From **Software Architecture** to **N-tiers Architectures**

Sébastien Mosser

Lecture #1.2, 09.02.2018

pictures: sxc.hu

Software Architecture **Definition**

“The **structure** of the system, which comprise **software elements**, externally **visible properties** of those elements, and the **relationships** among them.

[SAiP]

Architecture versus **Design**?

Architecture is a subset of **design**

"External" design

Software Architecture **Objectives**

- It has the **functionality** required by the customer
- It is safely buildable on the **required schedule**
- It **performs adequately**
- It is **reliable**
- It is **usable** and **safe to use**

De la presquitude des choses.

[Plonk et replonk]

thanks to Clèm for the reference

Software Architecture **Concerns**

Producibility

Functionality

Changeability

Security

Modularity

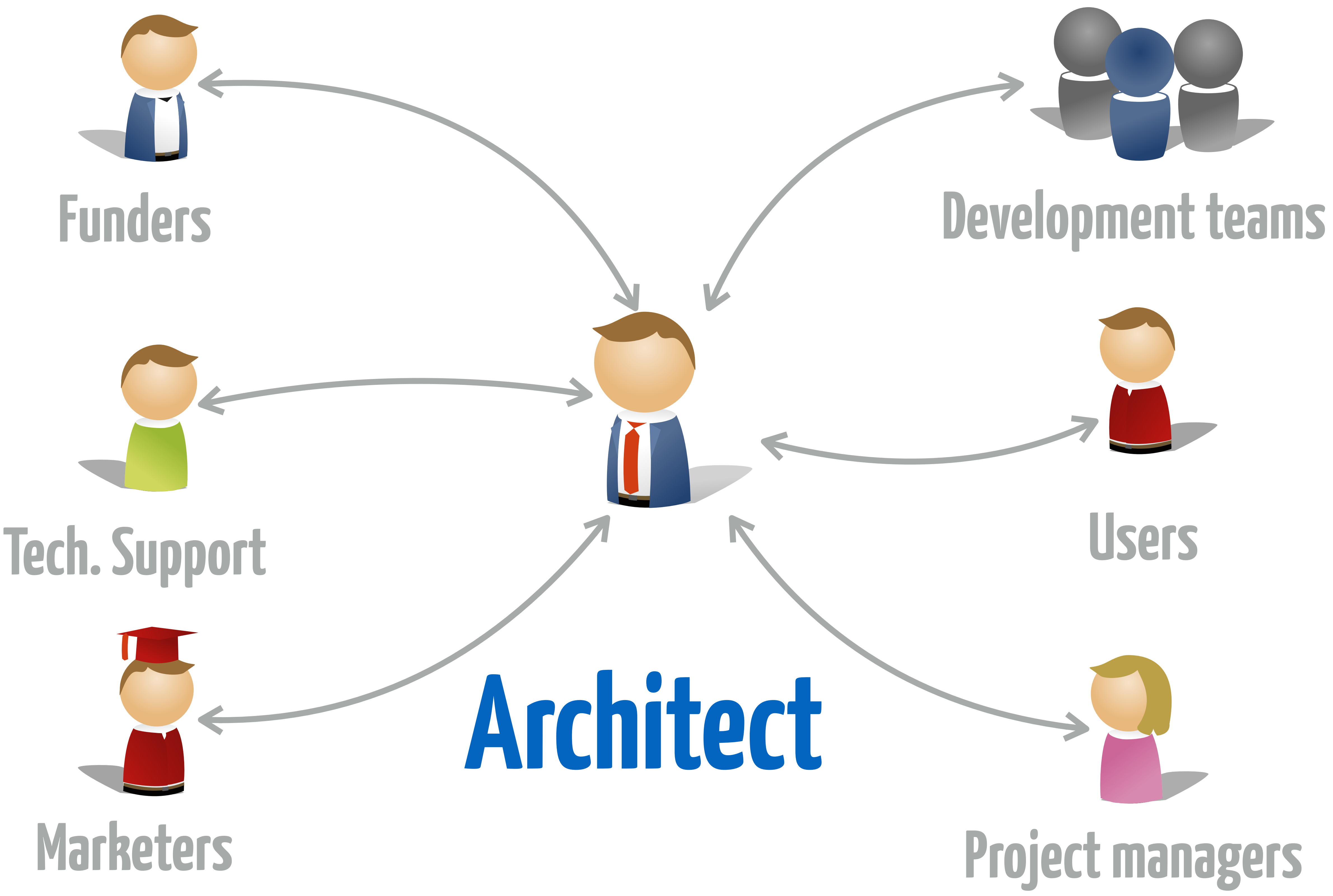
Performance

Ecosystem

Buildability

[BA]

The Architect **Ecosystem**



Expect the **unexpected**!

“ Parameters that were **never** going to **change** now need to be **modified**.

1 Architectural rule of thumb

Layered Architectures 2

3 Support from the UNIFIED MODELING LANGUAGE UML™

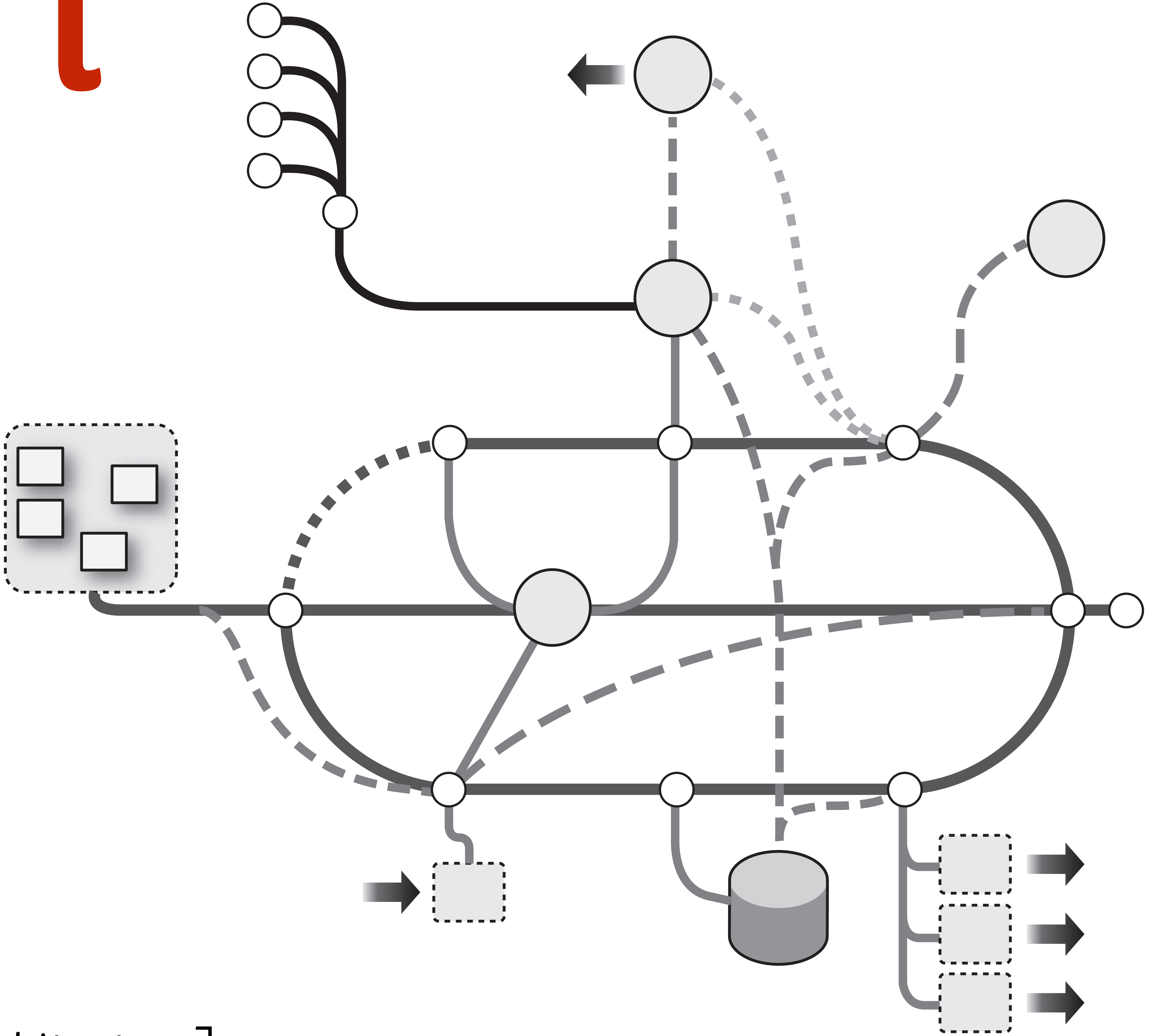
1

Architectural **rule of thumb**

“ Software **architecture** is **not set in stone**. **Change** if you need it.

[Beautiful Architecture]

Don't



[Beautiful Architecture]

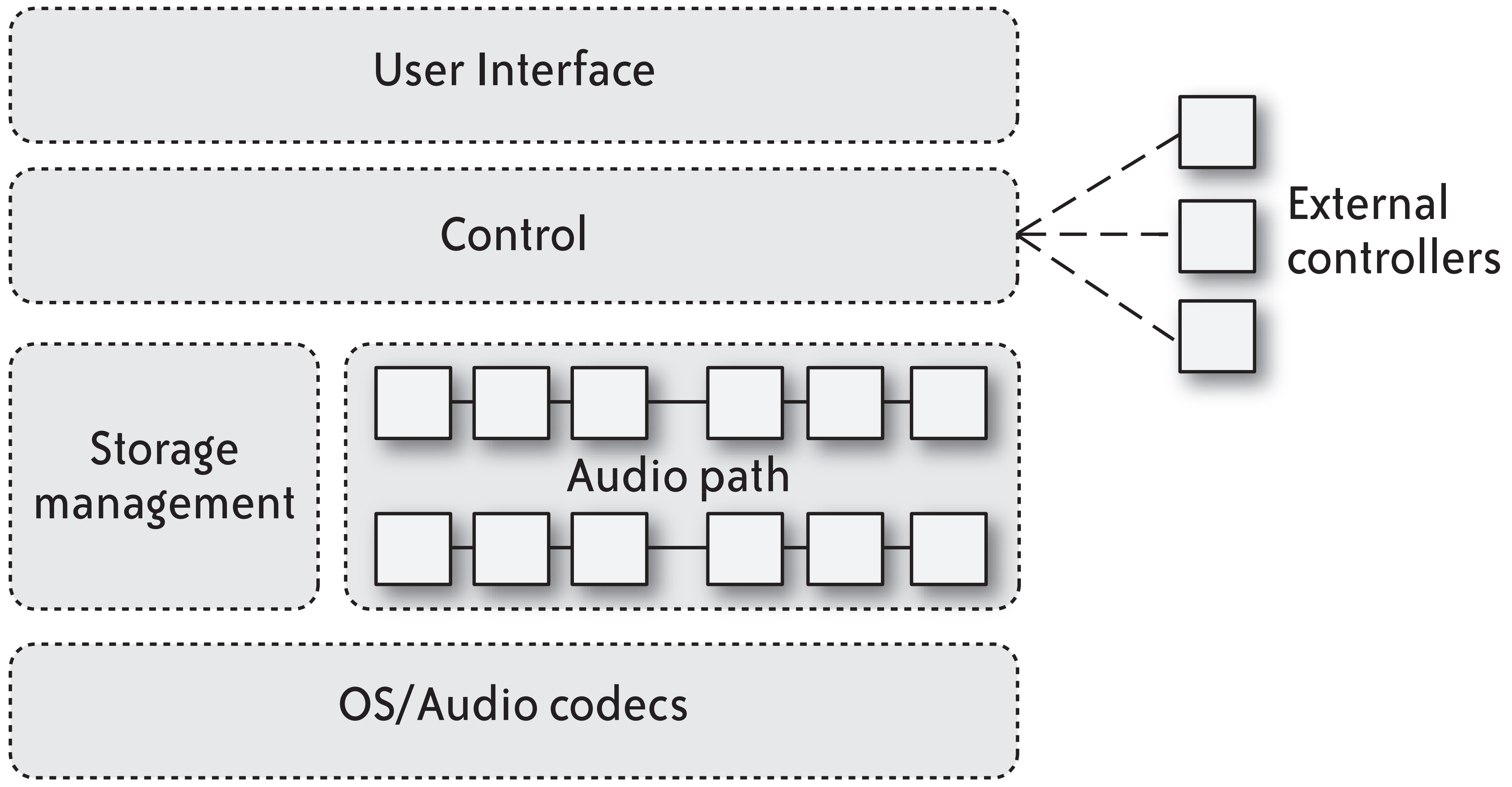
“ A **fuzzy architecture** leads to **individual** code, **duplication** of code and effort.

