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

Philippe Collet - with 90% Slides from Sébastien Mosser

Lecture #2

pictures: sxc.hu

POLYTECH NICE SOPHIA | UNIVERSITÉ CÔTE D'AZUR

CC BY NC

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 Clémentine Némo for the reference

Software Architecture **Concerns**

Producibility

Functionality

Changeability

Security

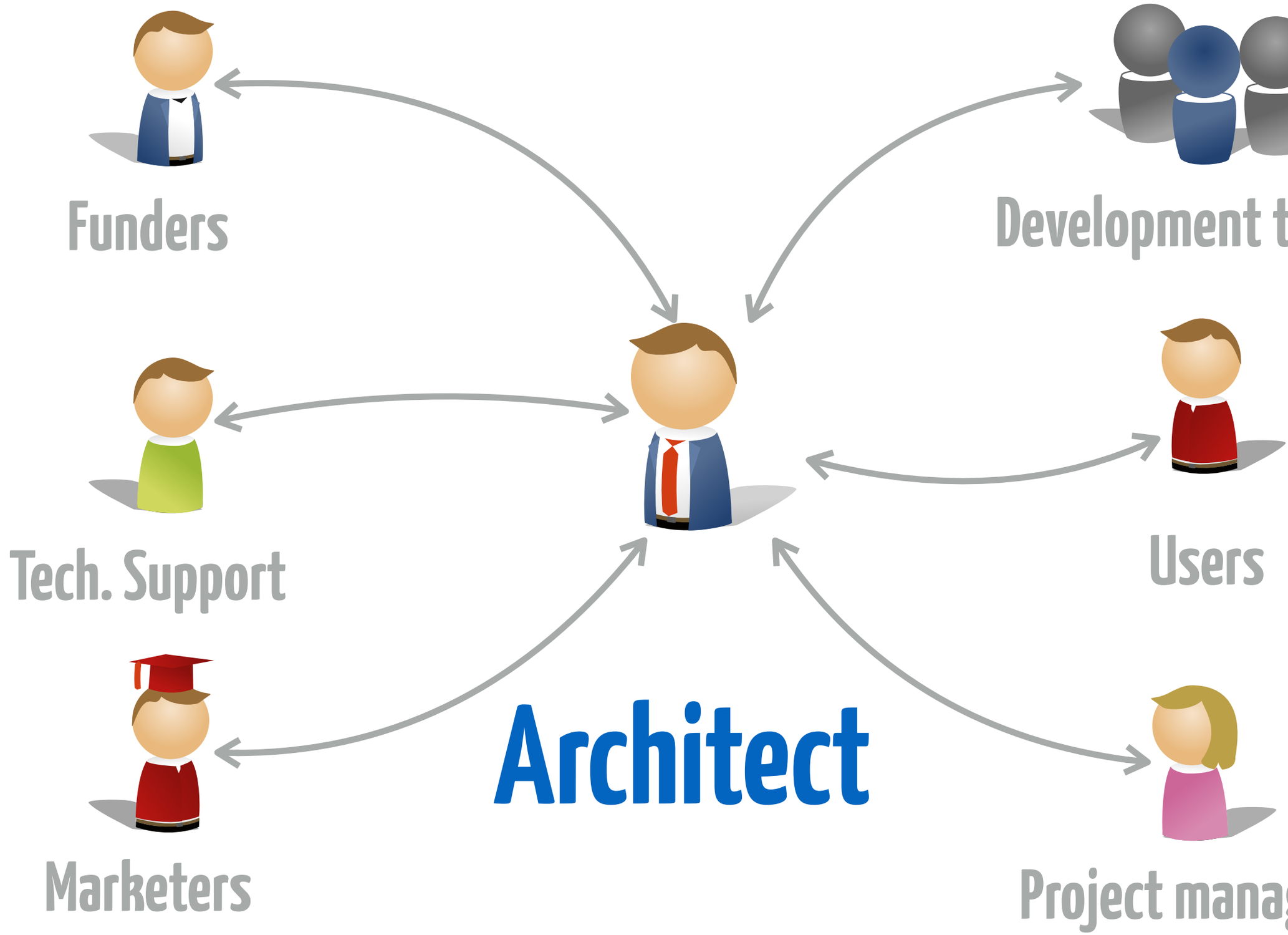
Modularity

Performance

Ecosystem

Buildability

The Architect **Ecosystem**



Expect the unexpected!

