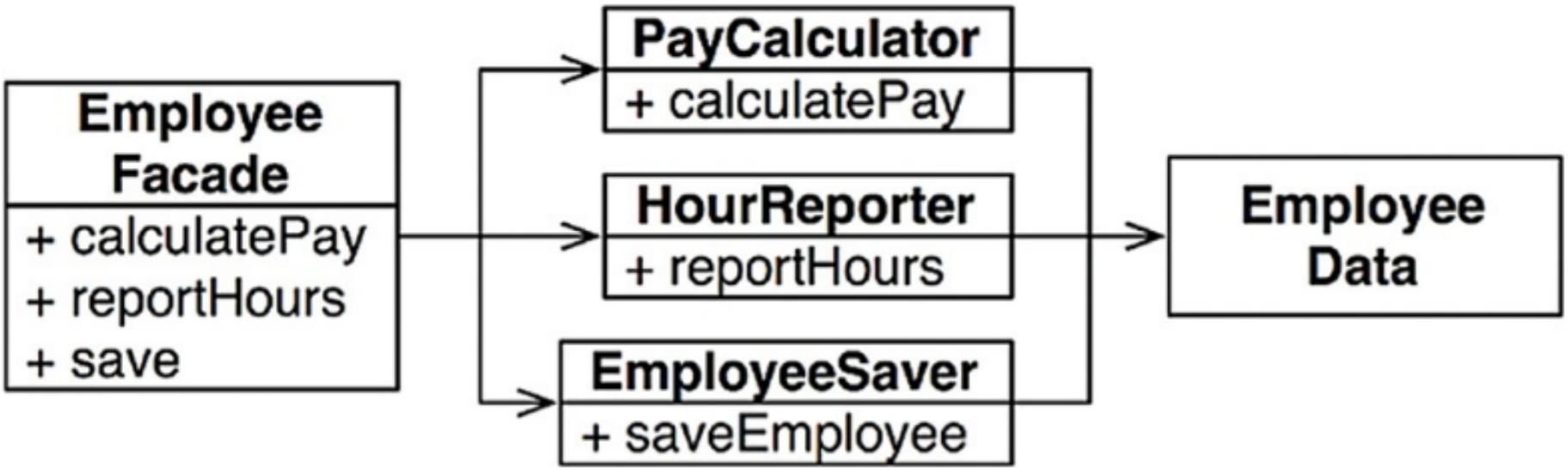


Figure 7.4 The Facade pattern

3 - SRP

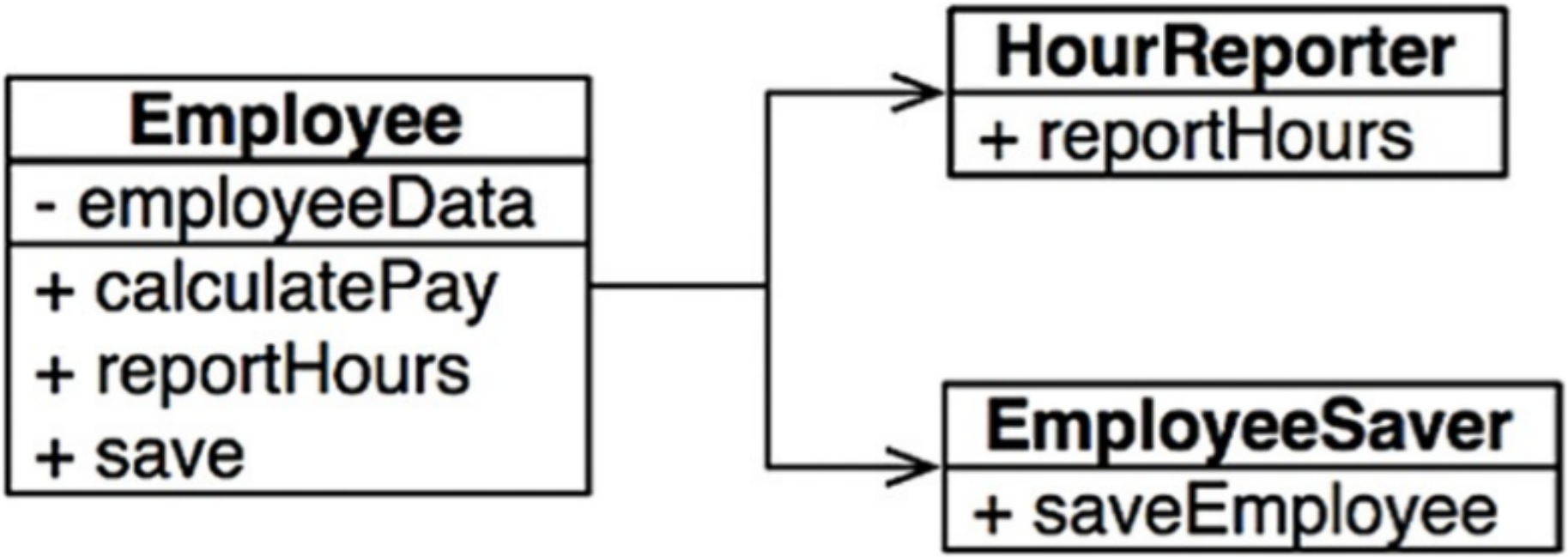
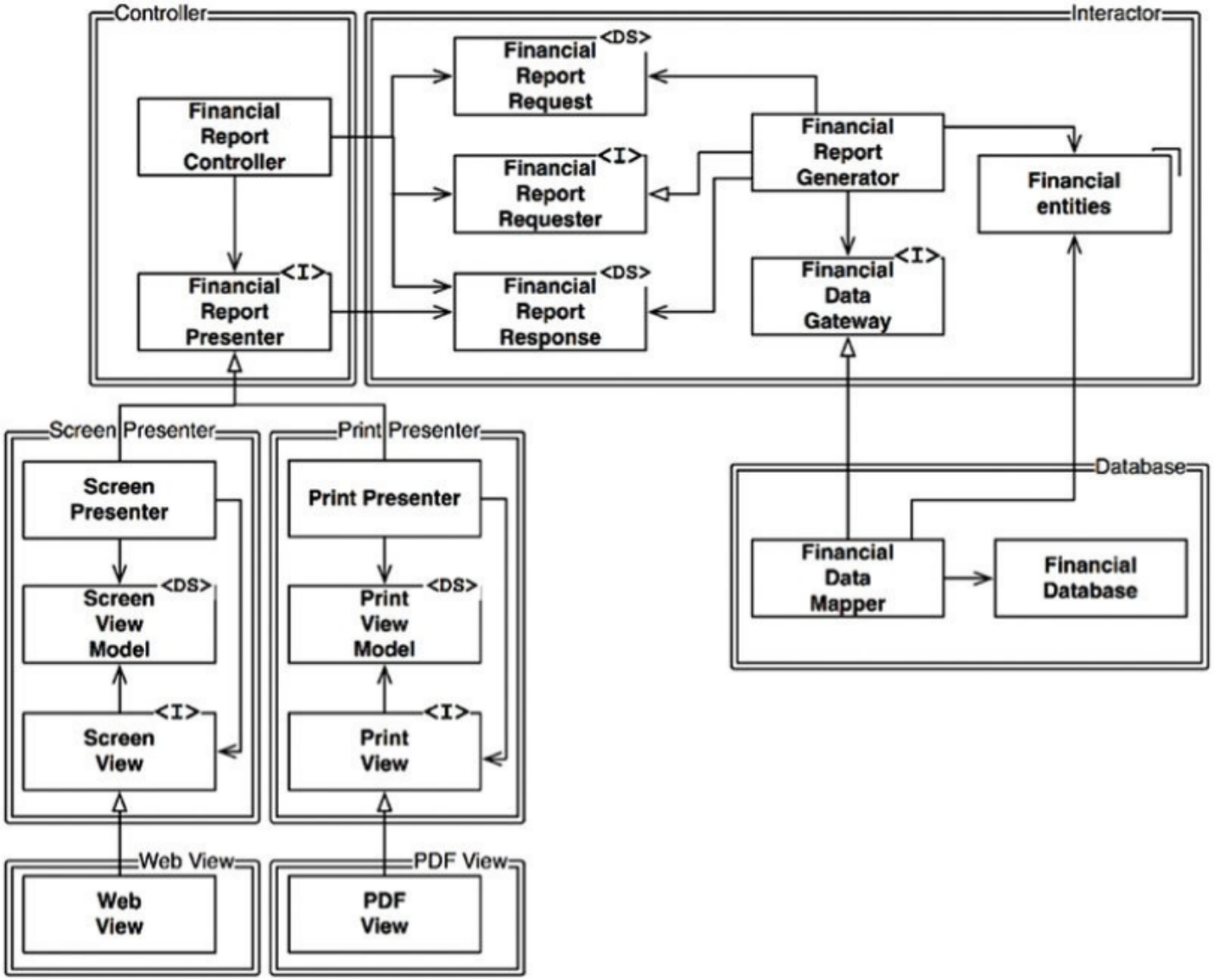


Figure 7.5 The most important method is kept in the original `Employee` class and used as a *Facade* for the lesser functions