[Beautiful Architecture]

“ **Bad** architectural **design**
leads to further
bad architectural **design**.

[Beautiful Architecture]

Do



[Beautiful Architecture]

“A **clear architectural** design leads to a **consistent system.**

[Beautiful Architecture]

Cohesion versus Coupling

Cohesion:

"how related functionality is gathered together".

Coupling:

"Measurement of interdependency between modules".

[Beautiful Architecture]

?

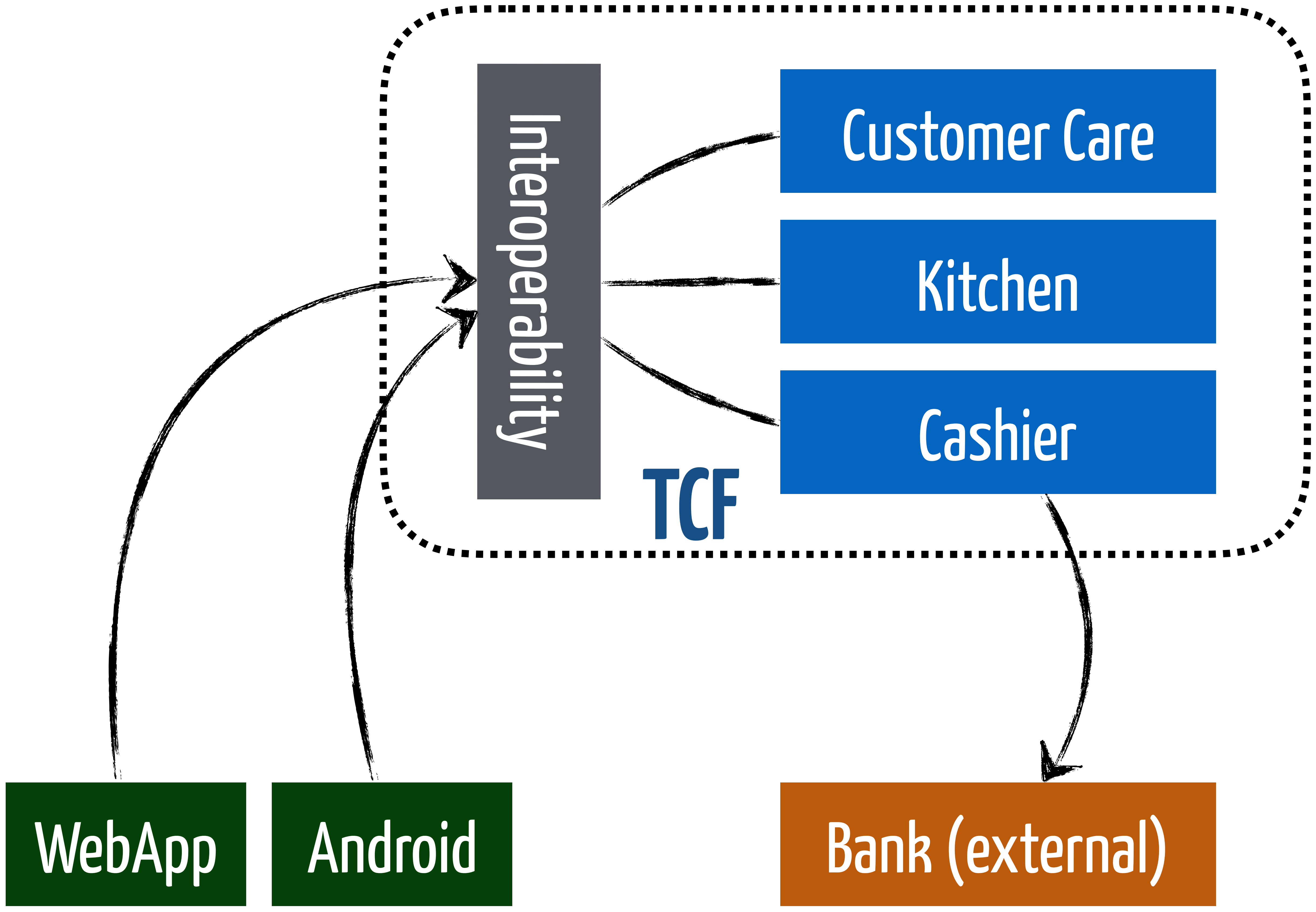
~~Low cohesion,~~
~~Strong coupling~~

Strong cohesion,
Low coupling

[Beautiful Architecture]

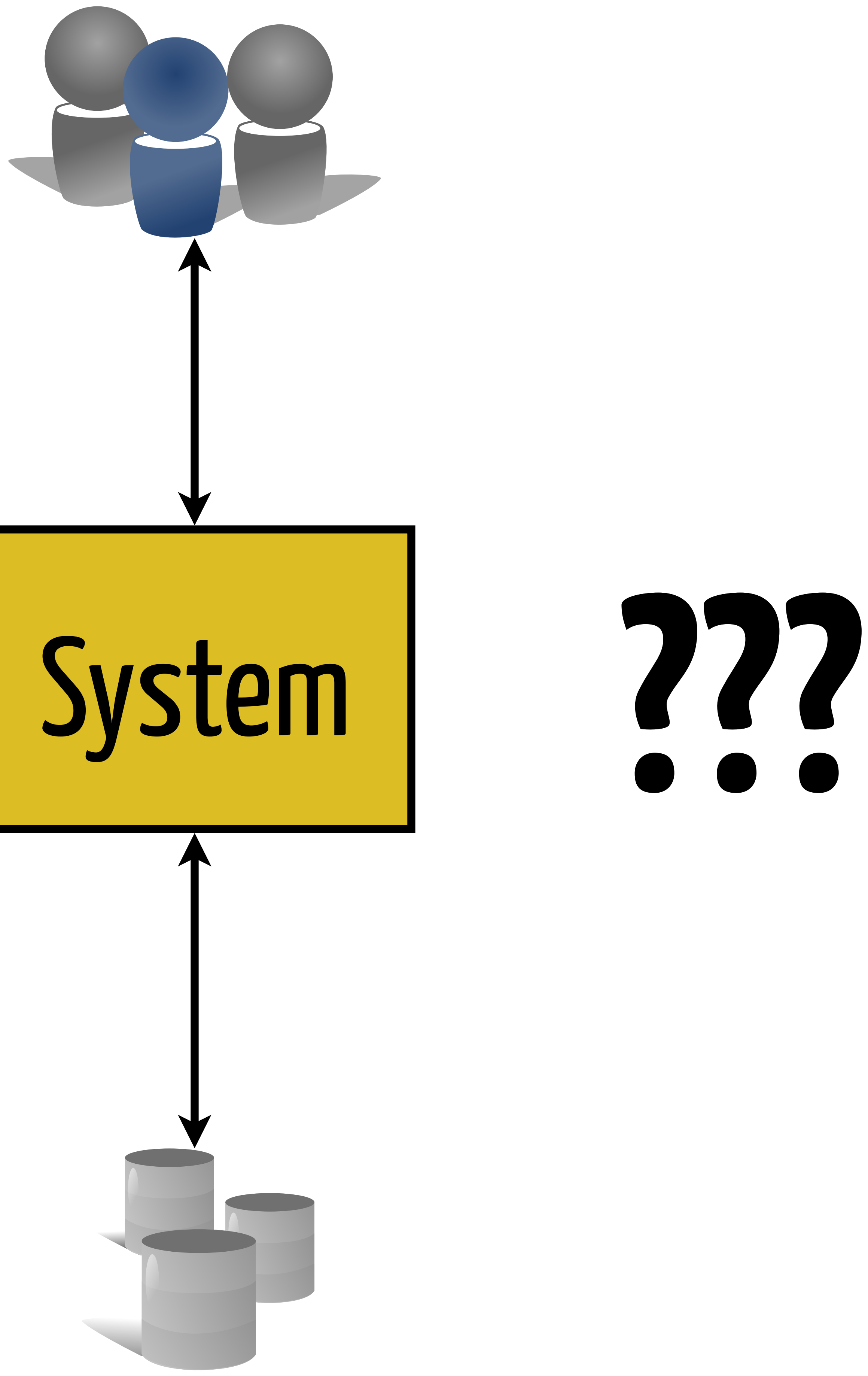
What are the “modules” in TCF?

How are they related to each others?


Interoperability
Customer Care
Kitchen
Cashier
TCF
WebApp
Android
Bank (external)

Layered Architectures

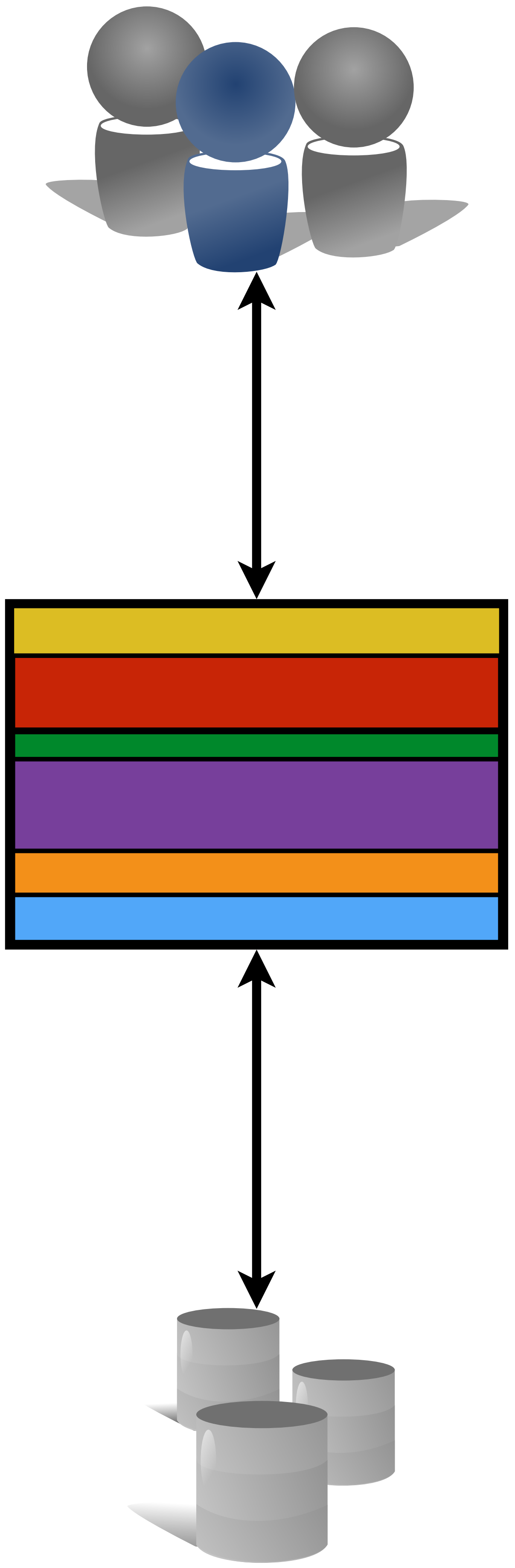
2


System
???