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

1 Architectural rule of thumb

2 Layered Architectures

3 Support from the UML

Architectural rule of thumb

[Beautiful Architecture]

“

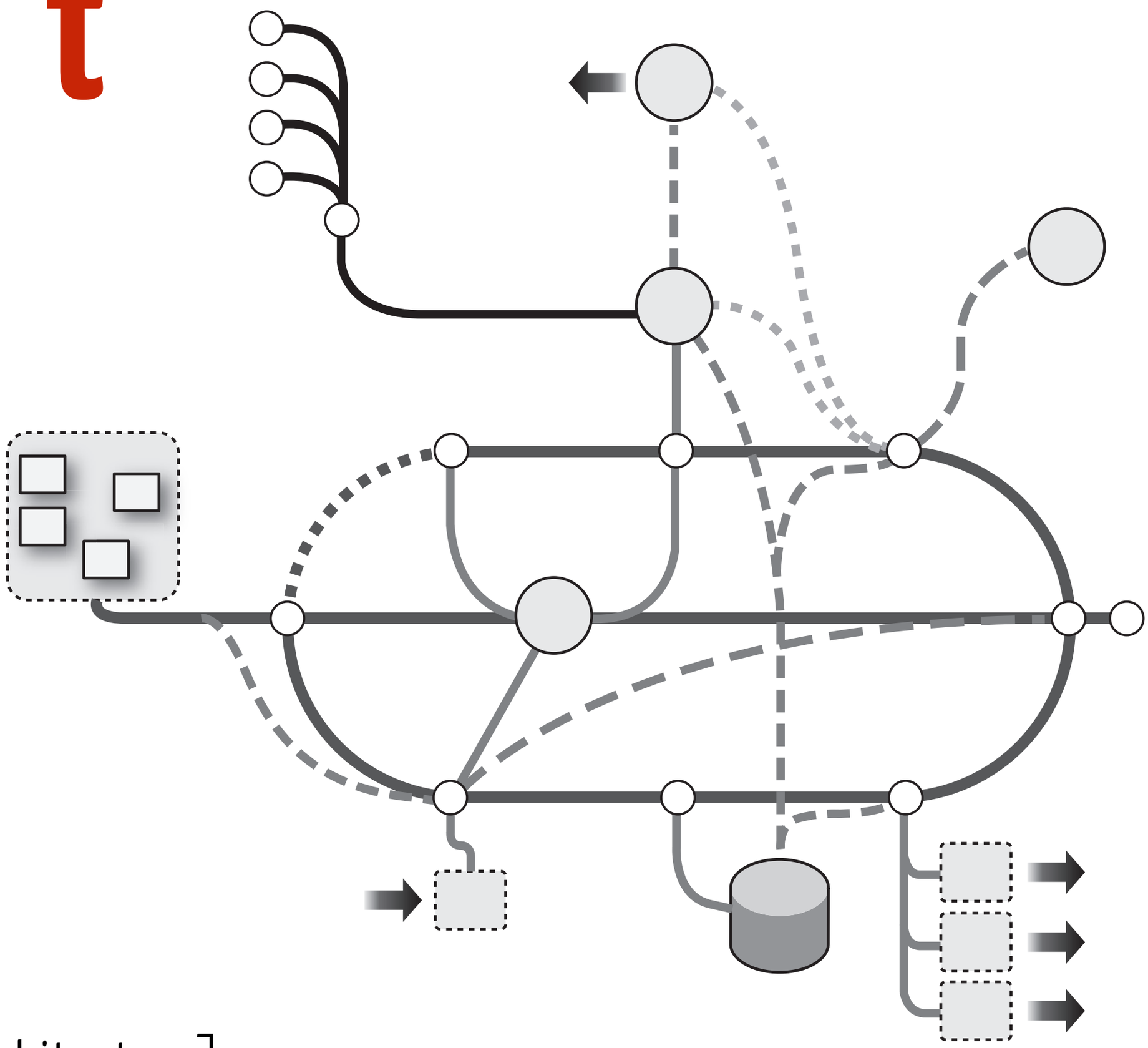
Software **architecture** is

not set in stone.

Change if you need it.

[Beautiful Architecture]

Don’t



[Beautiful Architecture]