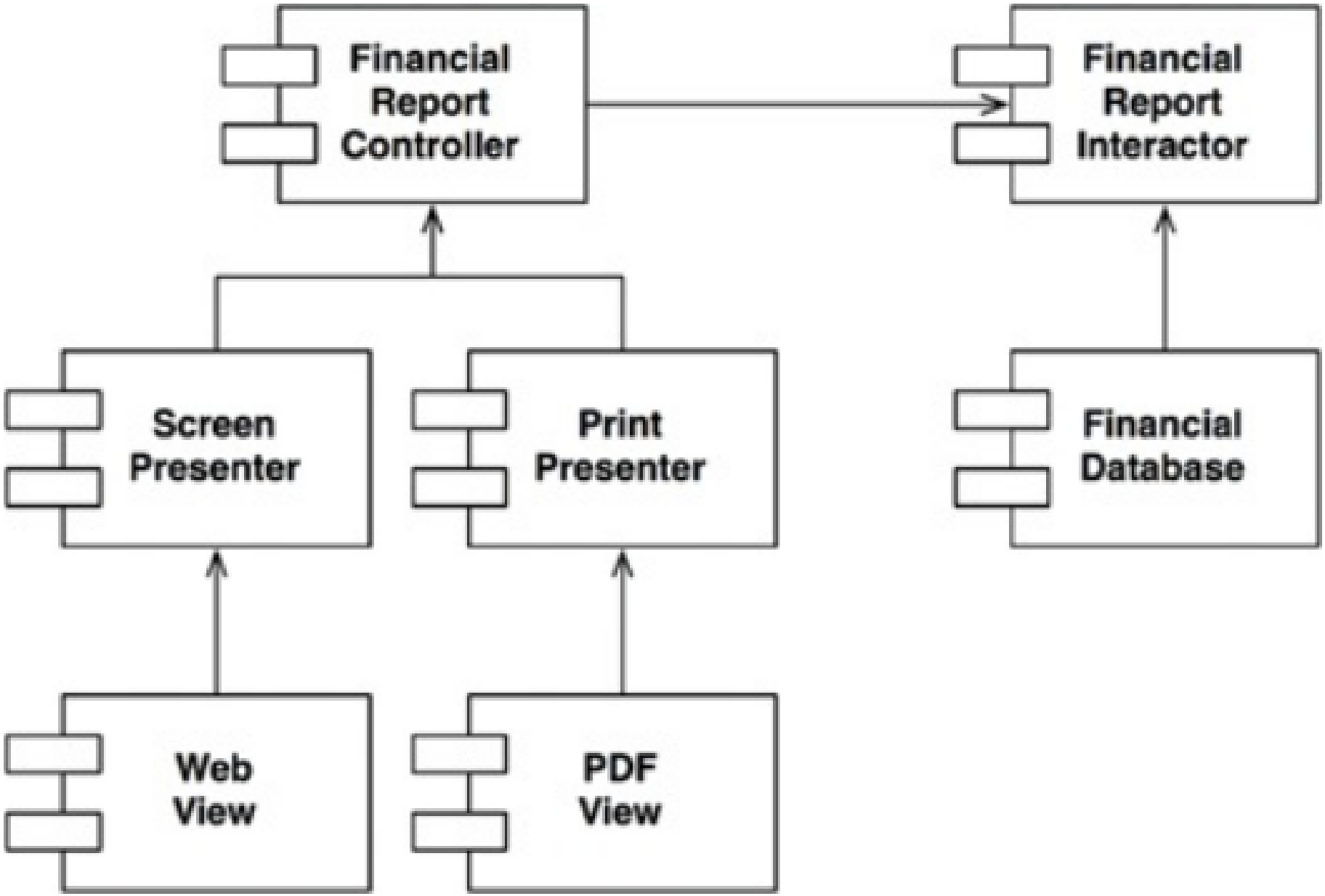
3 - OCP

Open-Closed Principle : "A software artifact should be open for extension but closed for modification."

3 - OCP


Controller
Financial Report Controller
Financial Report Presenter <I>
Interactor
Financial Report Request <DS>
Financial Report Requester <I>
Financial Report Response <DS>
Financial Report Generator
Financial Data Gateway <I>
Financial entities
Screen Presenter
Screen Presenter
Screen View Model <DS>
Screen View <I>
Print Presenter
Print Presenter
Print View Model <DS>
Print View <I>
Database
Financial Data Mapper
Financial Database
Web View
Web View
PDF View
PDF View

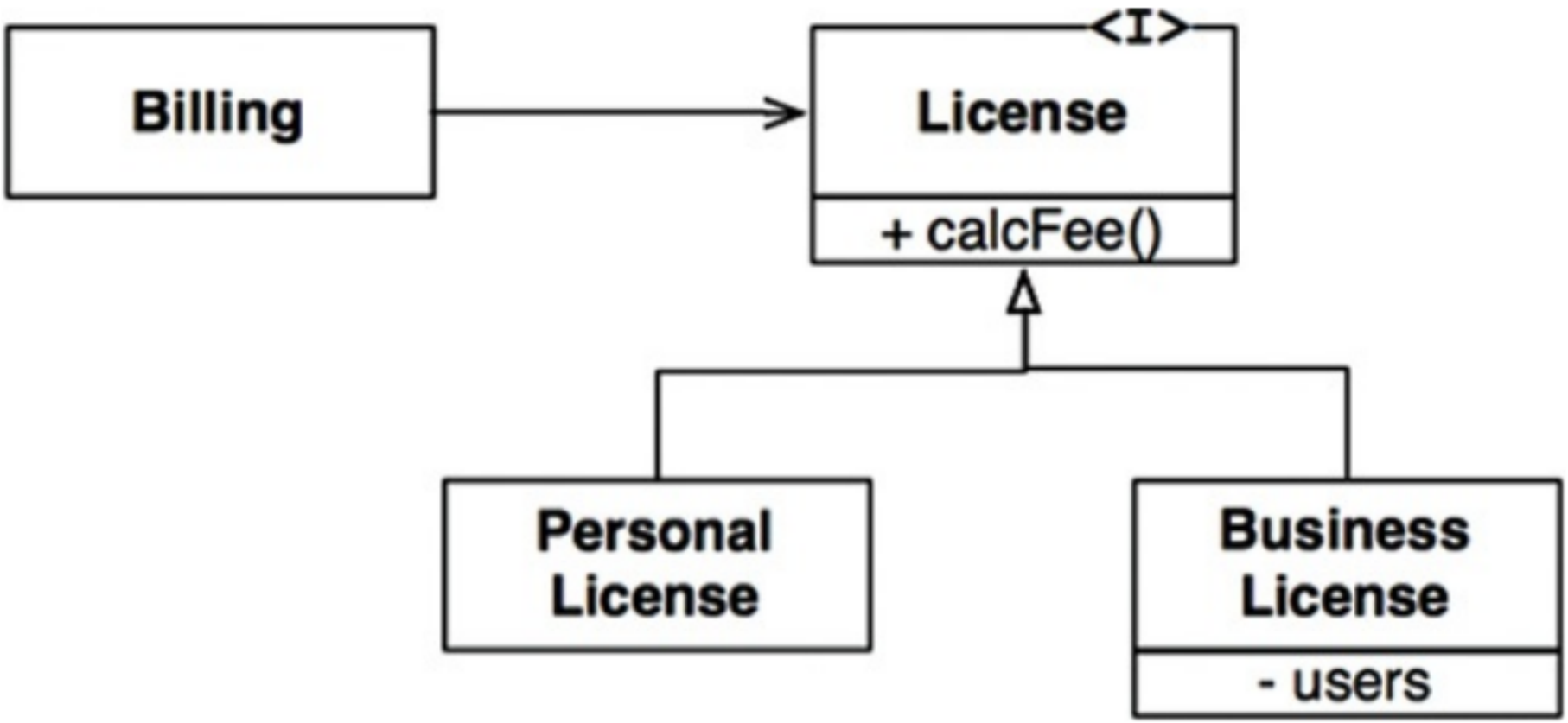
3 - OCP


Financial
Report
Controller
Financial
Report
Interactor
Screen
Presenter
Print
Presenter
Financial
Database
Web
View
PDF
View

3 - LSP

[Barbara] Liskov ['s] Substitution Principle : "If for each object o1 of type S there is an object o2 of type T such that for all programs P defined in terms of T, the behavior of P is unchanged when o1 is substituted for o2 then S is a subtype of T"

3 - LSP


Billing
<I>
License
+ calcFee()
Personal License
Business License
- users

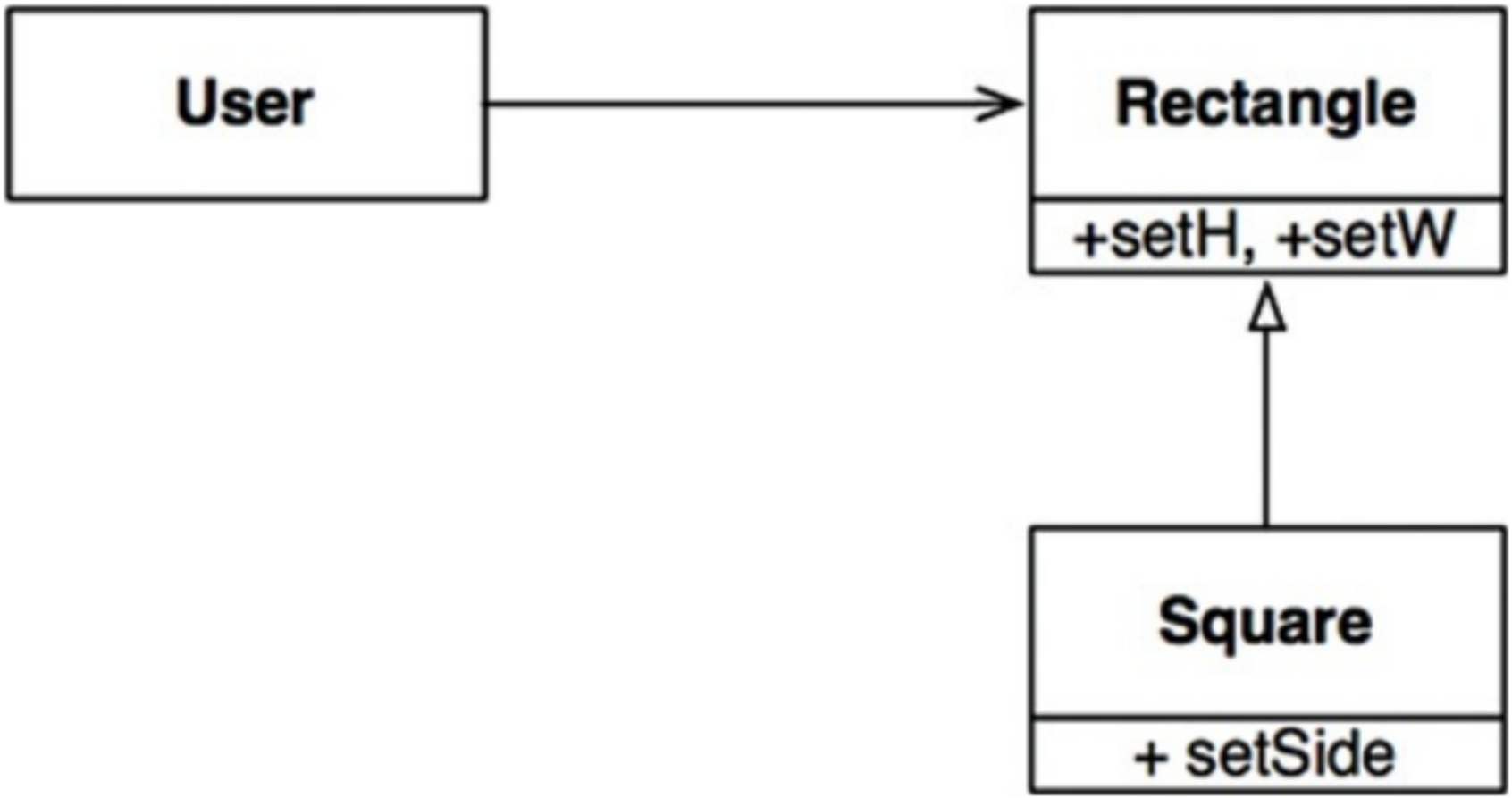
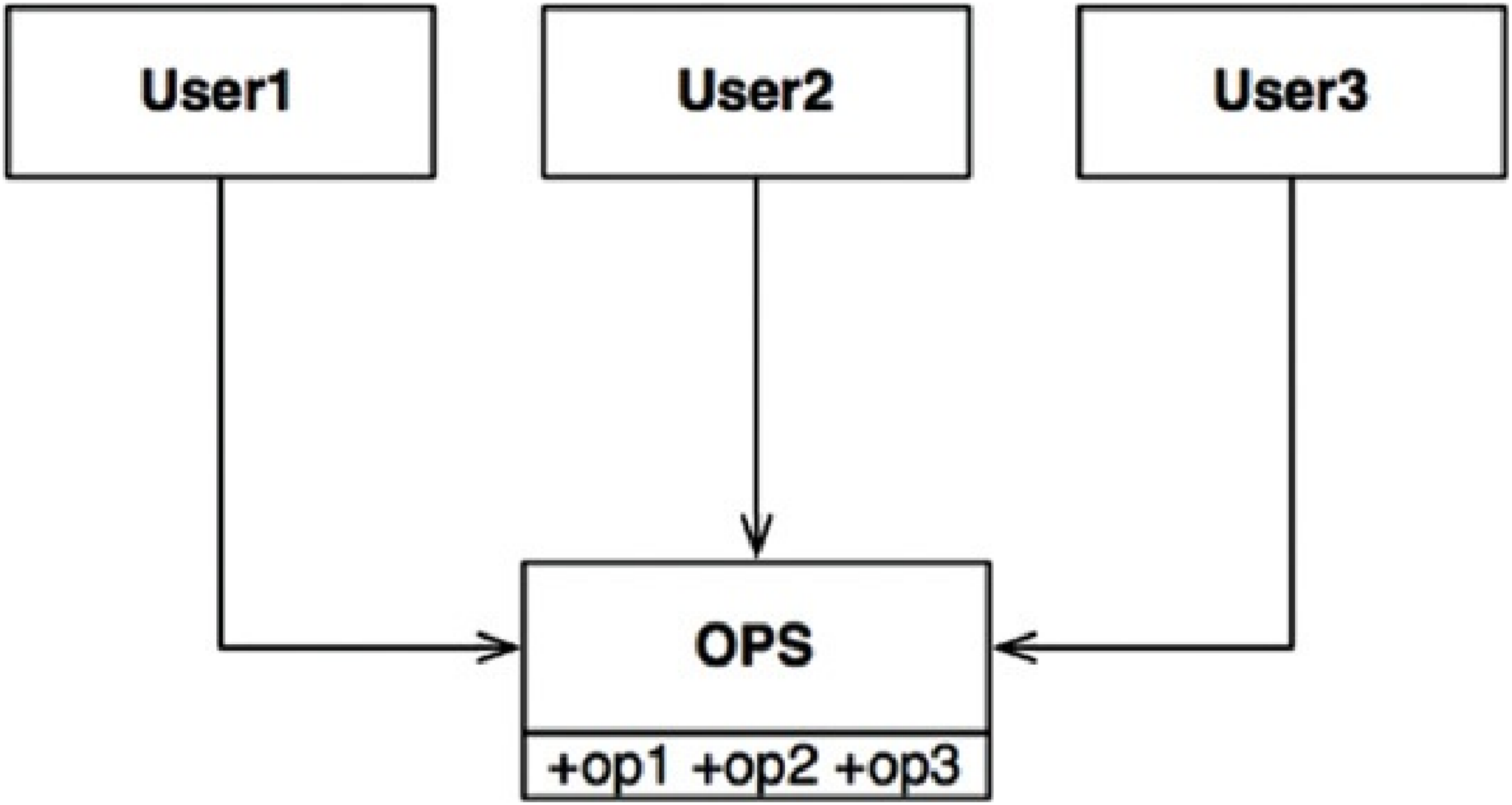