Layers

support

modularity



“Ma-gni-fique”

thanks to Clèm for the reference

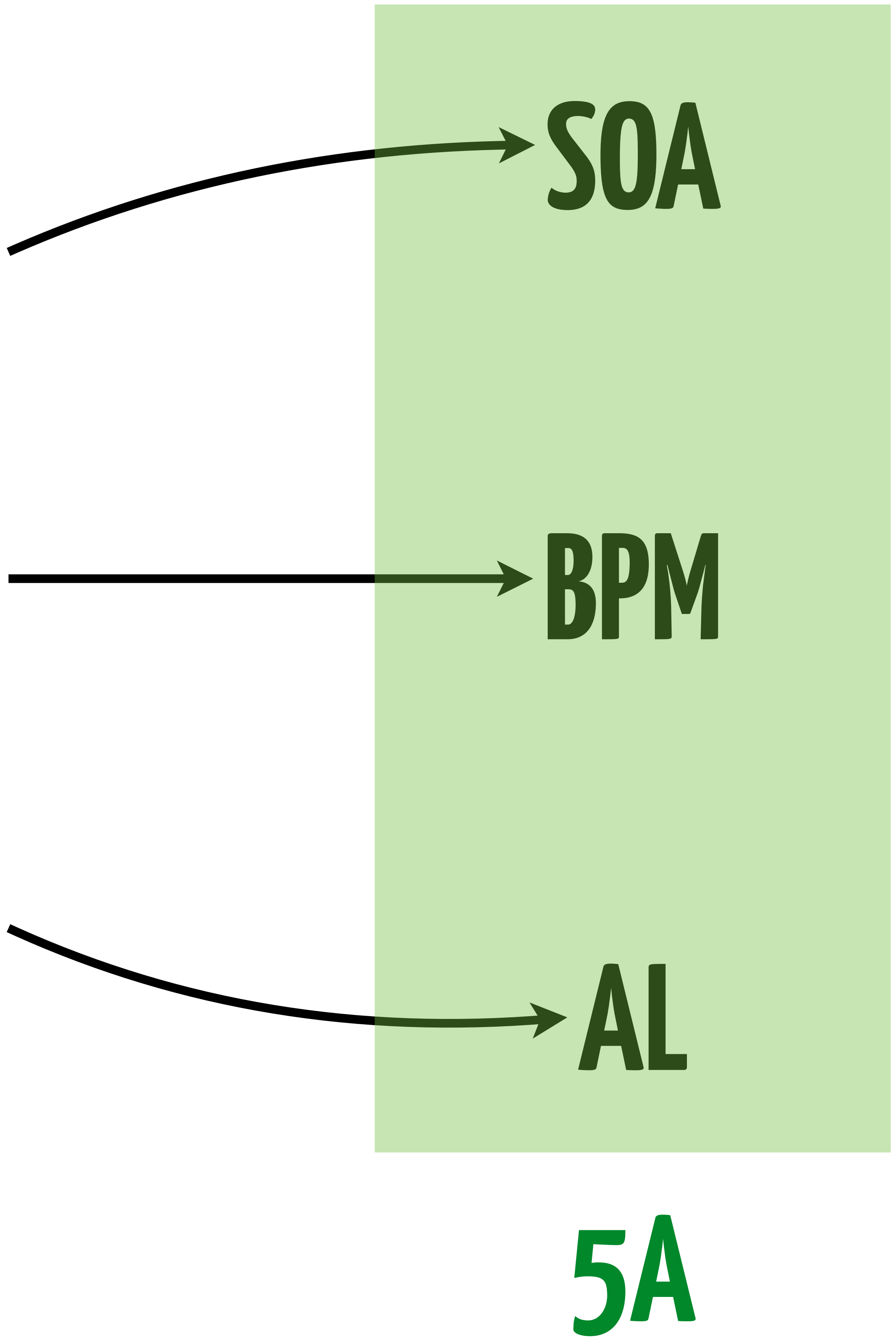
K eep

I t

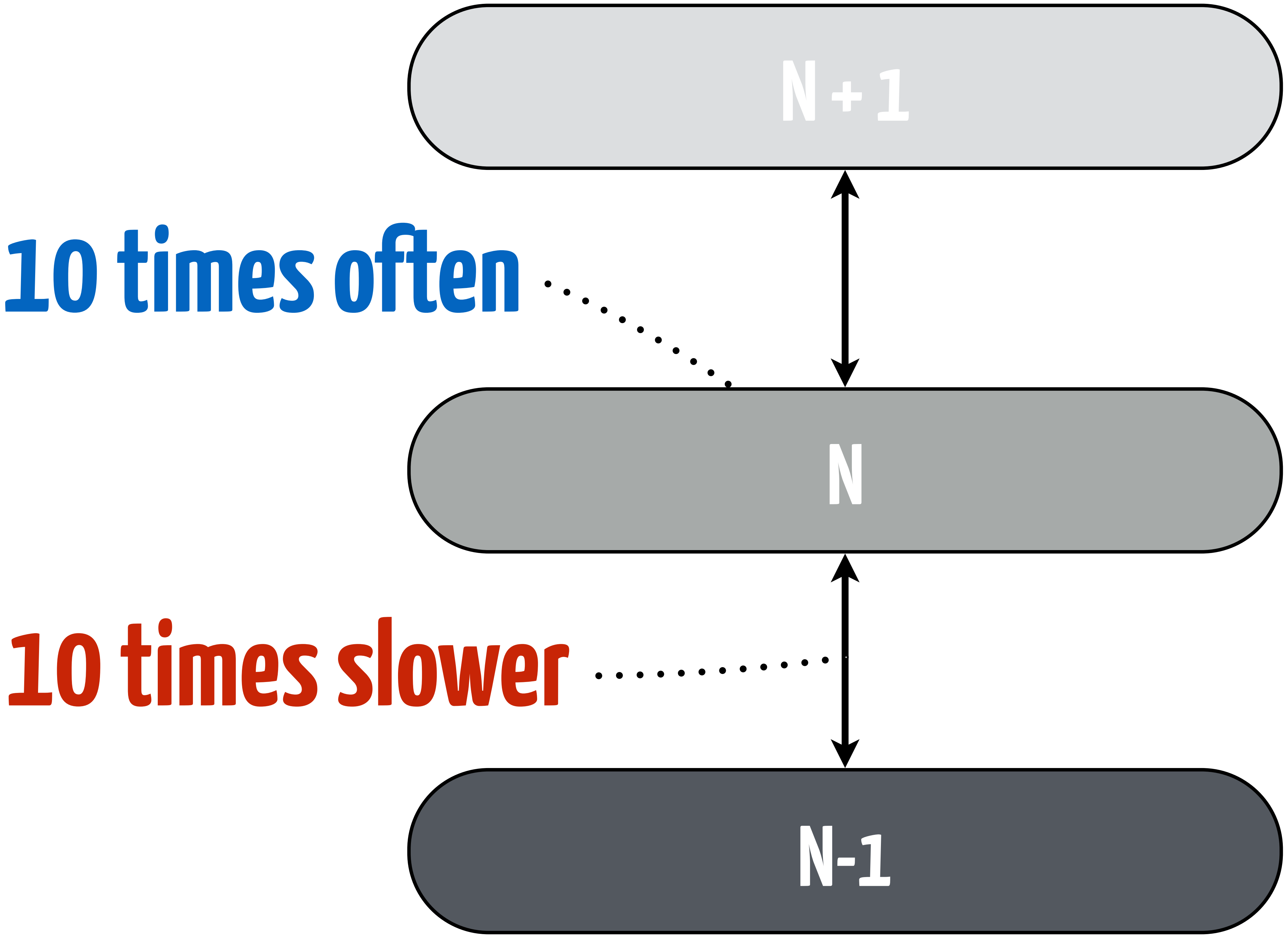
S tupid,

S imple.

4A



The rule of 10



Theory & **Practice**

Theory:

you know everything but nothing works

Practice:

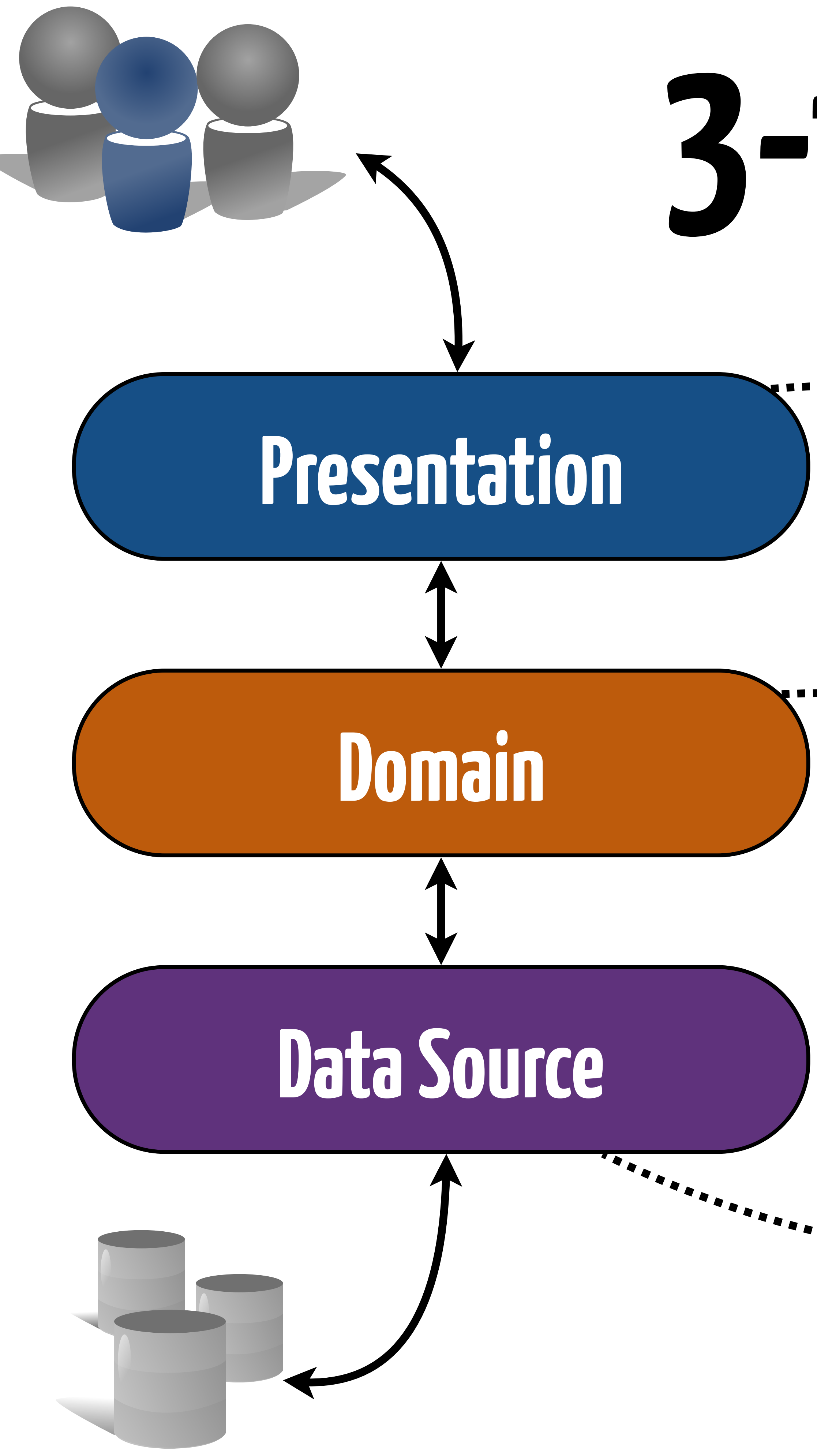
It works, but no one knows why

In theory, **N-tiers** architecture

In practice, N = 3

(or 5)