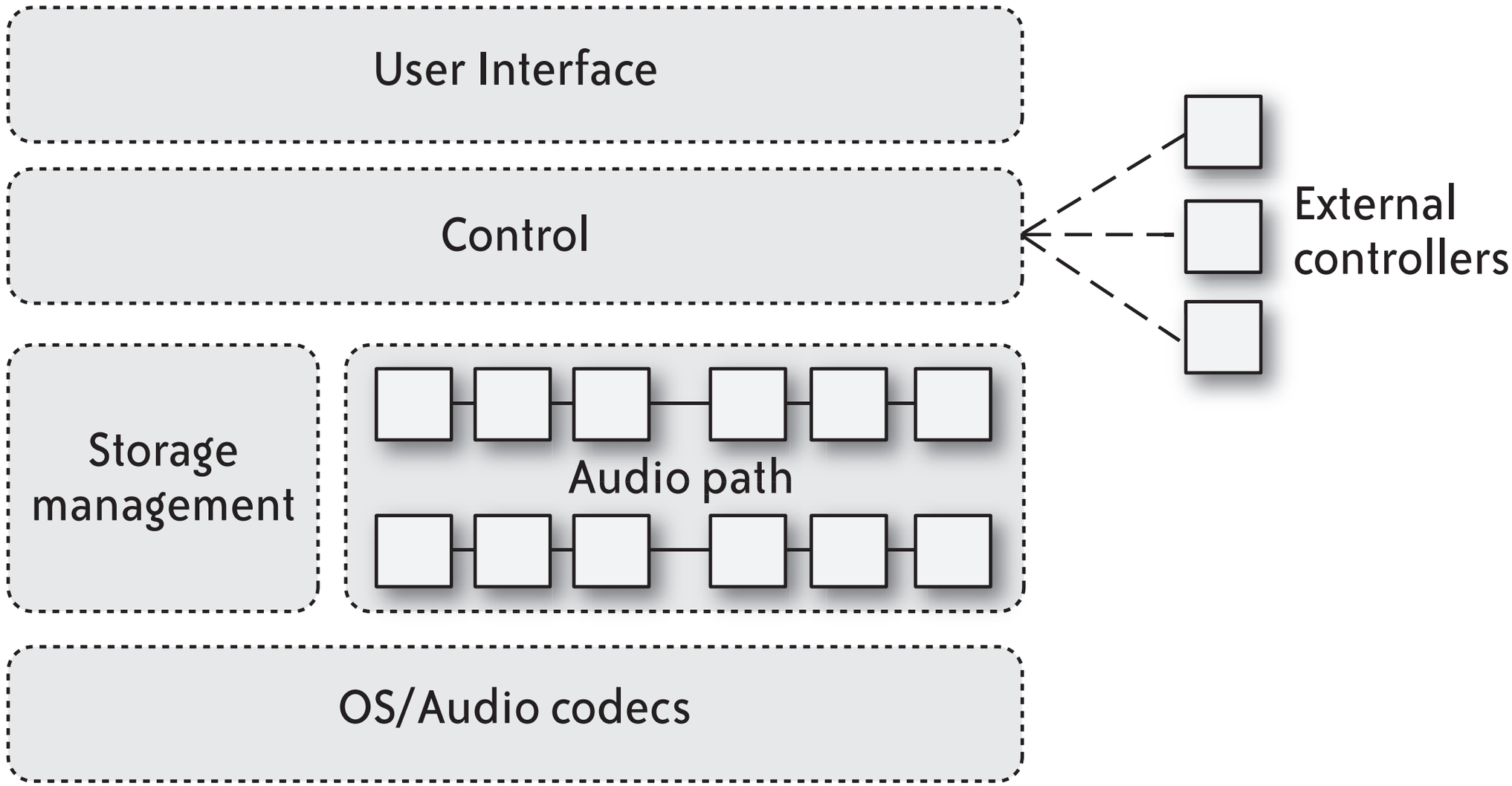
“A **fuzzy architecture** leads to **individual** code, **duplication** of code and effort.

[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]