Figure 9.2 The infamous square/rectangle problem

3 - ISP



3 - ISP

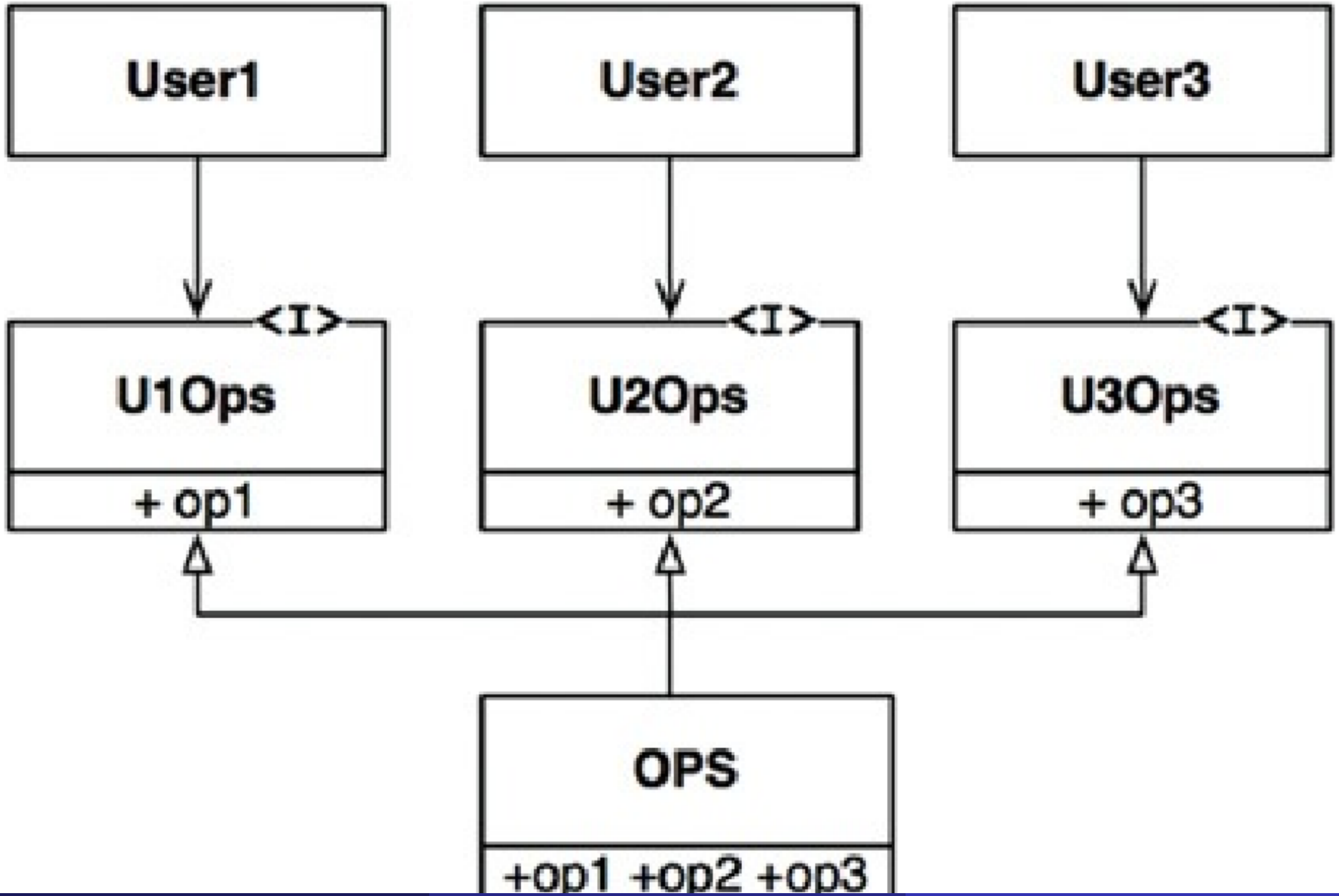

User1
User2
User3
<I>
U1Ops
+ op1
<I>
U2Ops
+ op2
<I>
U3Ops
+ op3
OPS
+op1 +op2 +op3

Figure 10.3 A problematic architecture

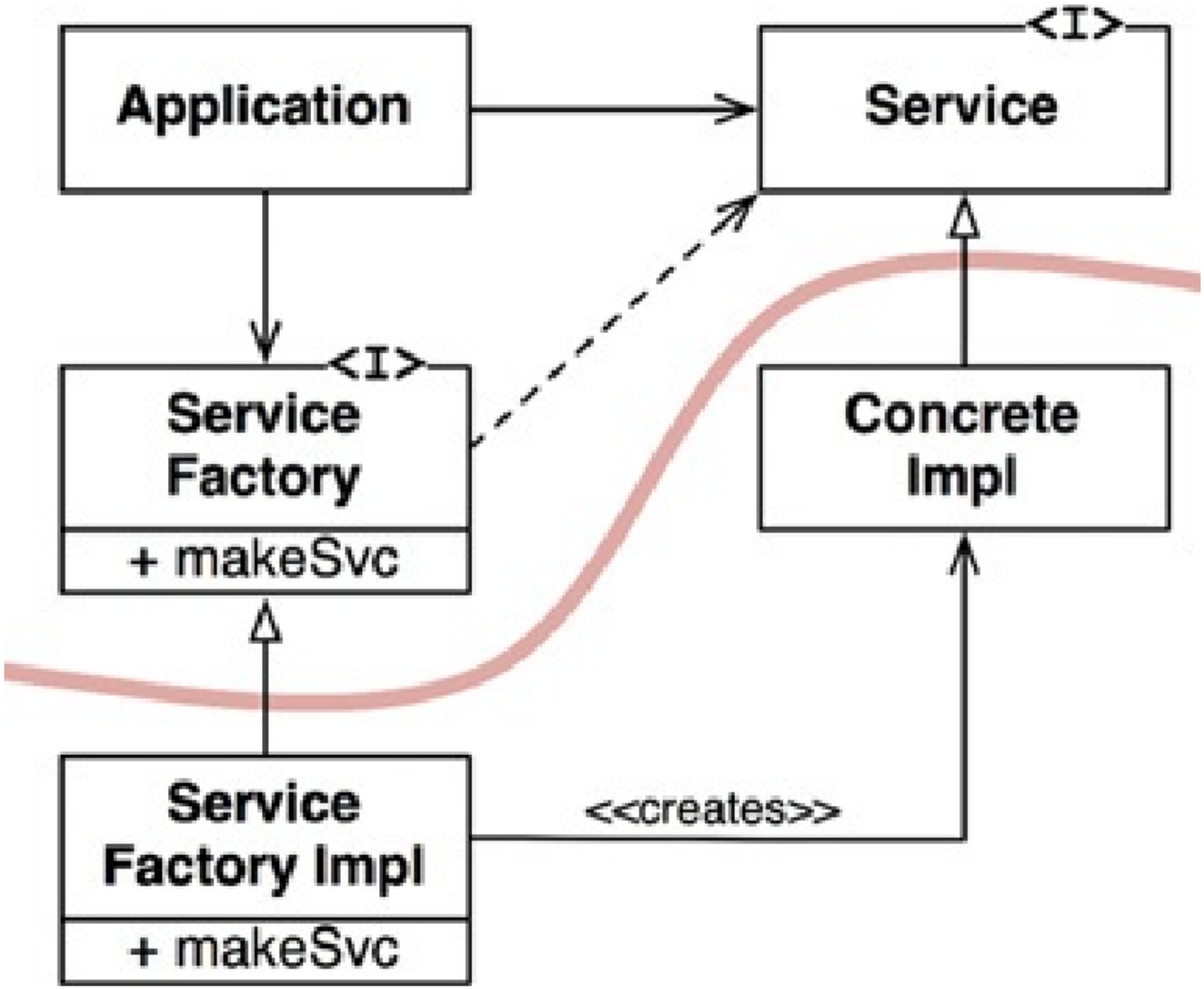
3 - DIP

"The Dependency Inversion Principle tells us that the most flexible systems are those in which source code dependencies refer only to abstractions, not to concretions."

3 - DIP

- Ne vous référez pas à des classes volatiles concrètes.
- Ne dérivez pas de classes volatiles concrètes.
- Ne surchargez pas de fonctions concrètes.