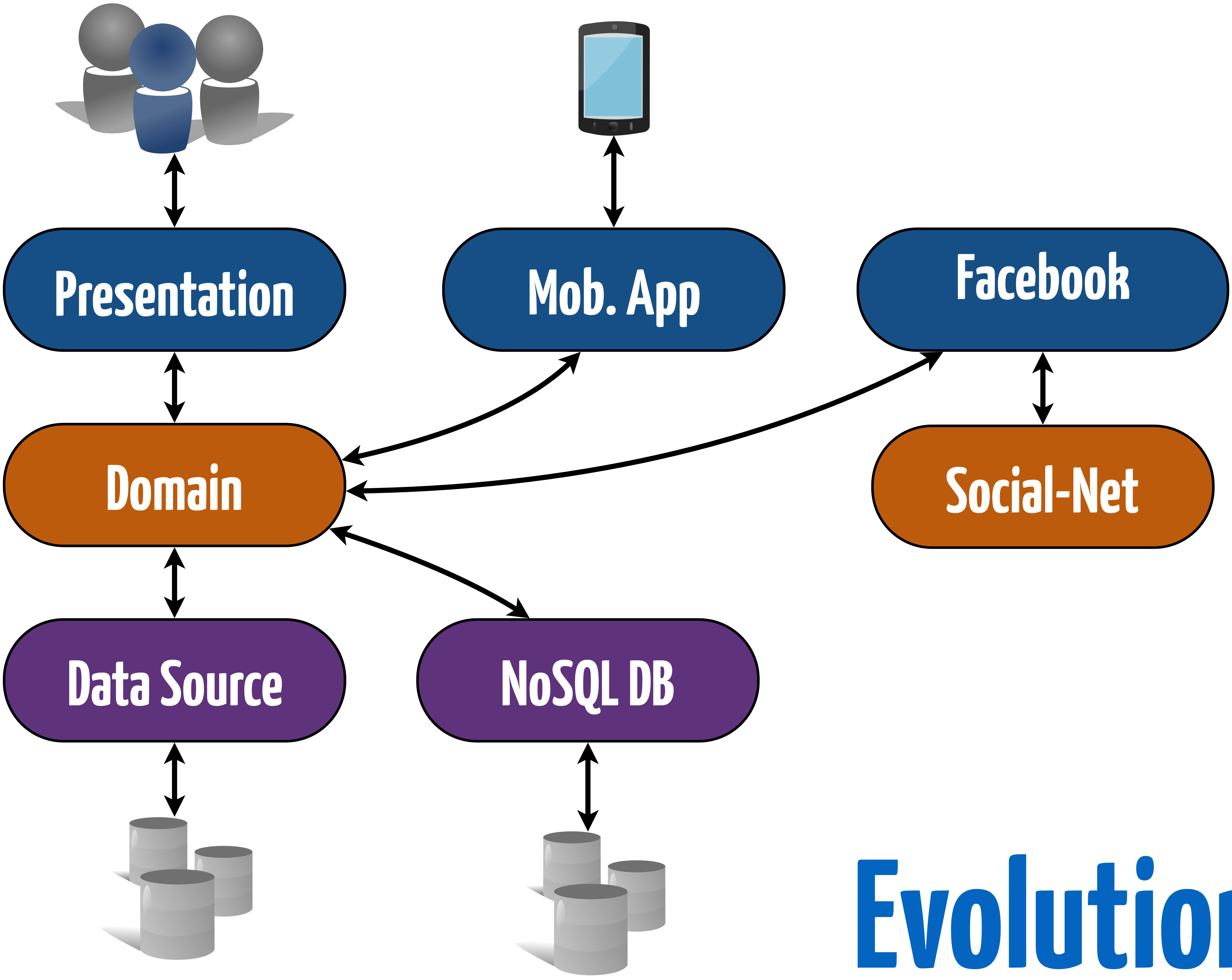
3-tiers architecture



Handle users

Business functionality

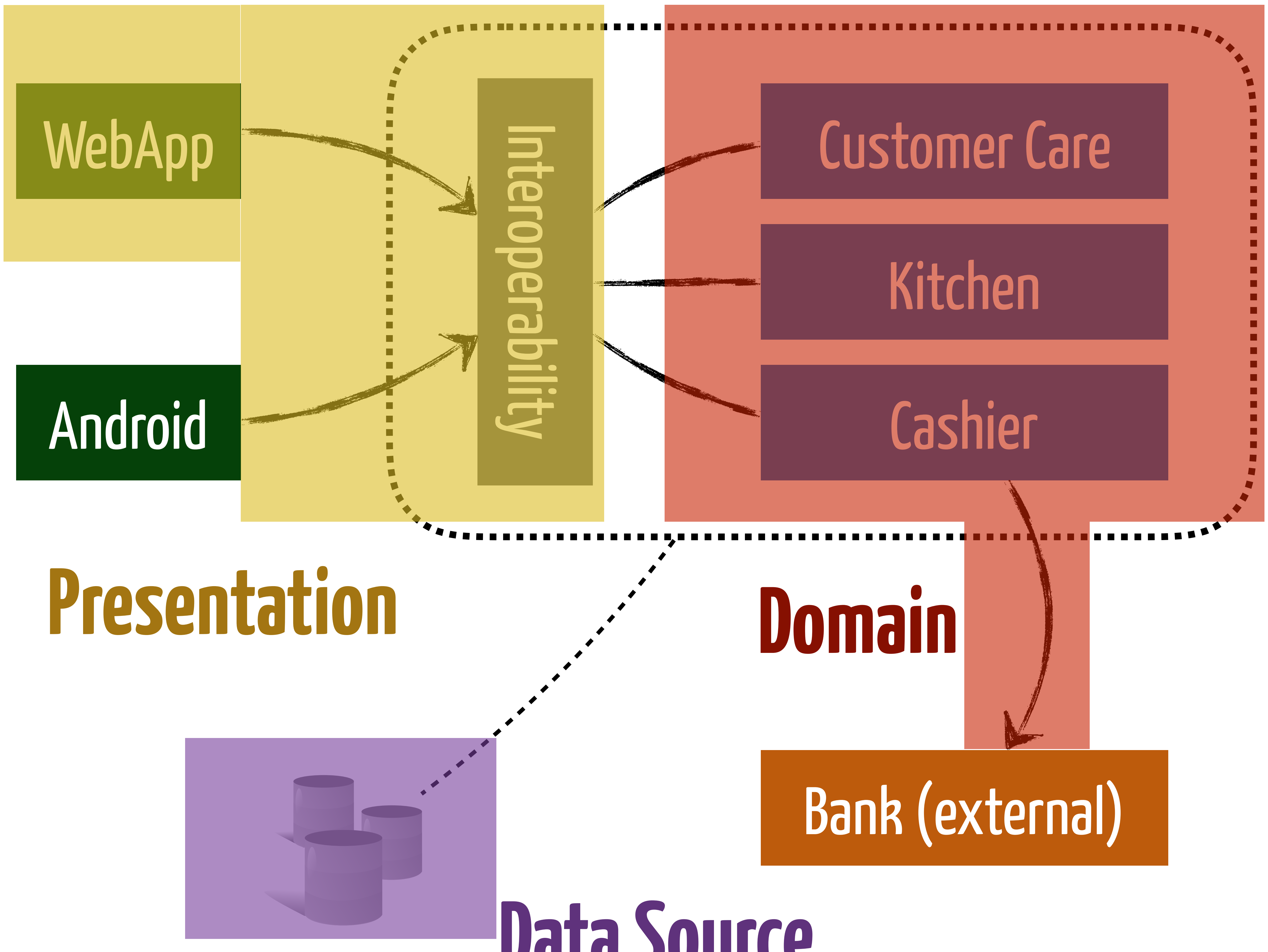
Handle data storage


Presentation
Mob. App
Facebook
Domain
Social-Net
Data Source
NoSQL DB

Evolution

Contents of the ≠ layers in TCF?

How to chose between layers?


WebApp
Android
Interoperability
Customer Care
Kitchen
Cashier
Presentation
Domain
Bank (external)
Data Source