TCF : The Cookie Factory 2

https://github.com/collet/4A_ISA_TheCookieFactory

Chapters

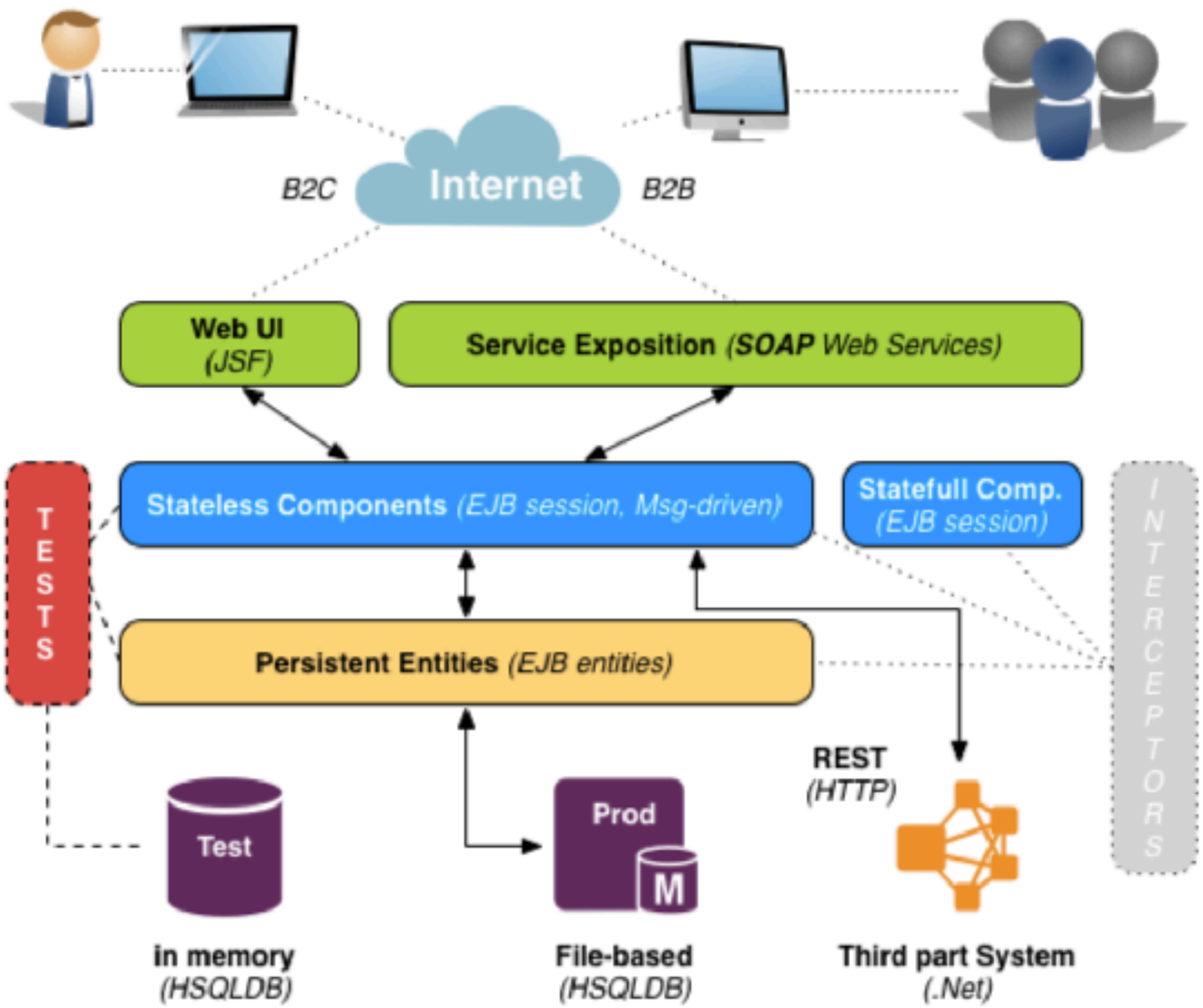
1. Architecture
2. Business components with EJB Sessions
3. Exposing components as Web Services (SOAP)
4. Consuming external Web Services (REST)
5. Unit testing *versus* Integration testing
6. Complete architecture overview
7. Message interceptors to support the NTUI (*Never Trust User Input*) golden rule
8. Making things persistent
9. Web user interface using JSF
10. Asynchronous Communication using Messages
11. Conclusions