3 - DIP


Application
<I>
Service
<I>
Service
Factory
+ makeSvc
Concrete
Impl
Service
Factory Impl
+ makeSvc
<<creates>>

3 - Design et Architecture

- Les principes SOLID nous disent comment agencer nos briques en pièces cohérentes.
- Les principes de composants nous disent comment agencer nos pièces en bâtiments cohérents.

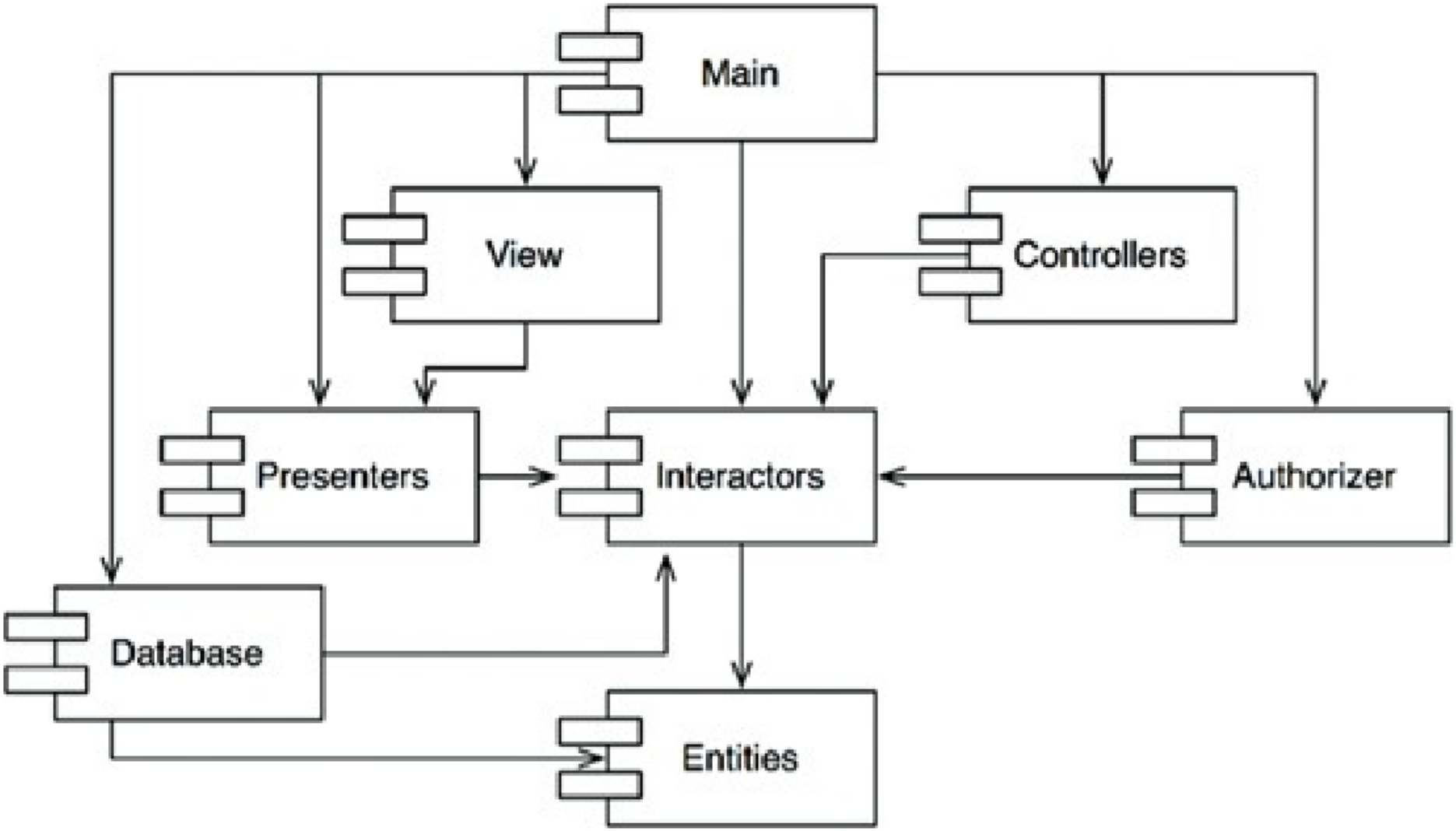
4 - La Stabilité

- Acyclic Dependencies Principle (ADP)
- Stable Dependencies Principle (SDP)
- Stable Abstractions Principle (SAP)

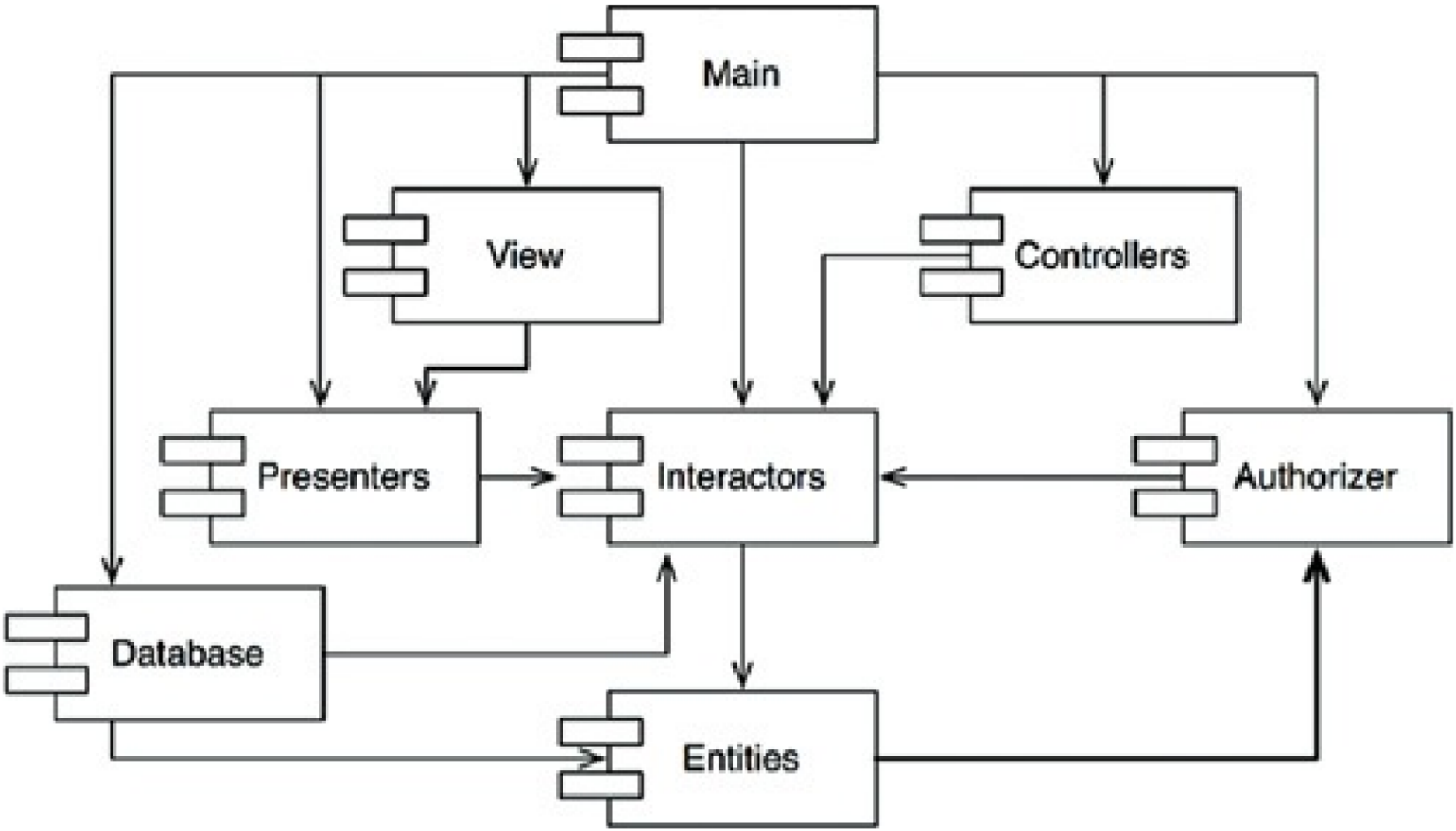
4 - ADP

"Allow no cycle in the component dependency graph."