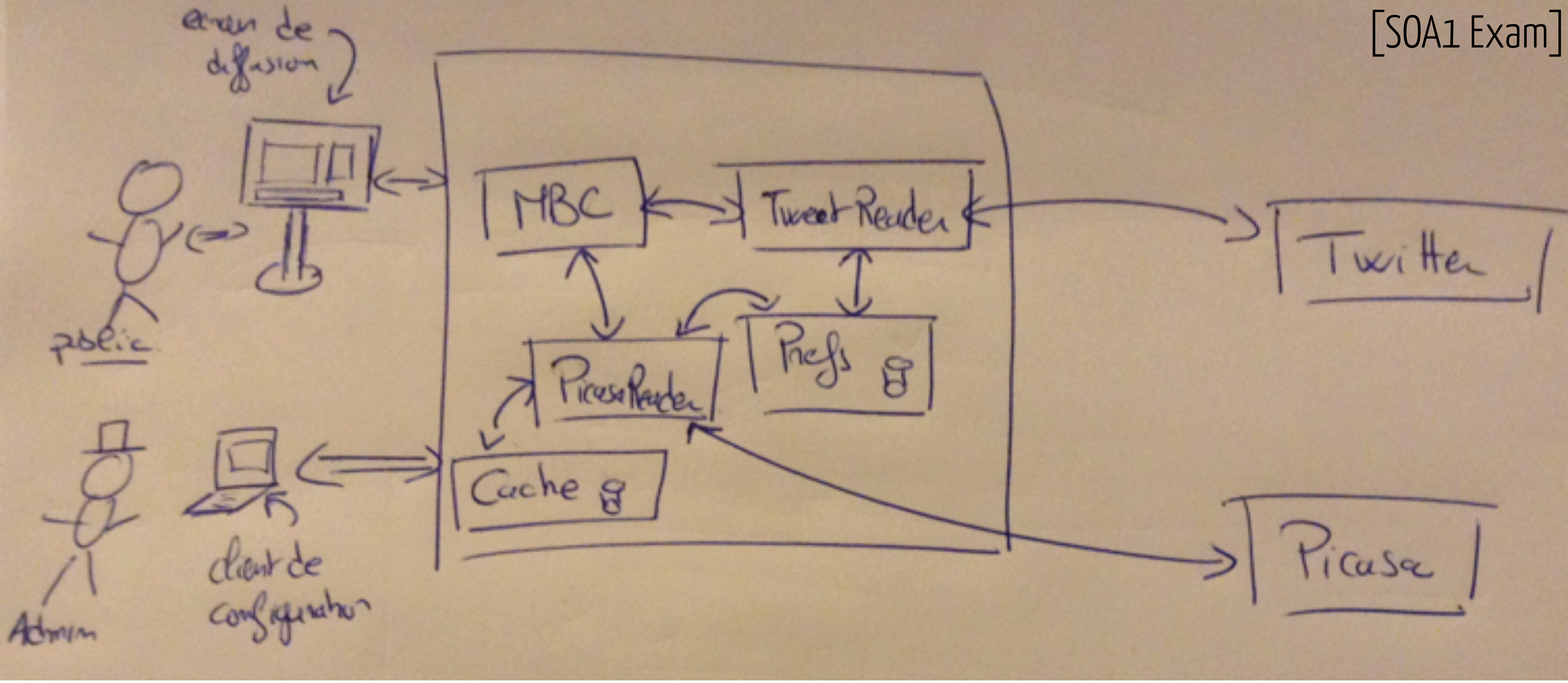
Support from the

UNIFIED
MODELING
LANGUAGE

UML™

3

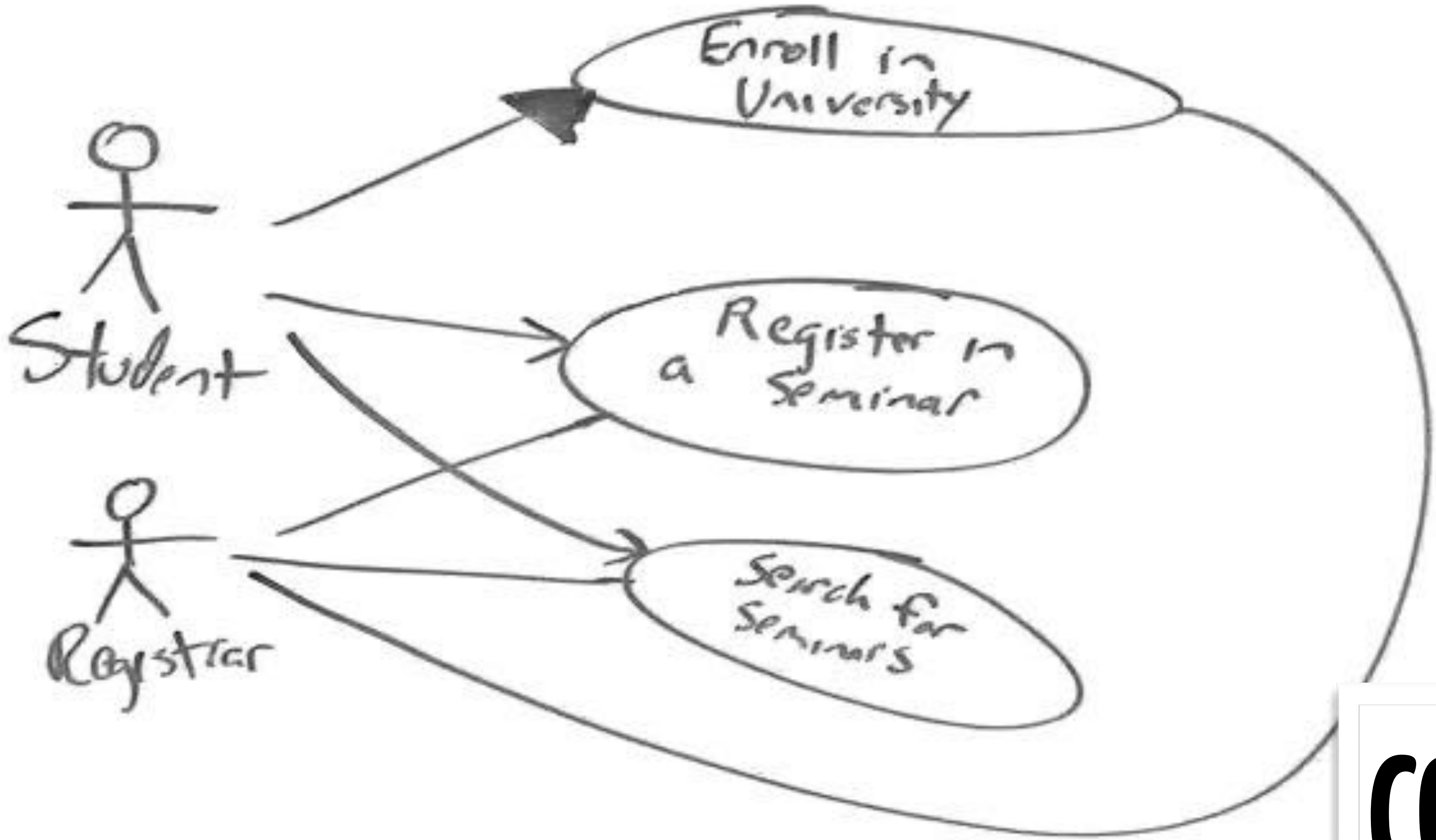
The UML is just a standard syntax for modeling



One can design a Software Architecture

without the UML

Use cases diagrams

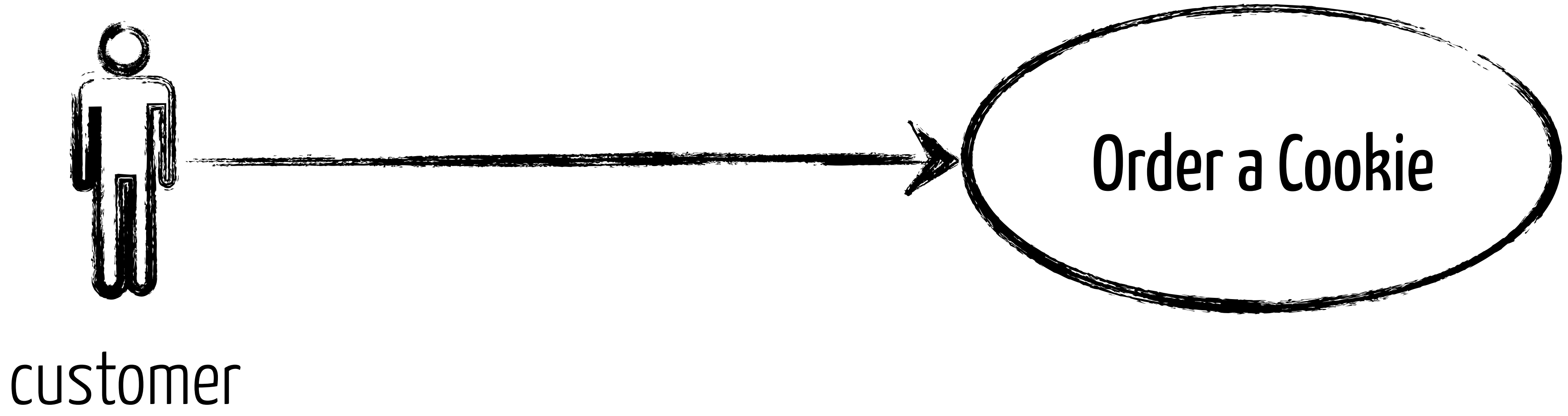


COO

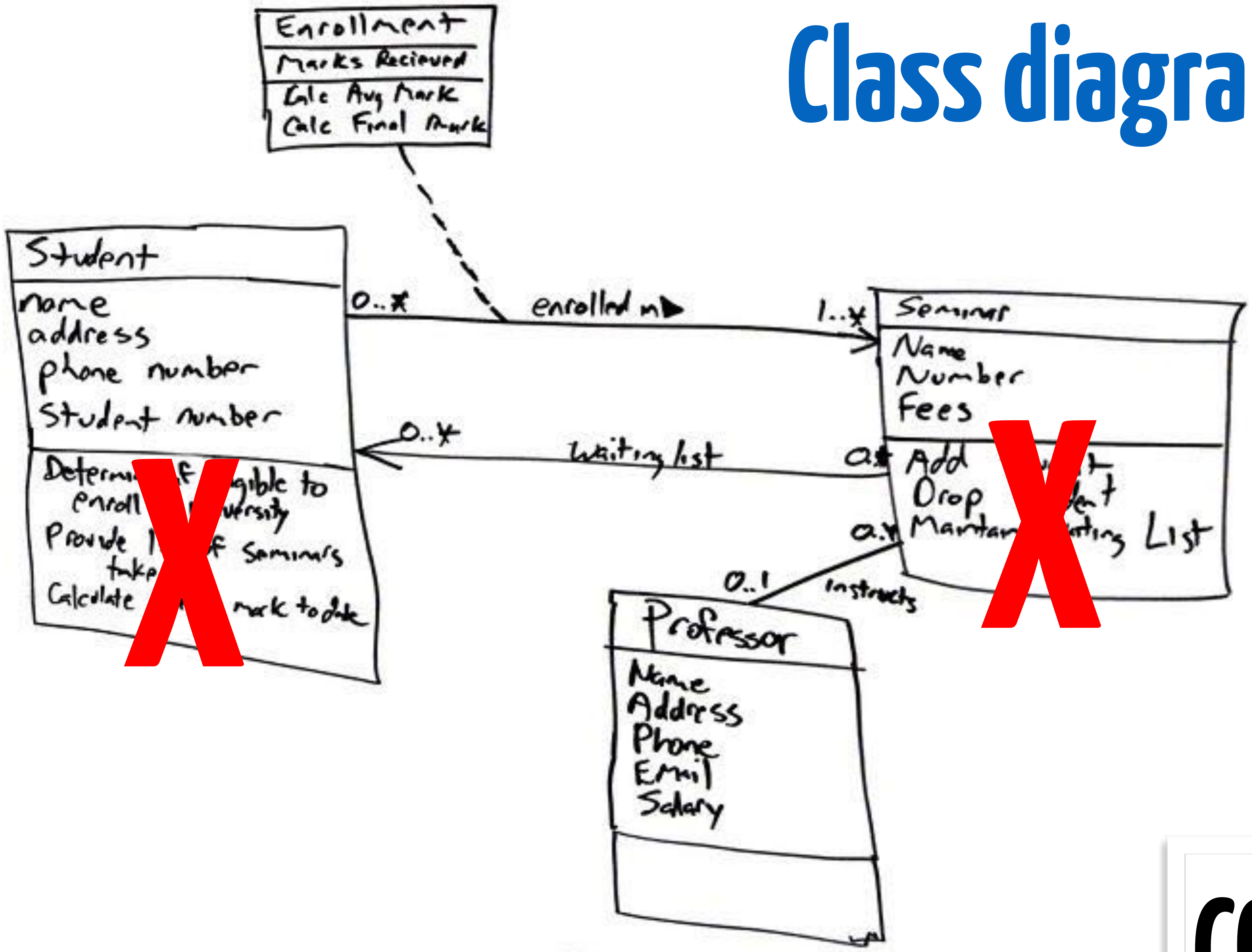
http://www.agilemodeling.com/artifacts/useCaseDiagram.htm

TCF MVP

Minimal & Viable Use Case ?