https://github.com/collet/4A_ISA_TheCookieFactory

Product vision

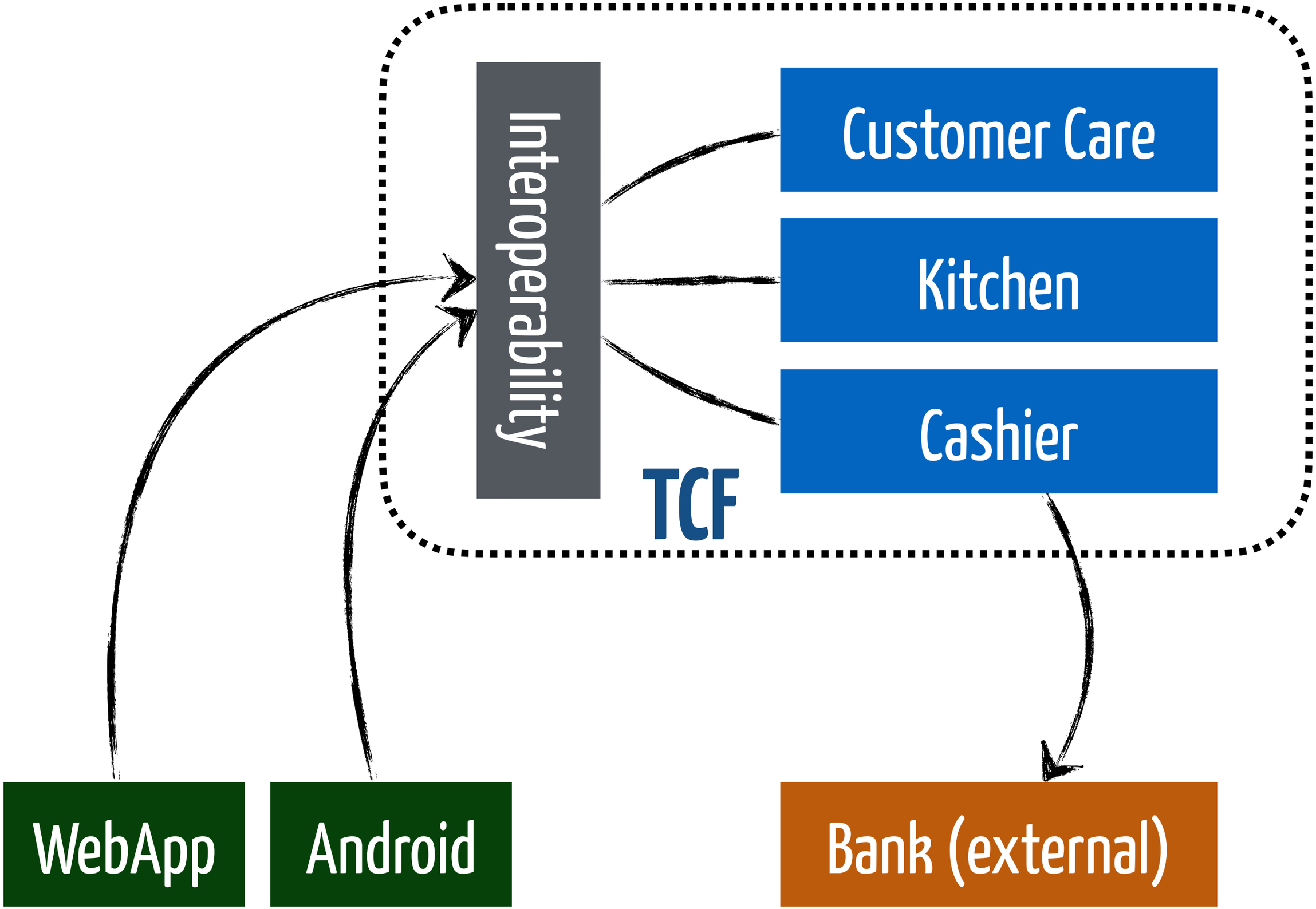
The Cookie Factory (TCF) is a major bakery brand in the USA. The *Cookie on Demand* (CoD) system is an innovative service offered by TCF to its customer. They can order cookies online thanks to an application, and select when they'll pick-up their order in a given shop. The CoD system ensures to TCF's happy customers that they'll always retrieve their pre-paid warm cookies on time.

The software architecture to be developed in this document will rely on the following stack:

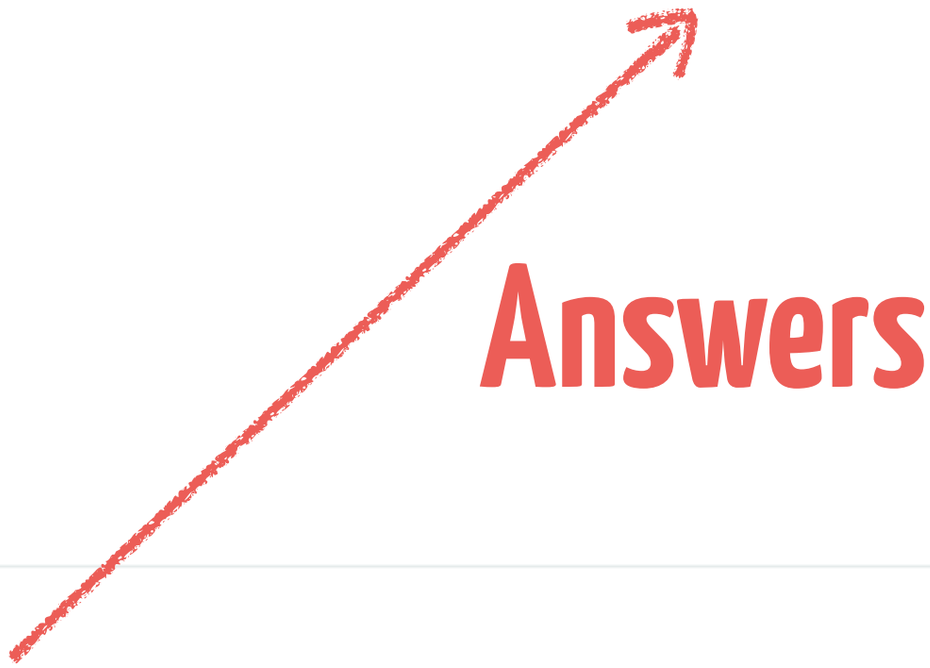


What are the “modules” in TCF?