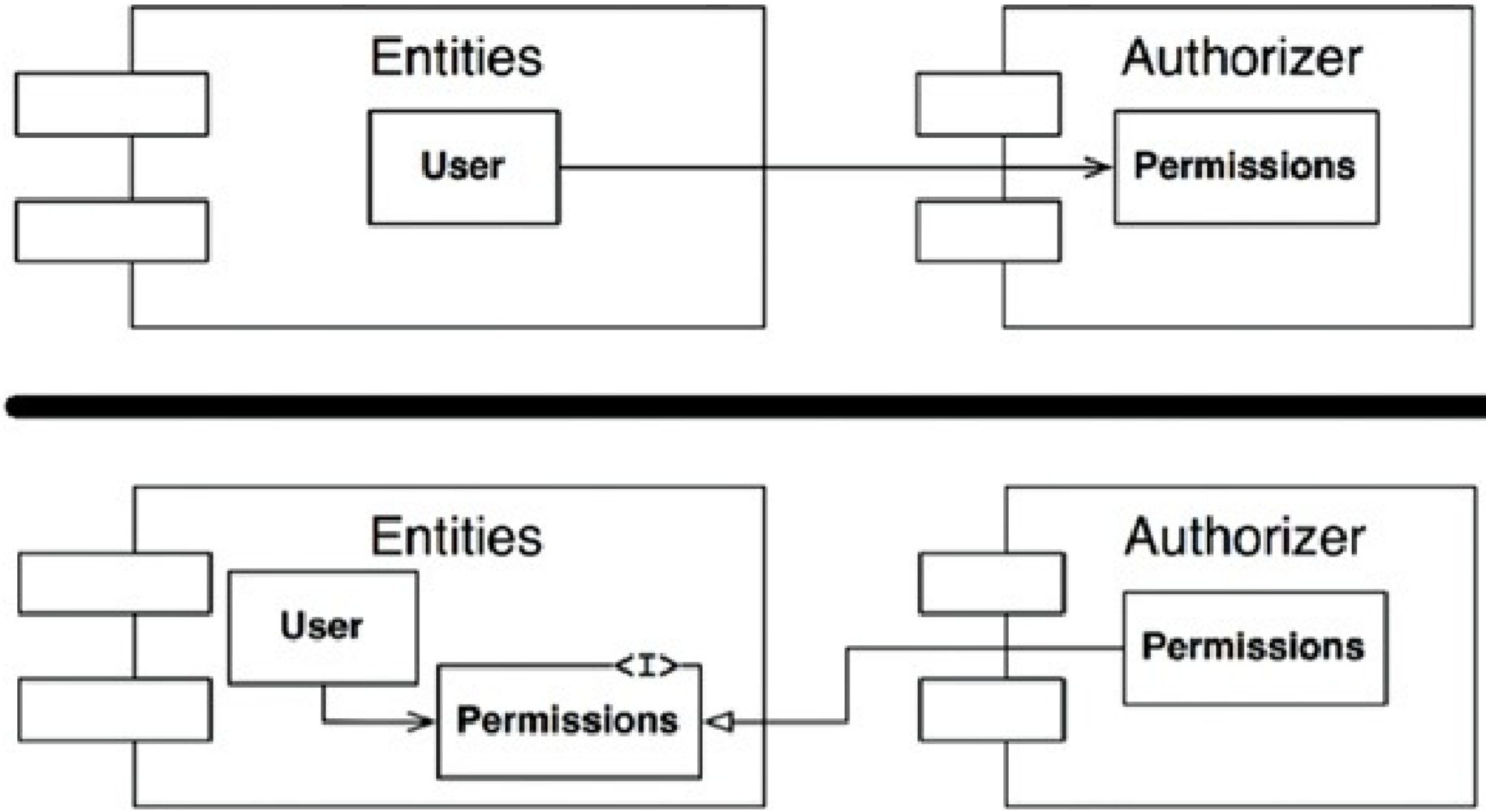
4 - ADP


Main
View
Controllers
Presenters
Interactors
Authorizer
Database
Entities

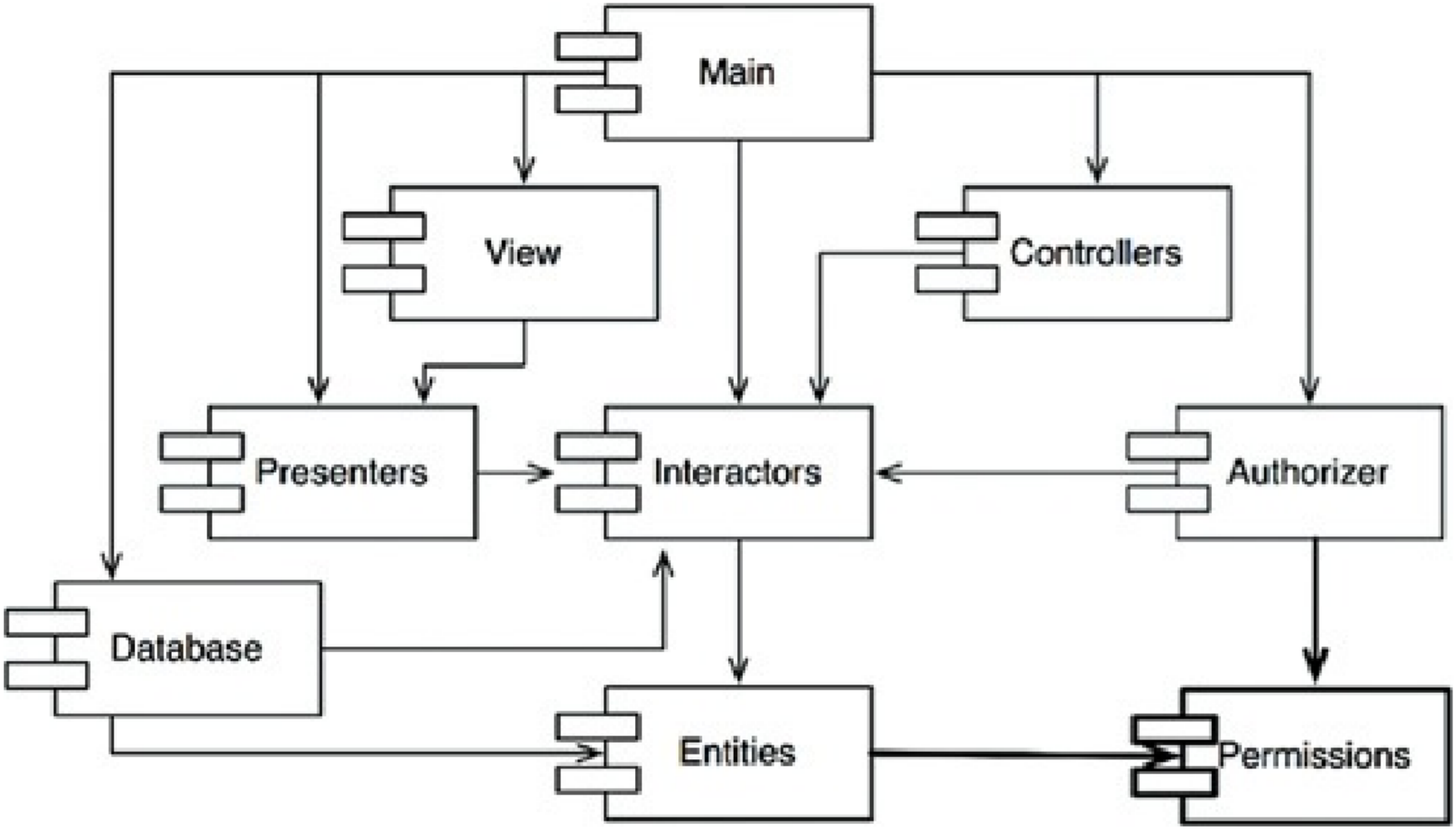
4 - ADP


Main
View
Controllers
Presenters
Interactors
Authorizer
Database
Entities

4 - ADP


Entities
User
Authorizer
Permissions
Entities
User
<I>
Permissions
Authorizer
Permissions

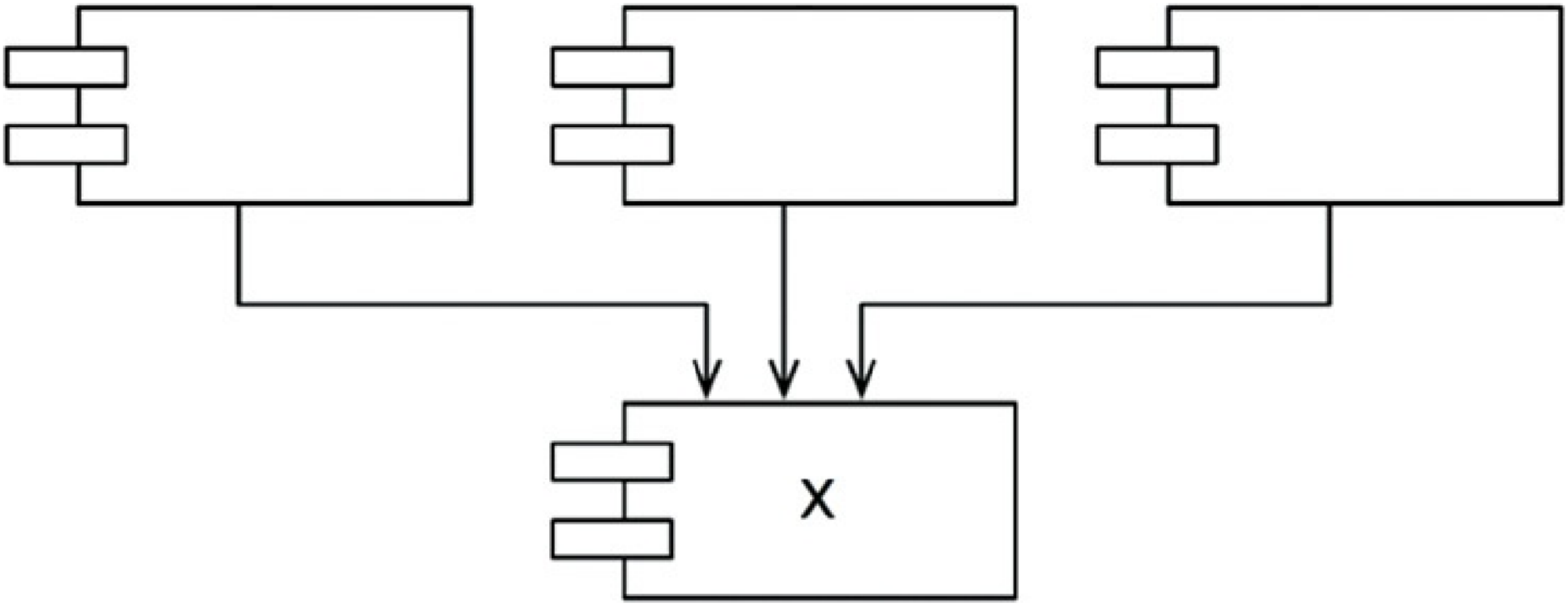
4 - ADP


Main
View
Controllers
Presenters
Interactors
Authorizer
Database
Entities
Permissions

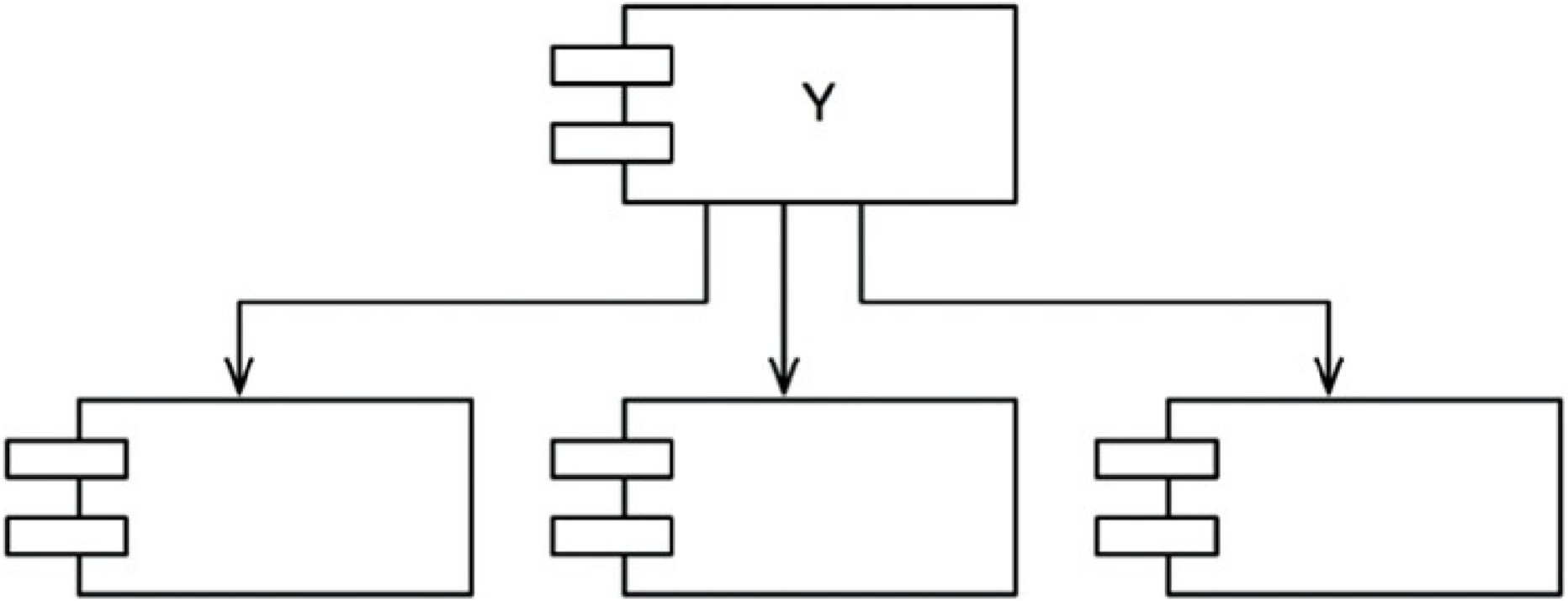
4 - SDP

"Depend on the direction of Stability"