Order a Cookie
customer

Class diagrams



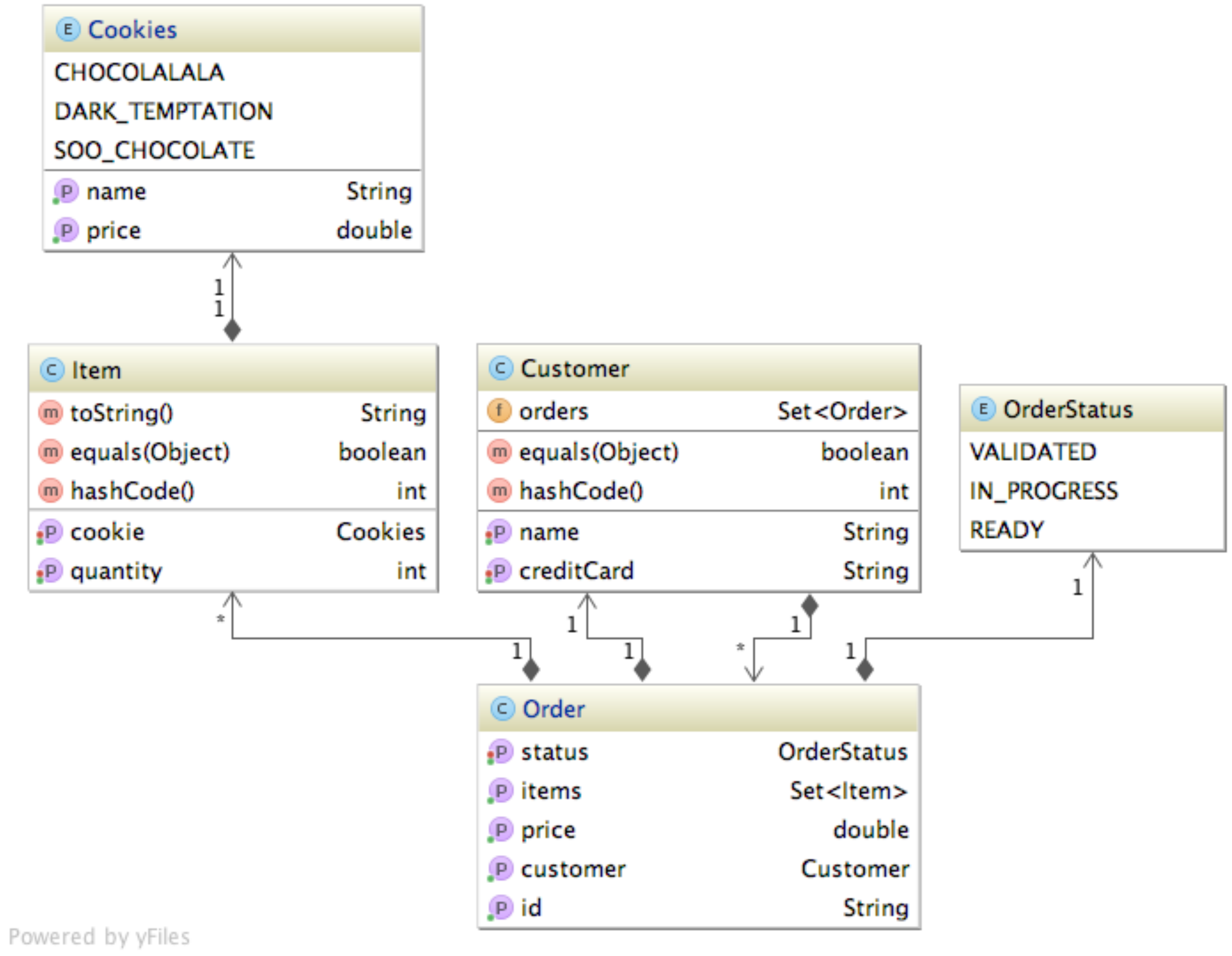
Business objects => **no methods**

http://www.agilemodeling.com/artifacts/classDiagram.htm

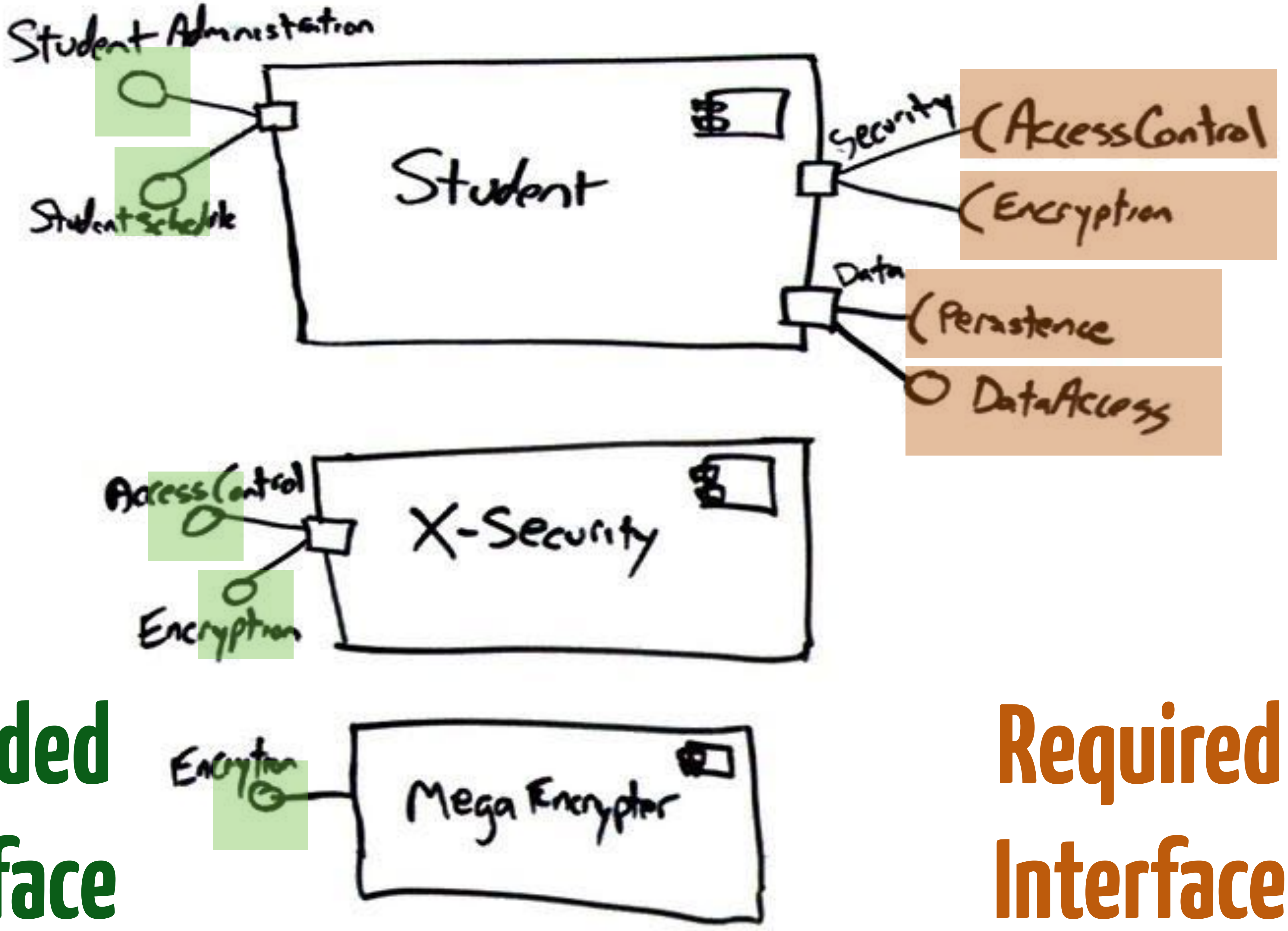
COO

TCF MVP

Business Objects (≠ “objects” as in COO)


Cookies
CHOCOLALALA
DARK_TEMPTATION
SOO_CHOCOLATE
name String
price double
1
1
Item
toString() String
equals(Object) boolean
hashCode() int
cookie Cookies
quantity int
Customer
orders Set<Order>
equals(Object) boolean
hashCode() int
name String
creditCard String
OrderStatus
VALIDATED
IN_PROGRESS
READY
*
1
1
1
1
*
1
1
1
Order
status OrderStatus
items Set<Item>
price double
customer Customer
id String
Powered by yFiles