How are they related to each others?


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

https://github.com/collet/4A_ISA_TheCookieFactory/blob/develop/chapters/Architecture.md

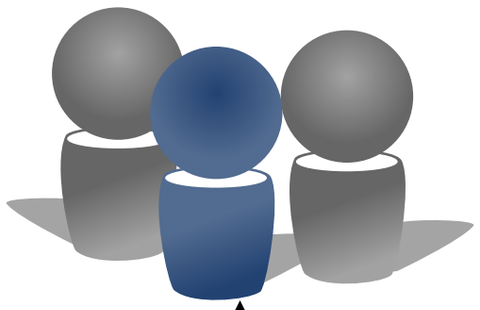


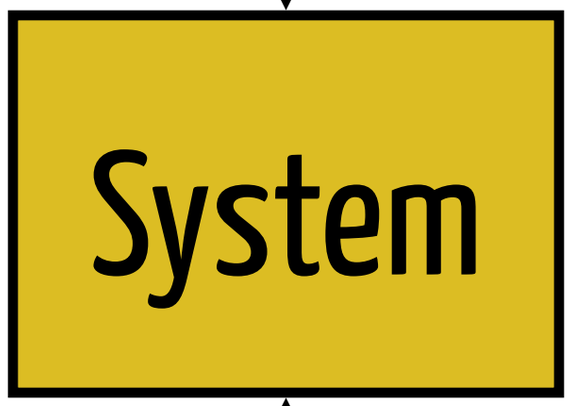
Chapters

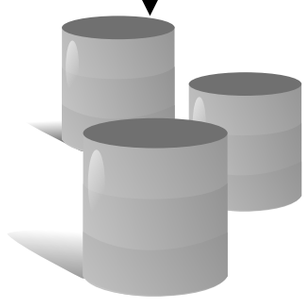
1. Architecture
2. Business components with EJB Sessions
3. Exposing components as Web Services (SOAP)
4. Consuming external Web Services (REST)
5. Unit testing *versus* Integration testing
6. Complete architecture overview
7. Message interceptors to support the NTUI (*Never Trust User Input*) golden rule
8. Making things persistent
9. Web user interface using JSF
10. Asynchronous Communication using Messages
11. Conclusions

Layered Architectures

2




System

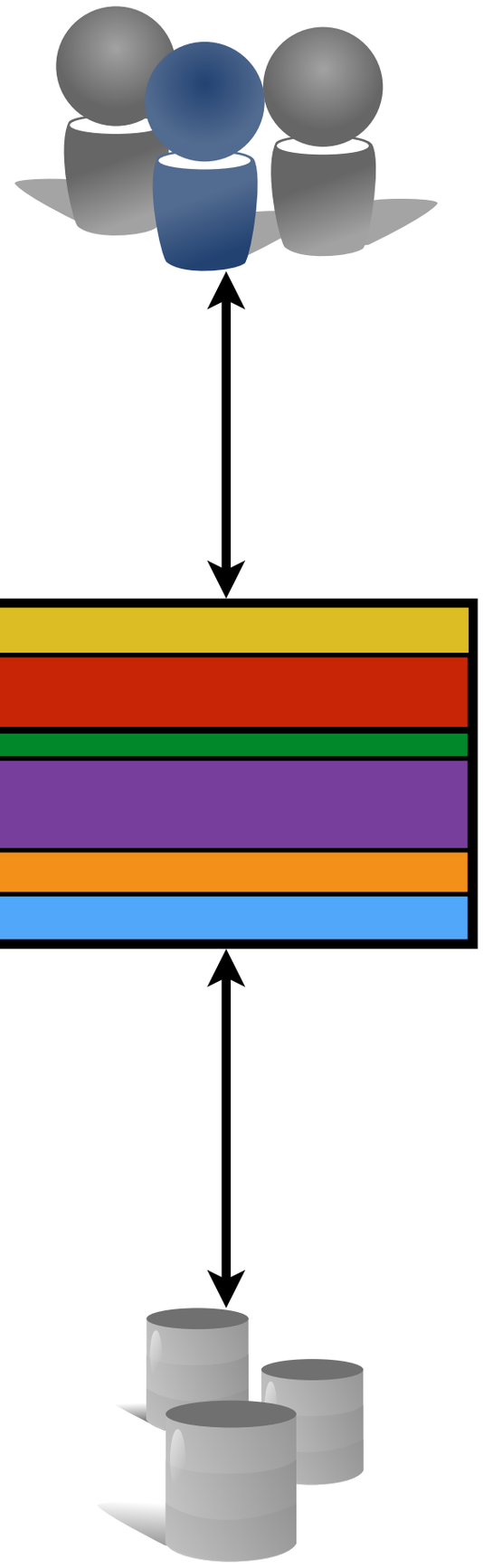


???