5 - Définition de la stabilité



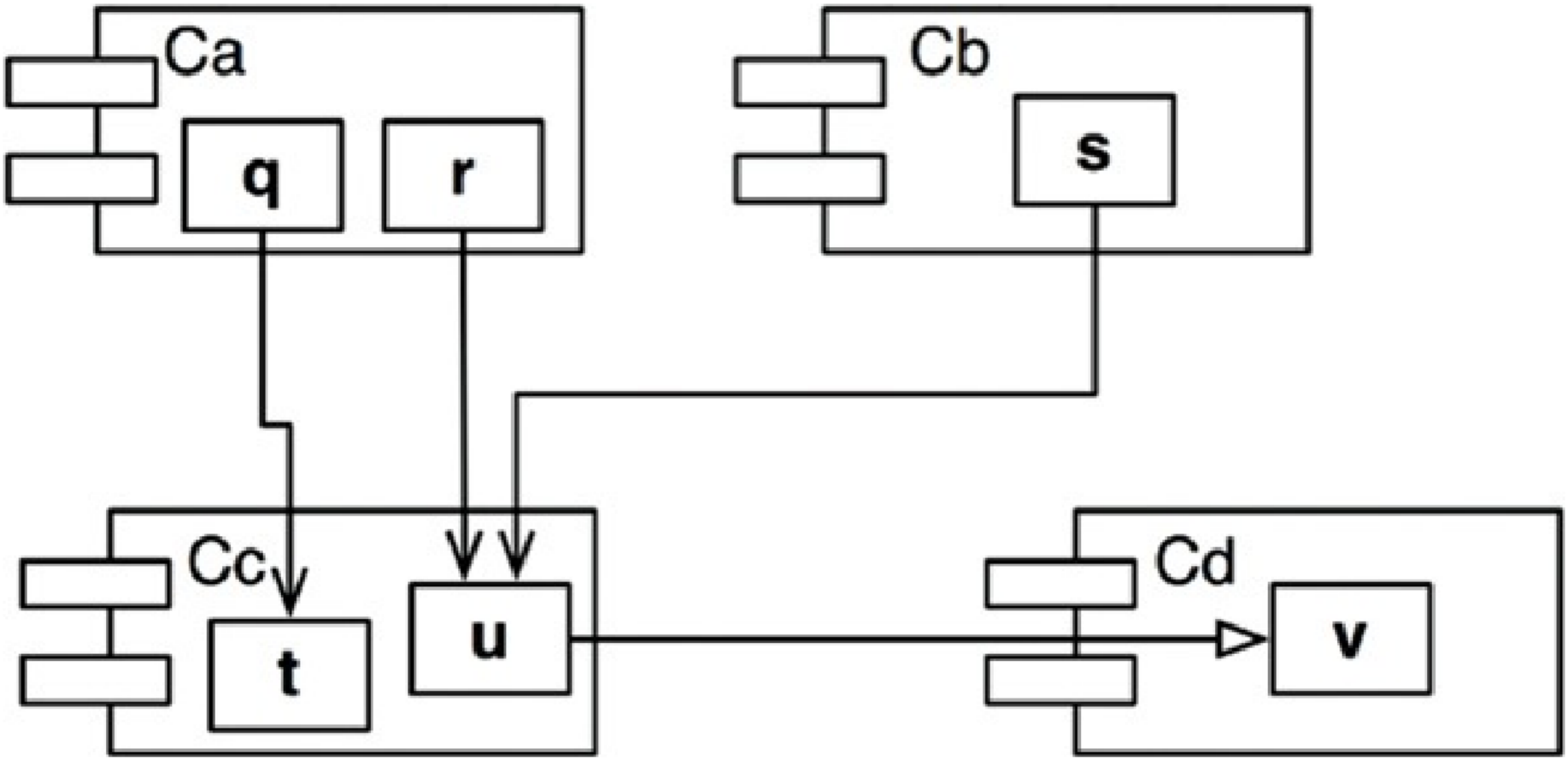
5 - Définition de la stabilité



5 - Définition de la stabilité

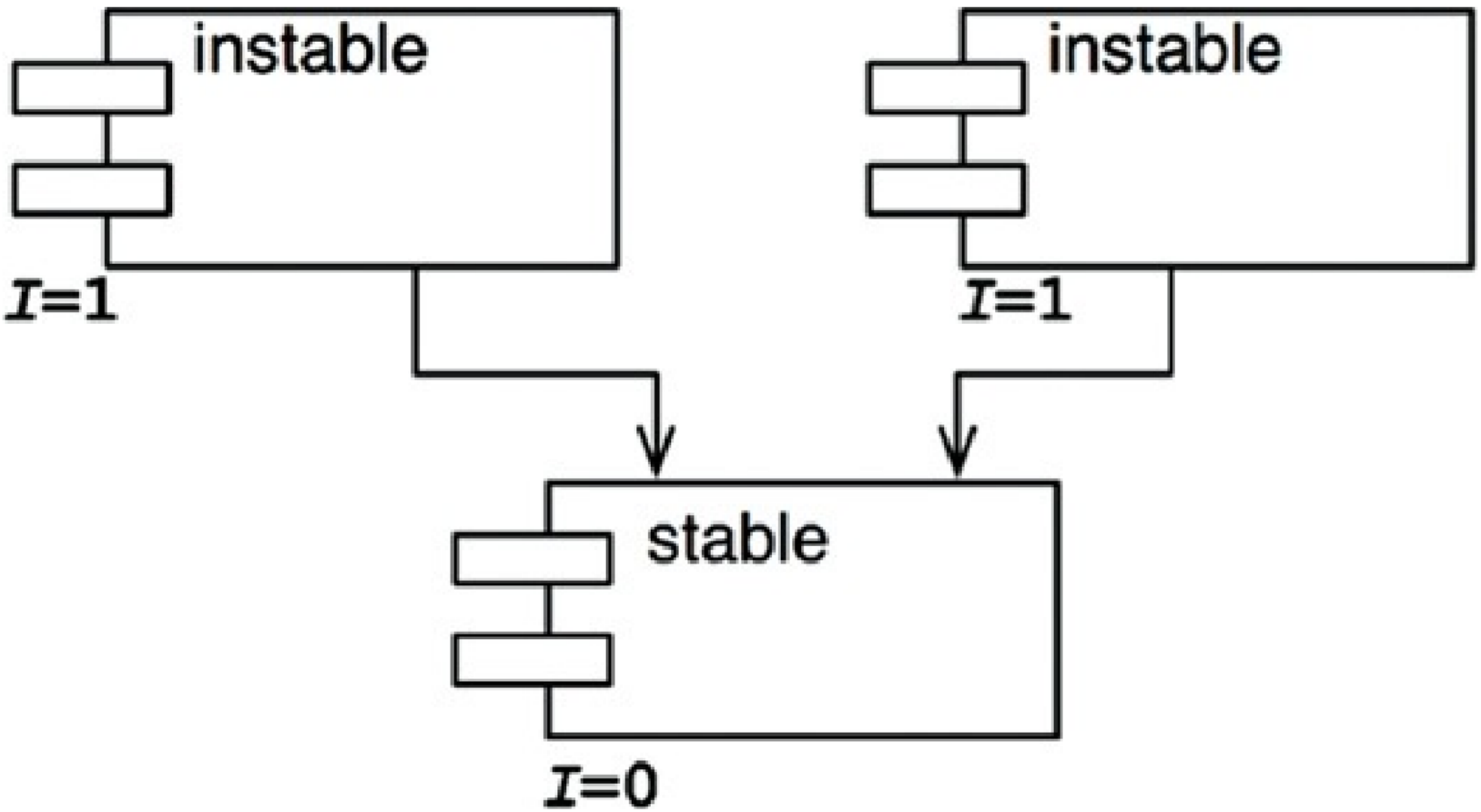
- Fan-in : Incoming dependencies
- Fan-out : Outgoing dependencies
- Instability : $I = \frac{Fan-out}{(Fan-in+Fan-out)} \in [0, 1]$

5 - Définition de la stabilité

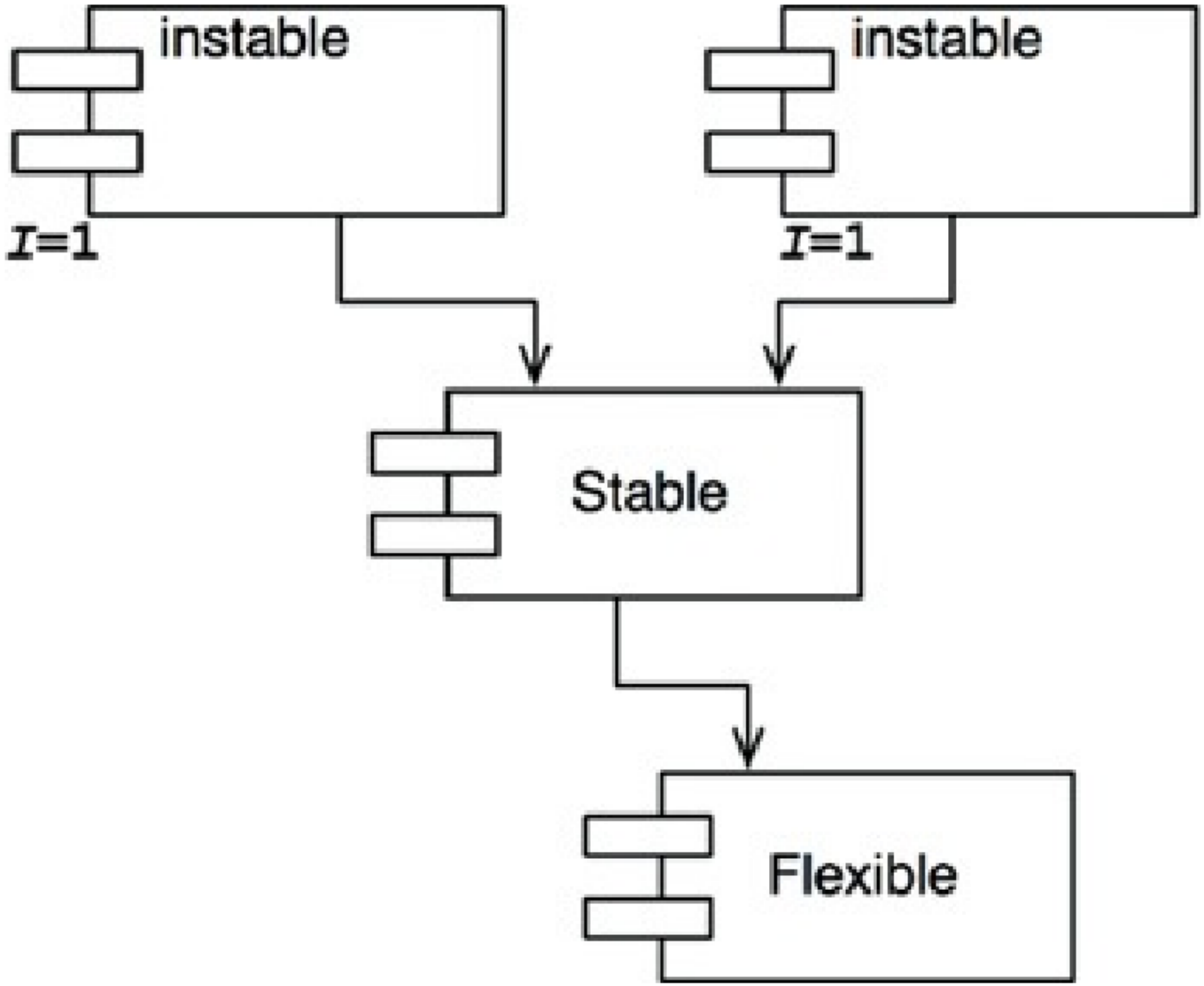


L'instabilité doit décroître si on suit l'ordre de dépendance.