Components diagrams



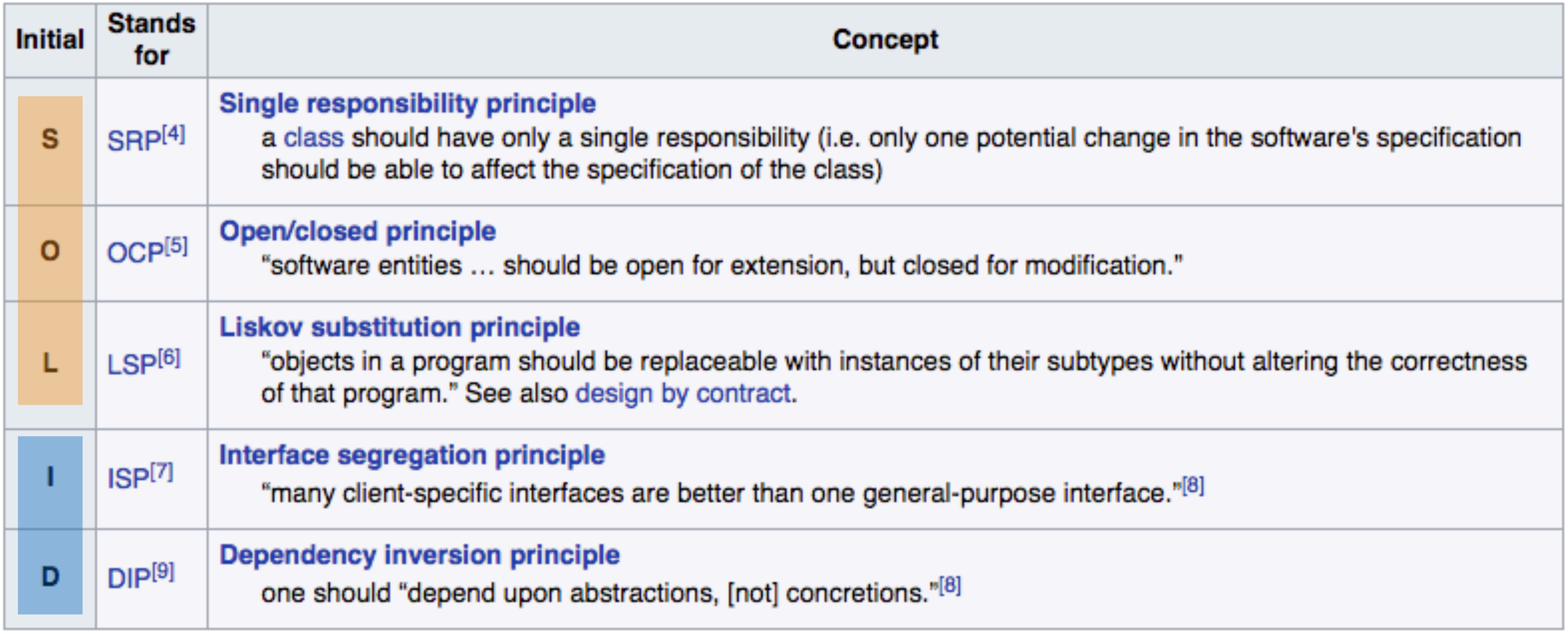
Provided Interface

Required Interface

http://www.agilemodeling.com/artifacts/componentDiagram.htm

I & D of the **SOLID** principles

3A

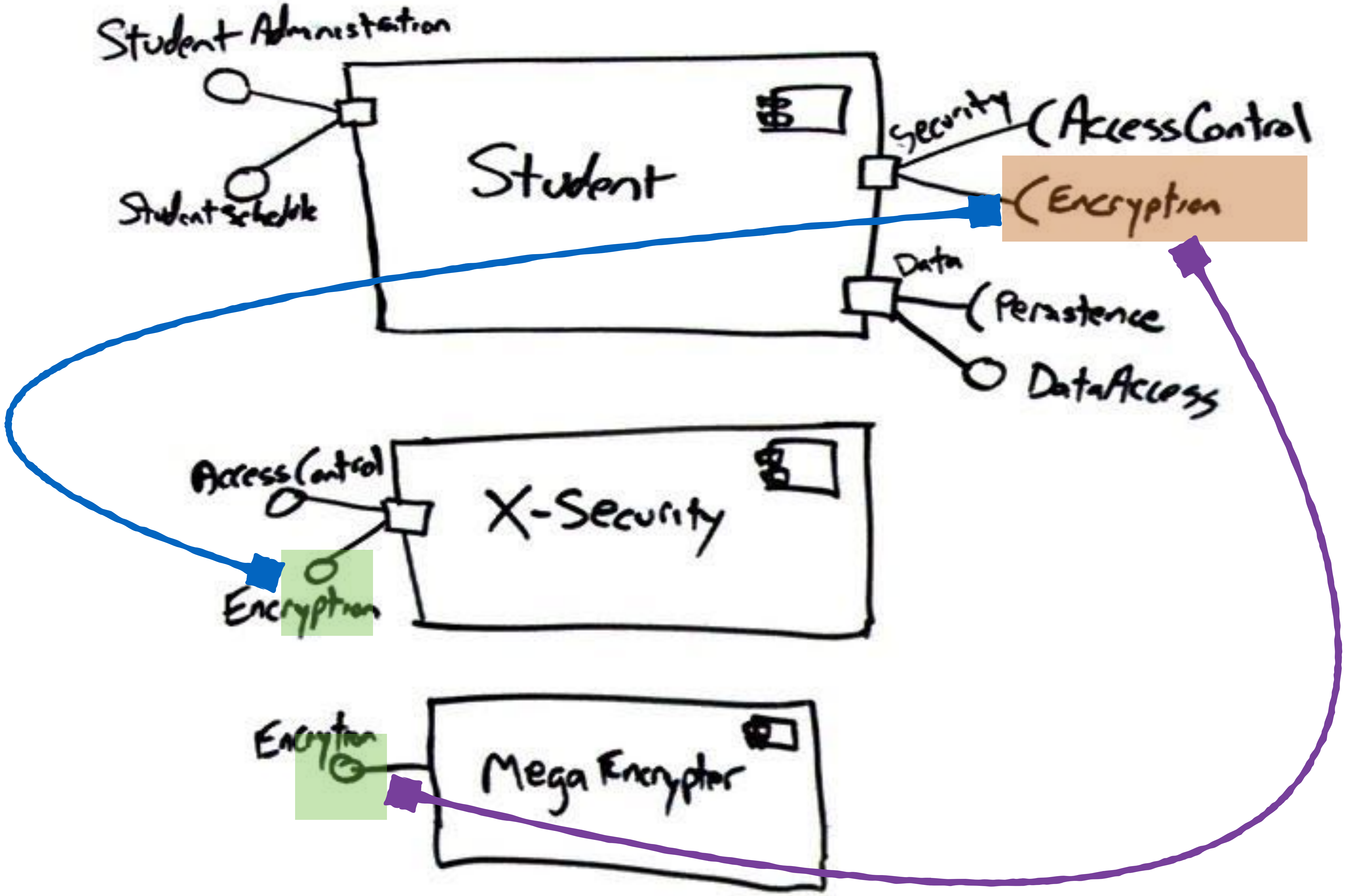


Initial	Stands for	Concept
S	SRP[4]	**Single responsibility principle** a class should have only a single responsibility (i.e. only one potential change in the software's specification should be able to affect the specification of the class)
O	OCP[5]	**Open/closed principle** "software entities … should be open for extension, but closed for modification."
L	LSP[6]	**Liskov substitution principle** "objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program." See also design by contract.
I	ISP[7]	**Interface segregation principle** "many client-specific interfaces are better than one general-purpose interface."[8]
D	DIP[9]	**Dependency inversion principle** one should "depend upon abstractions, [not] concretions."[8]

4A

https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)

Binding Components



http://www.agilemodeling.com/artifacts/componentDiagram.htm

Example of Implementation

```
class Student implements
      StudentAdministration, StudentSchedule {

  ~~Encryption e = new MegaEncrypter(…)~~

}
```

```
class Student implements
      StudentAdministration, StudentSchedule {

  AssemblyContext ctx = …
  Encryption e = ctx.inject(Encryption.class)

}
```

Teaser: Annotation-based injection

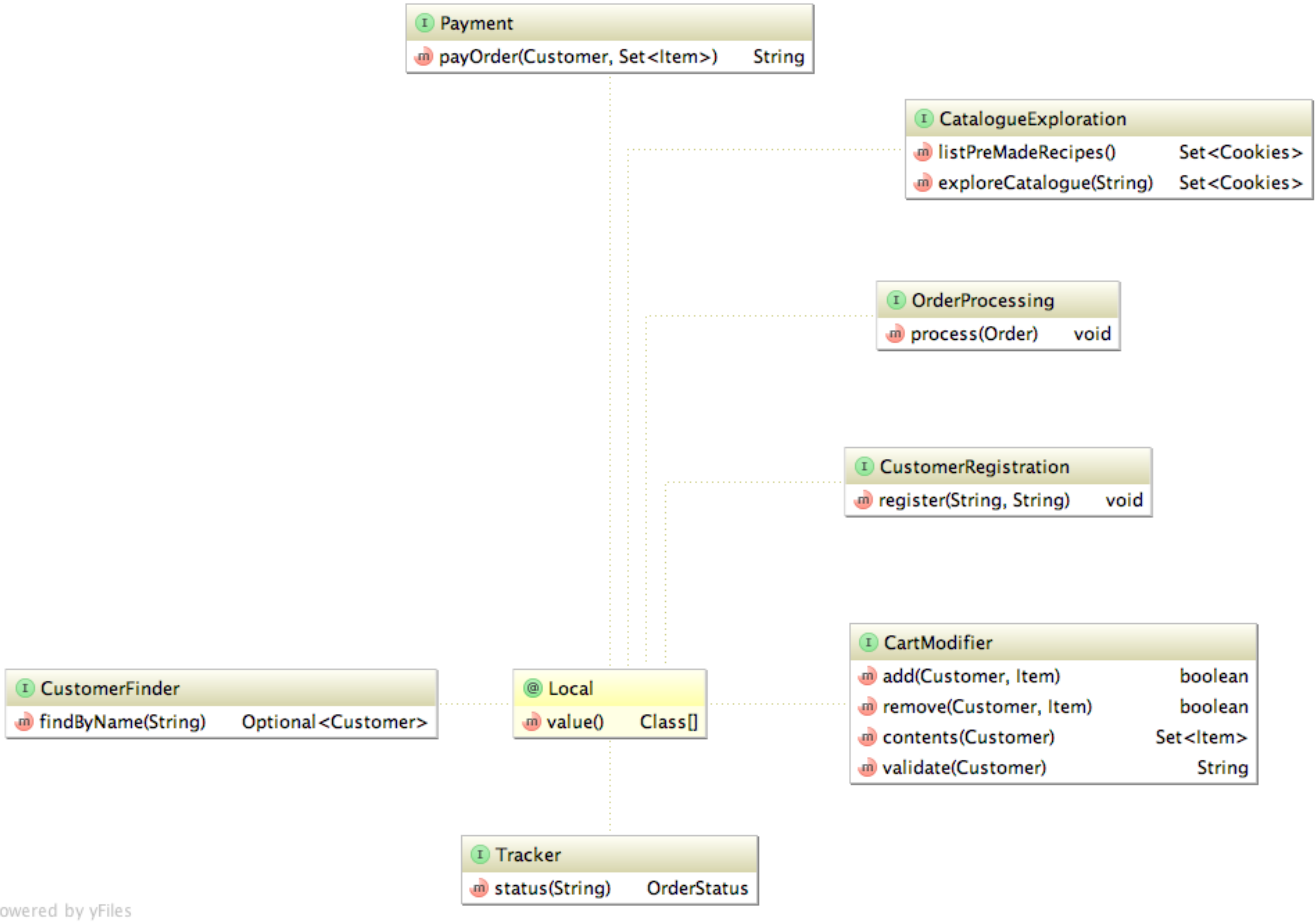
Provided Interface

```
class Student implements
      StudentAdministration, StudentSchedule {

  @Inject
  private Encryption e;

}
```

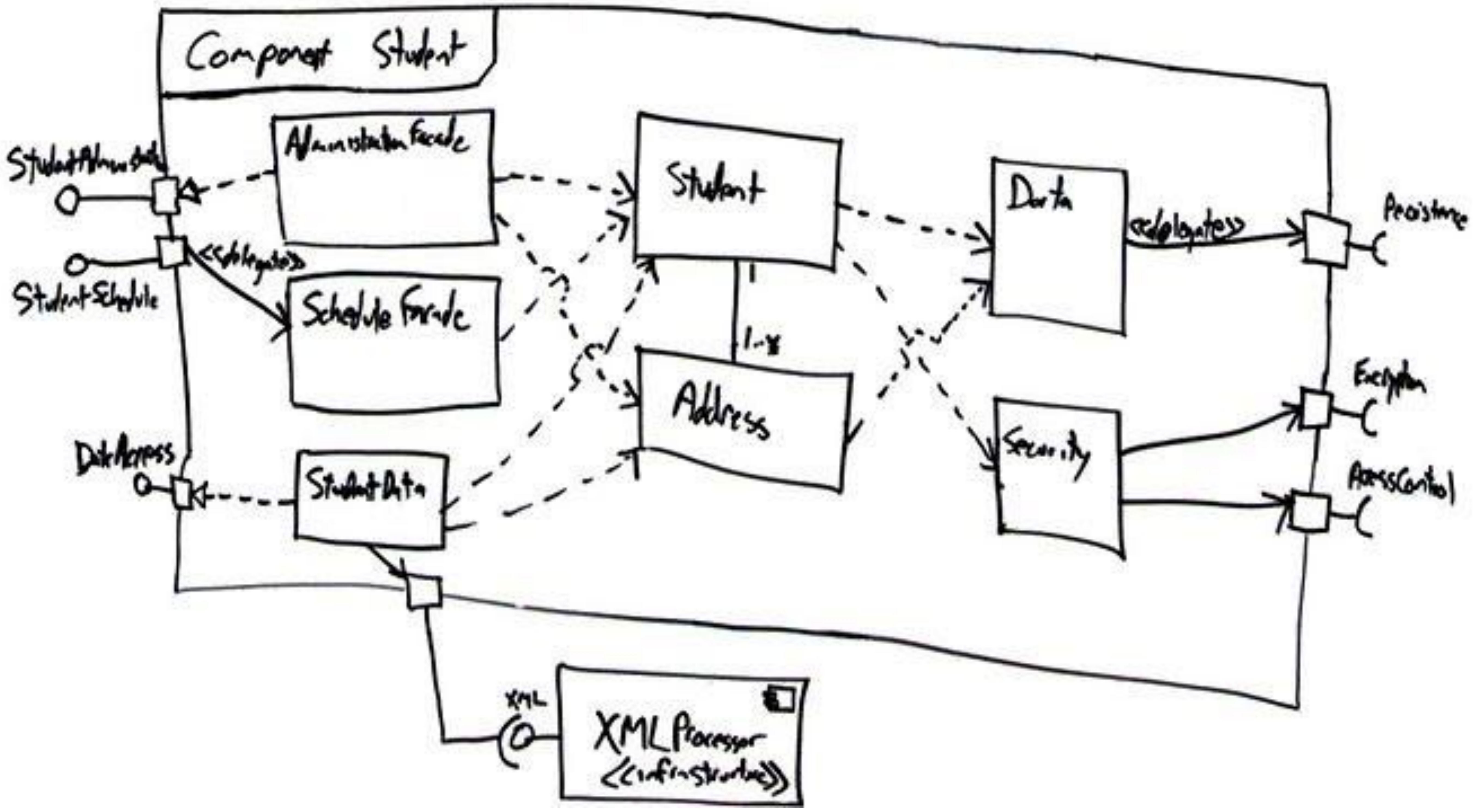
Required Interface

TCF MVP

Functional Interfaces for TCF ? Components ?


I Payment
m payOrder(Customer, Set<Item>) String
I CatalogueExploration
m listPreMadeRecipes() Set<Cookies>
m exploreCatalogue(String) Set<Cookies>
I OrderProcessing
m process(Order) void
I CustomerRegistration
m register(String, String) void
I CartModifier
m add(Customer, Item) boolean
m remove(Customer, Item) boolean
m contents(Customer) Set<Item>
m validate(Customer) String
I CustomerFinder
m findByName(String) Optional<Customer>
@ Local
m value() Class[]
I Tracker
m status(String) OrderStatus
owered by yFiles

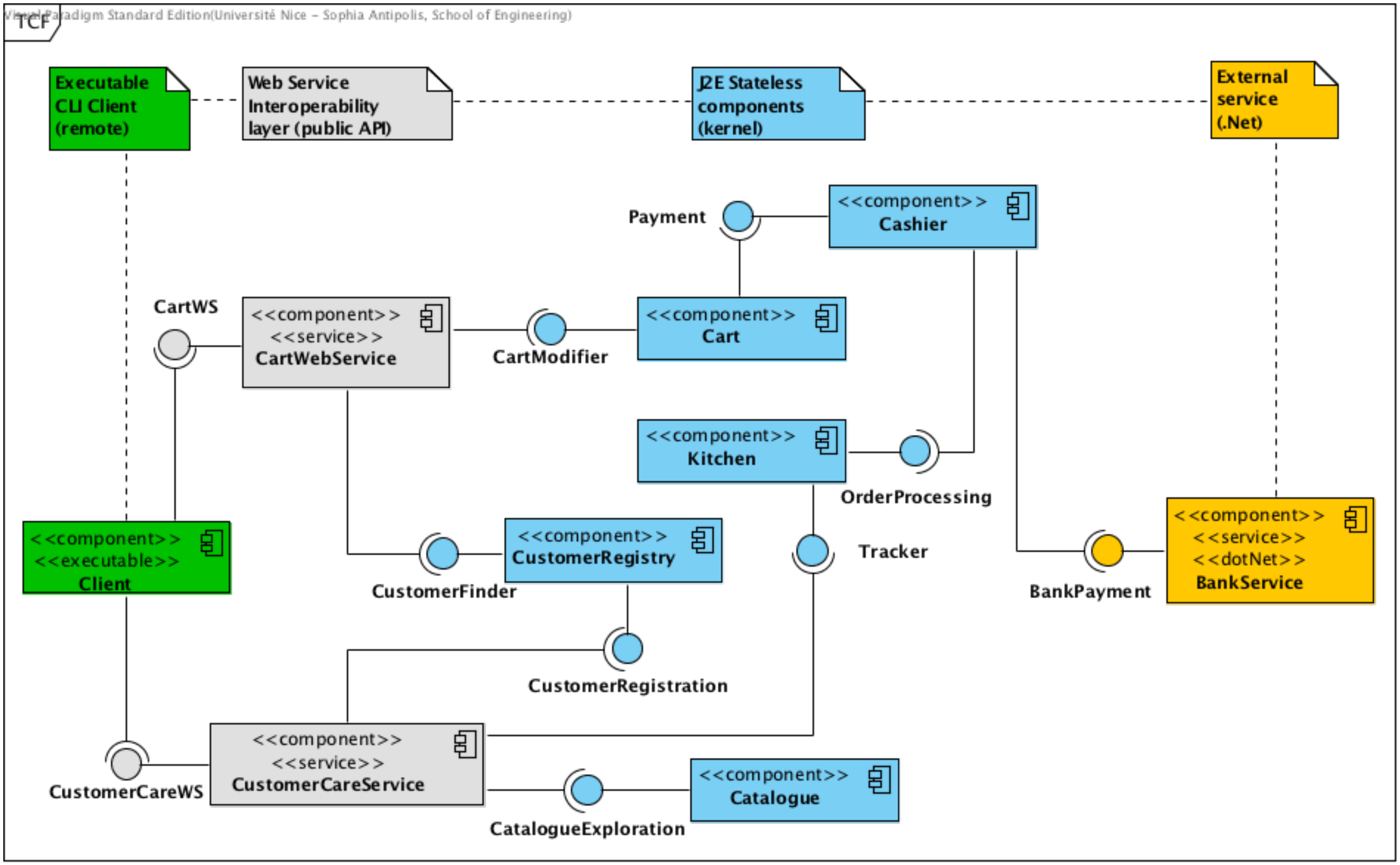
Components Assembly



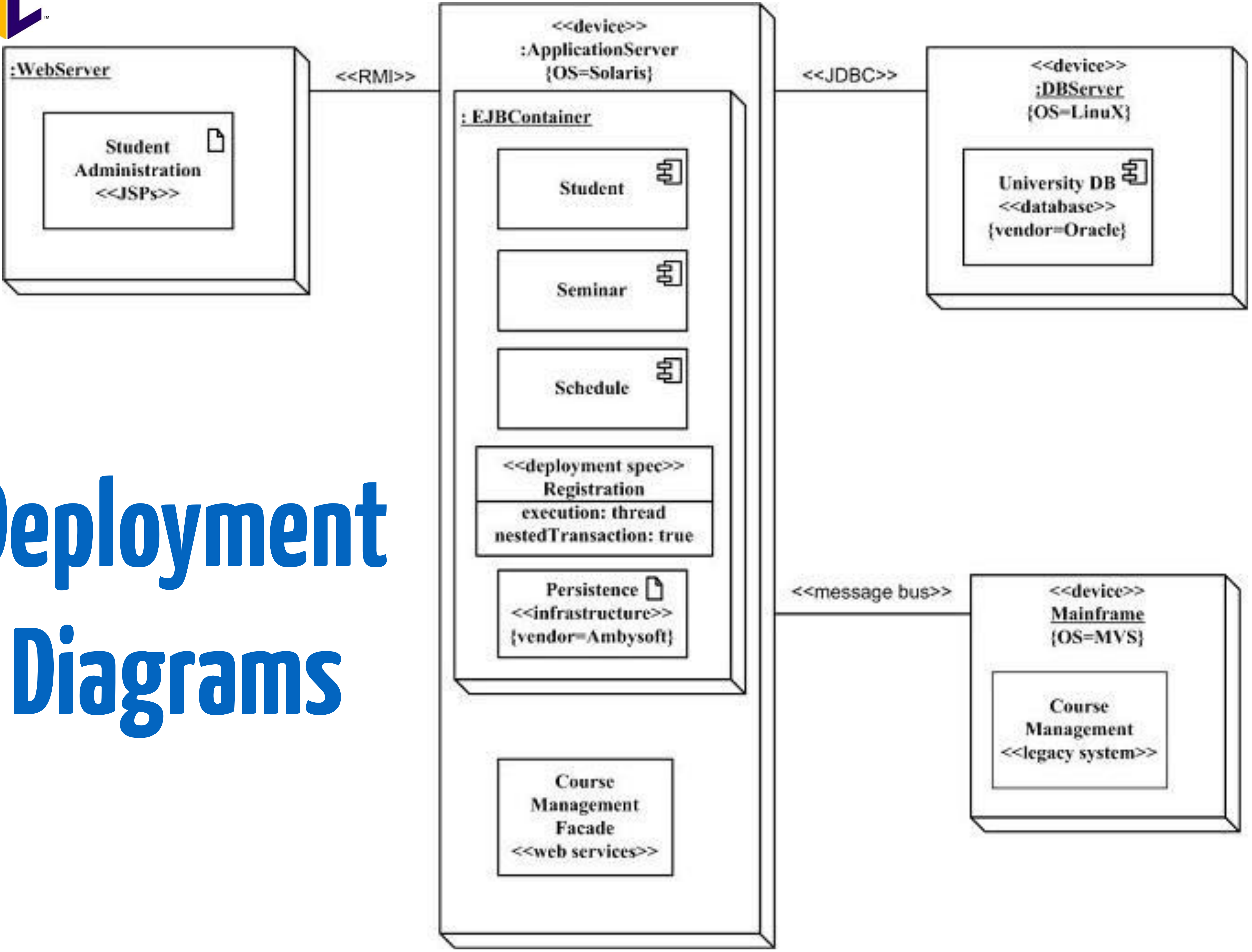
http://www.agilemodeling.com/artifacts/componentDiagram.htm

TCF MVP

Component Assembly ?


TCF
Visual Paradigm Standard Edition(Université Nice - Sophia Antipolis, School of Engineering)
Executable CLI Client (remote)
Web Service Interoperability layer (public API)
J2E Stateless components (kernel)
External service (.Net)
Payment
<<component>> Cashier
CartWS
<<component>> <<service>> CartWebService
CartModifier
<<component>> Cart
<<component>> Kitchen
OrderProcessing
<<component>> <<executable>> Client
<<component>> CustomerRegistry
CustomerFinder
Tracker
BankPayment
<<component>> <<service>> <<dotNet>> BankService
CustomerRegistration
CustomerCareWS
<<component>> <<service>> CustomerCareService
CatalogueExploration
<<component>> Catalogue

Deployment Diagrams



http://www.agilemodeling.com/artifacts/deploymentDiagram.htm

TCF MVP

How to deploy TCF ?