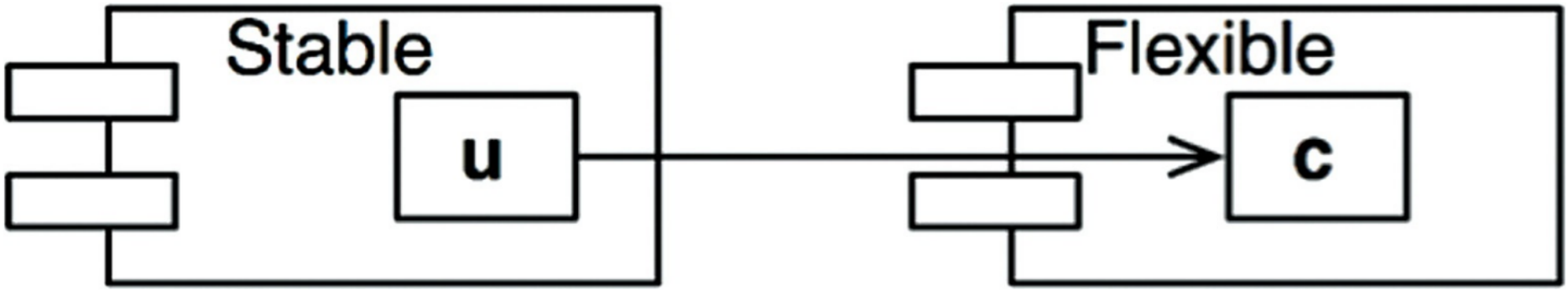
4 - SDP



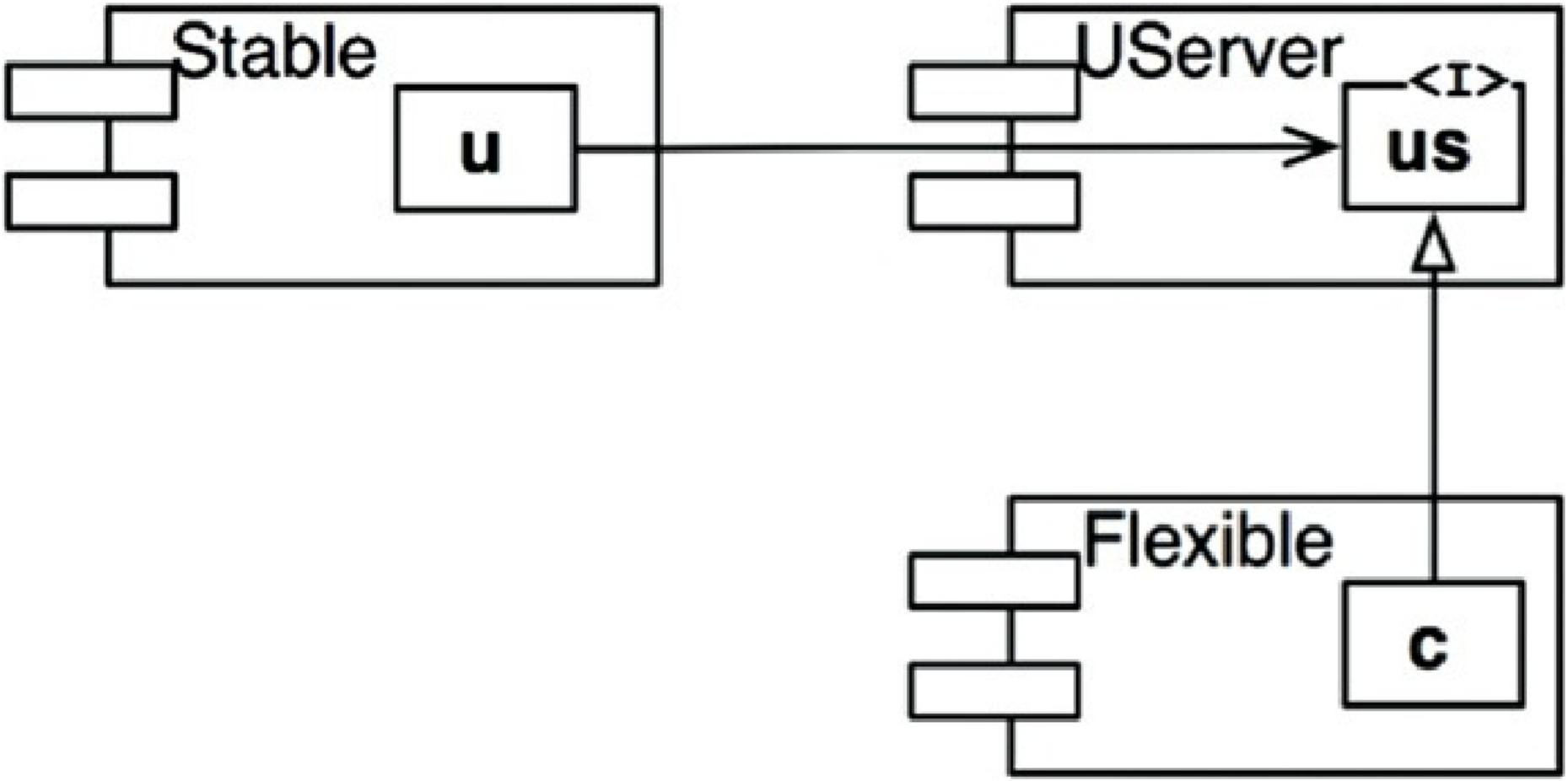
4 - SDP



4 - SDP



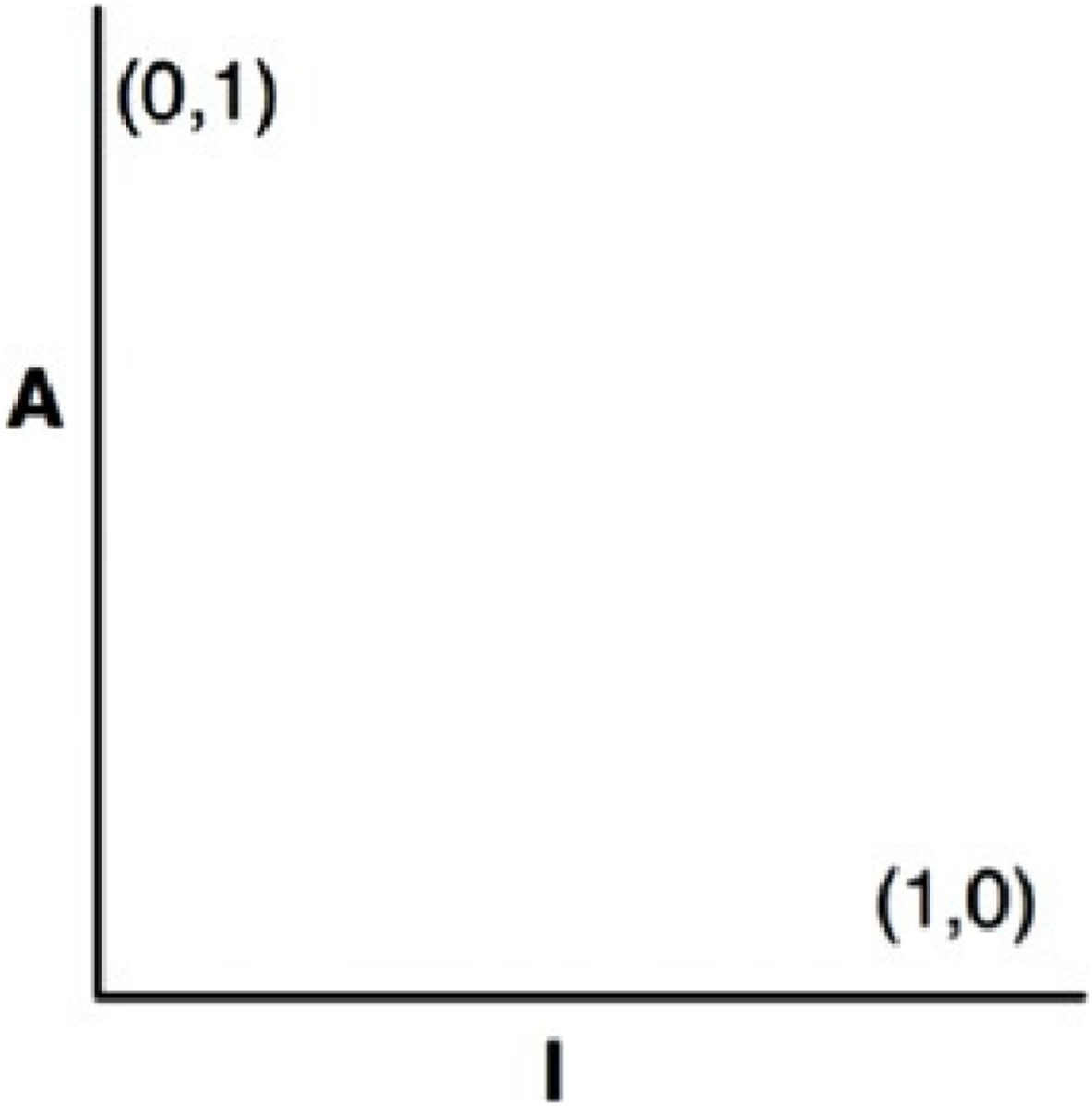
4 - SDP



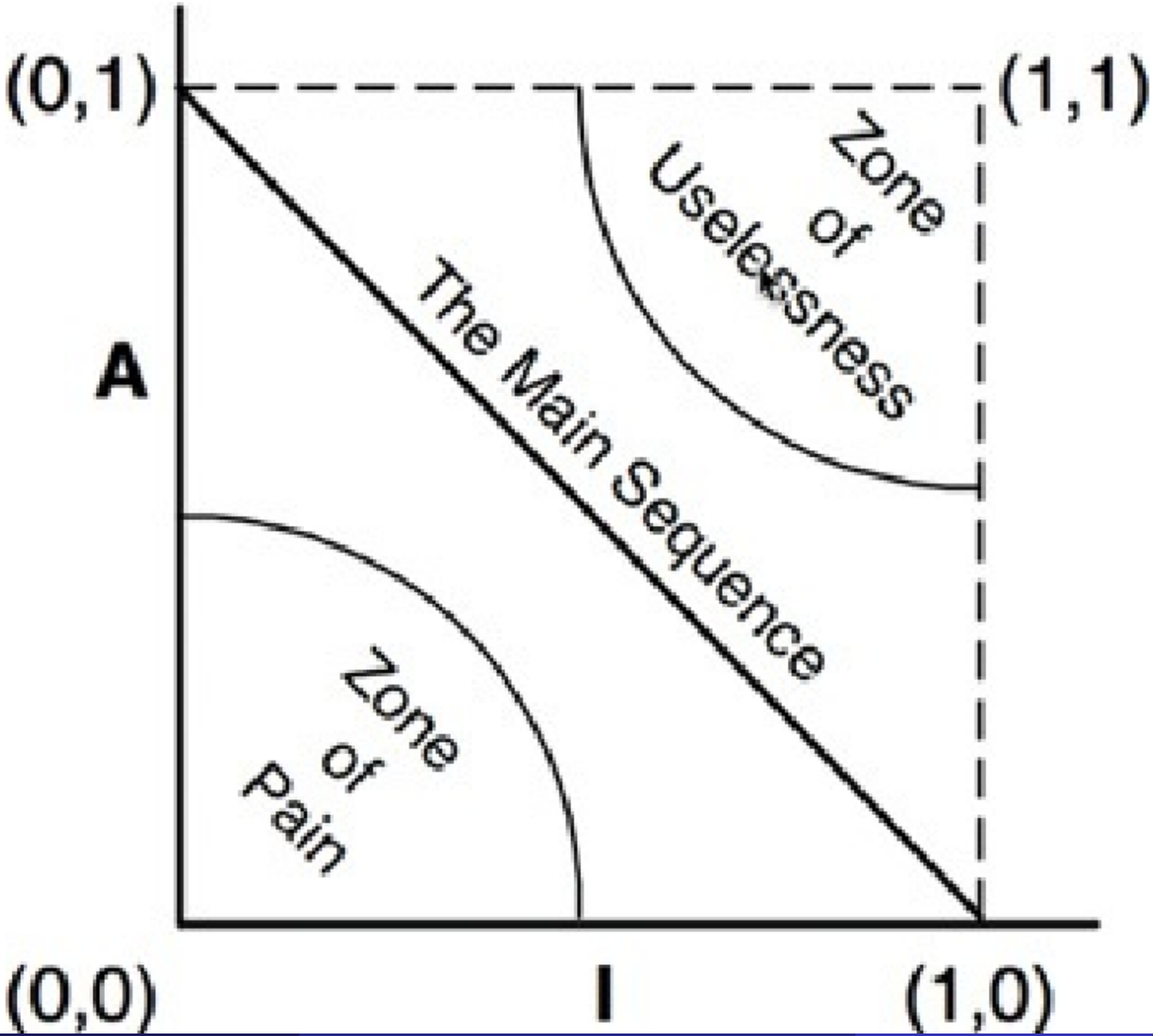
6 - Définition de l'abstraction

- nC : Number of Classes in the component
- nA : Number of Abstract classes in the component
- Abstractness : $A = \frac{nA}{nC} \in [0, 1]$

4 - SAP



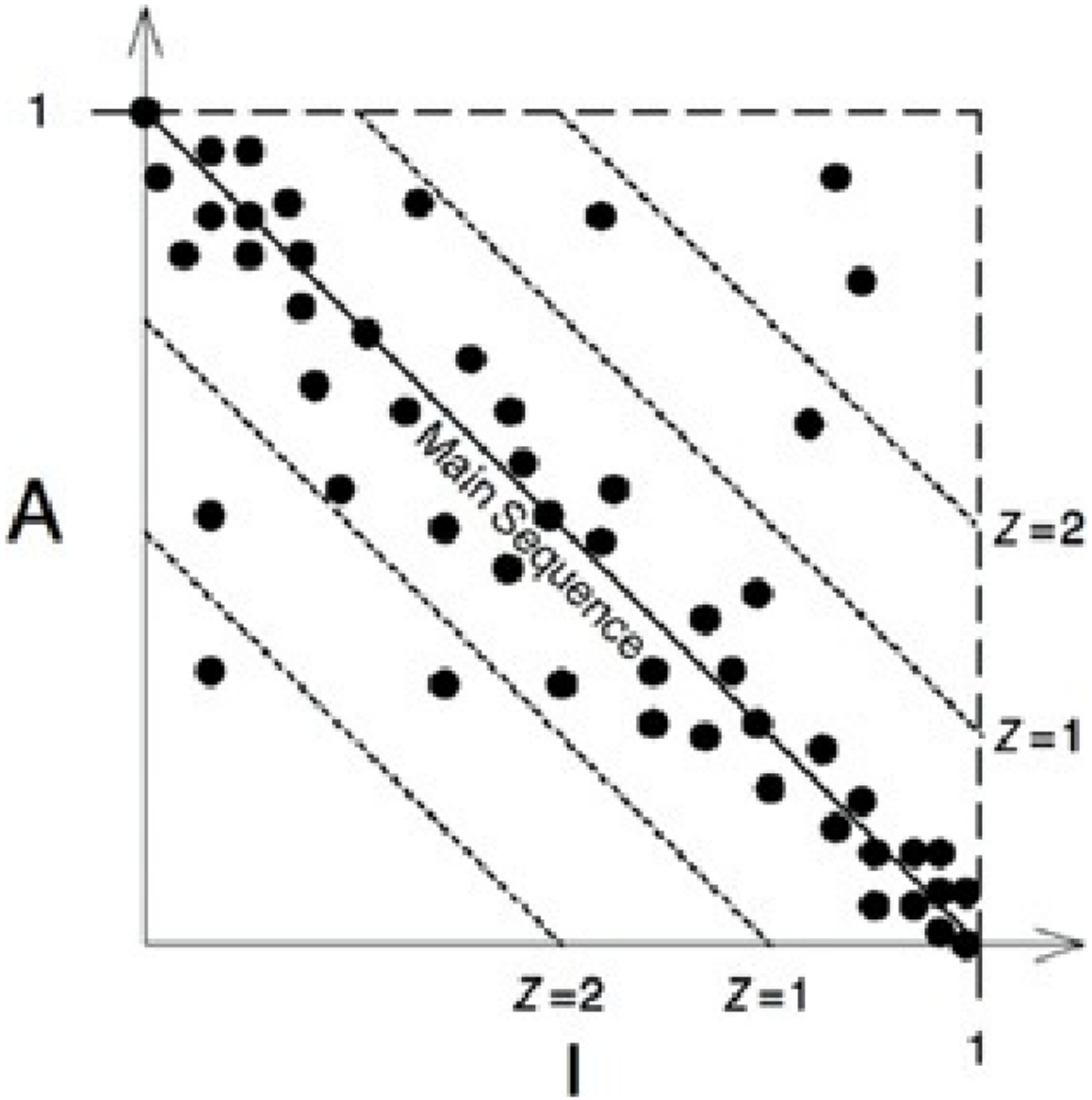
4 - SAP


(0,1)
(1,1)
Zone
of
Uselessness
The Main Sequence
A
Zone
of
Pain
(0,0)
I
(1,0)

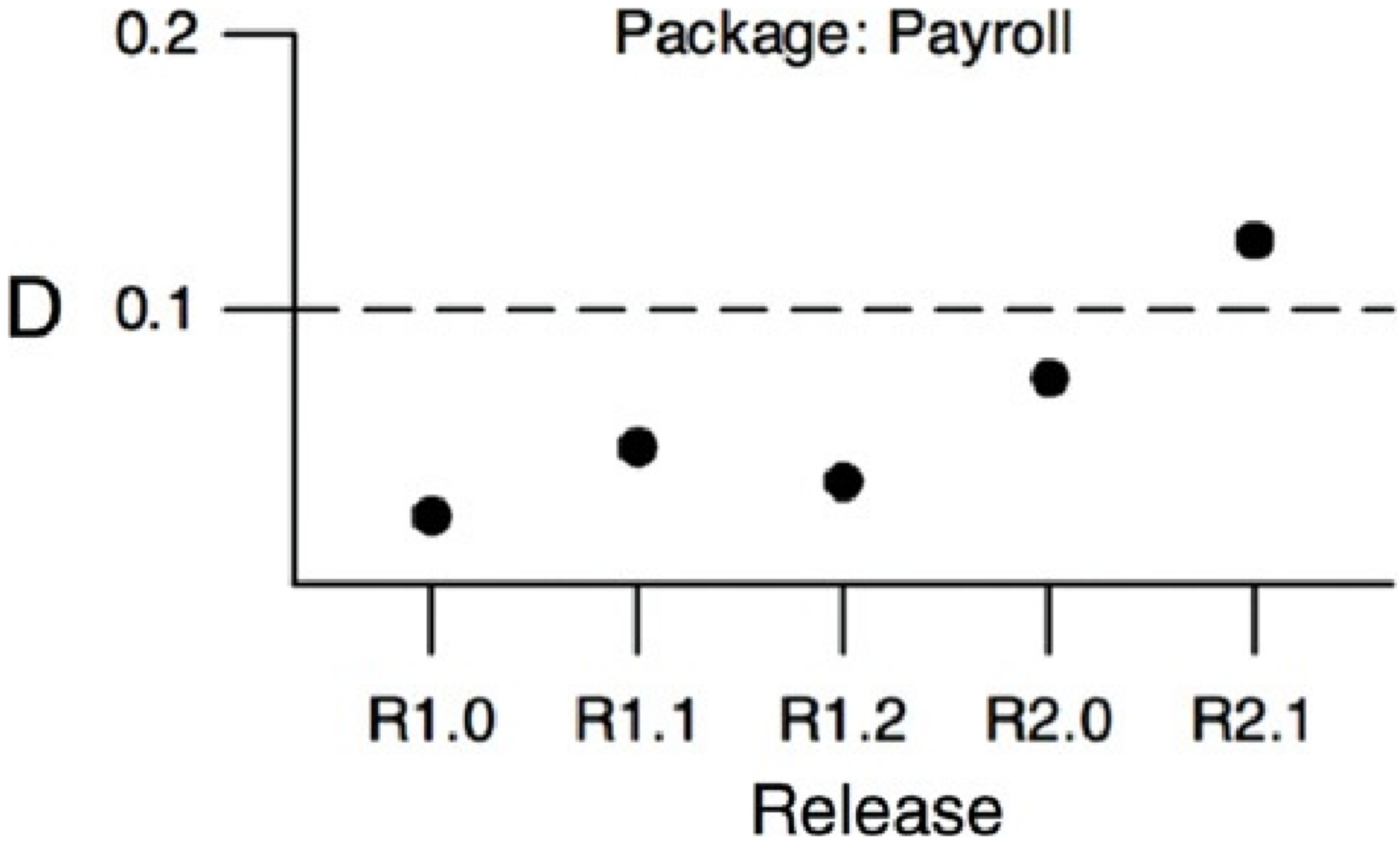
7 - Distance à la séquence principale

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

4 - SAP



4 - SAP



8 - Merc[i]+

Merci beaucoup pour votre attention !