Layers

support

modularity



“Ma-gni-fique”

thanks to Clèm for the reference

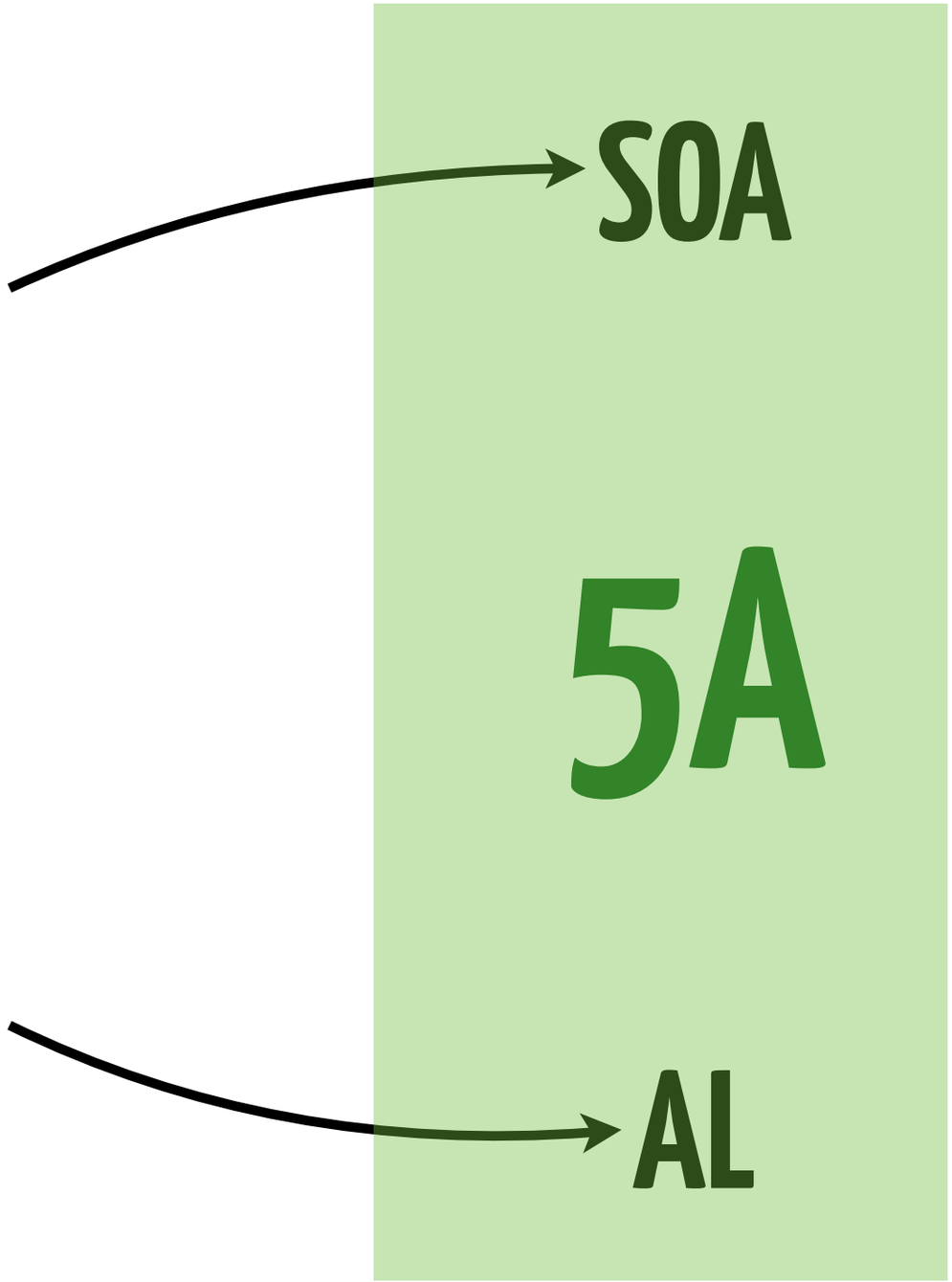
Keep

I t

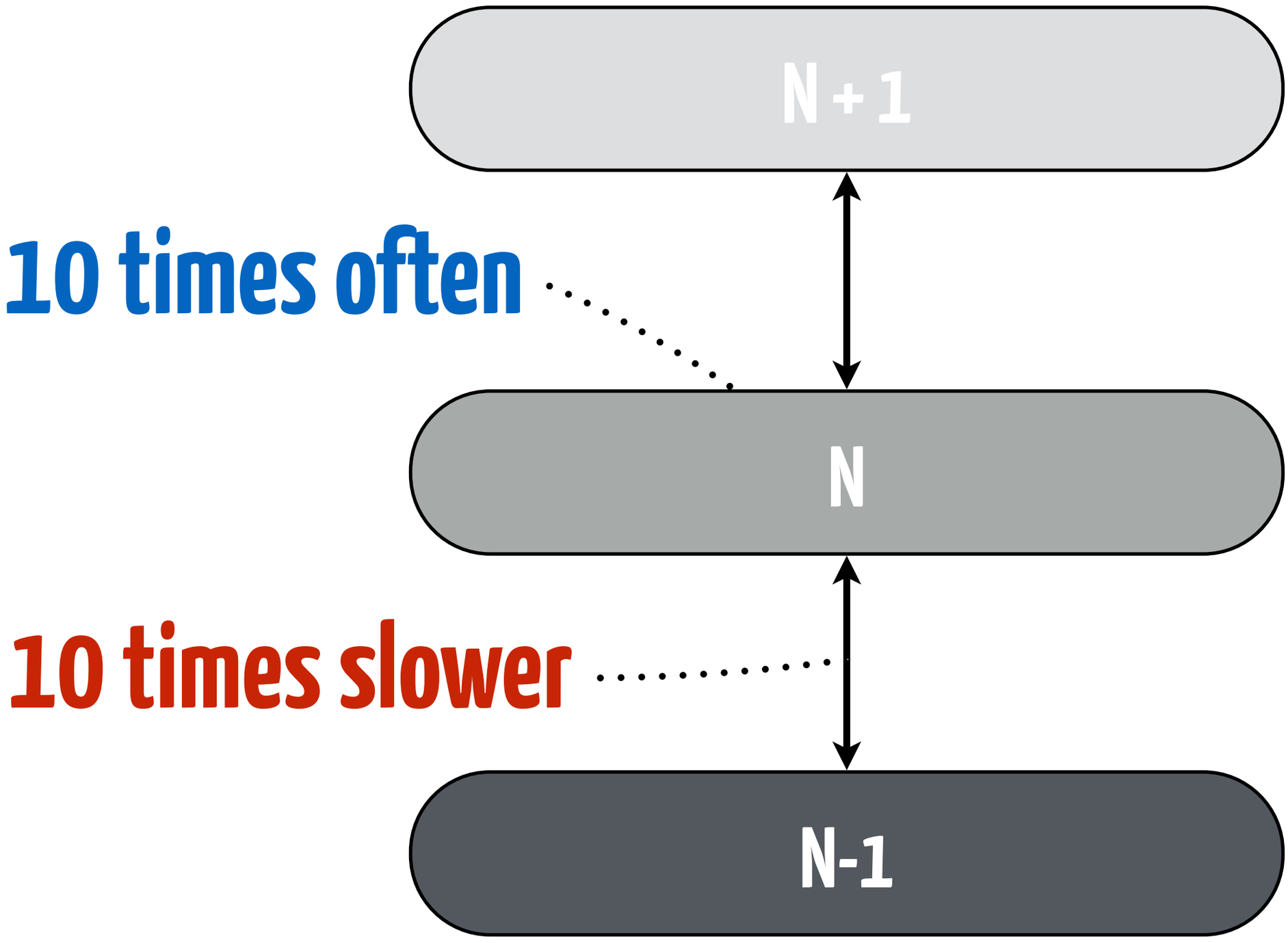
4A

S tupid,

S imple.



The rule of 10



Theory & Practice

Theory:

you know everything but nothing works

Practice:

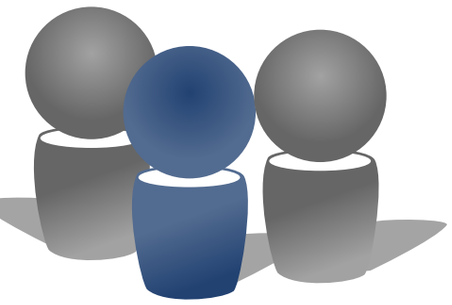
It works, but no one knows why

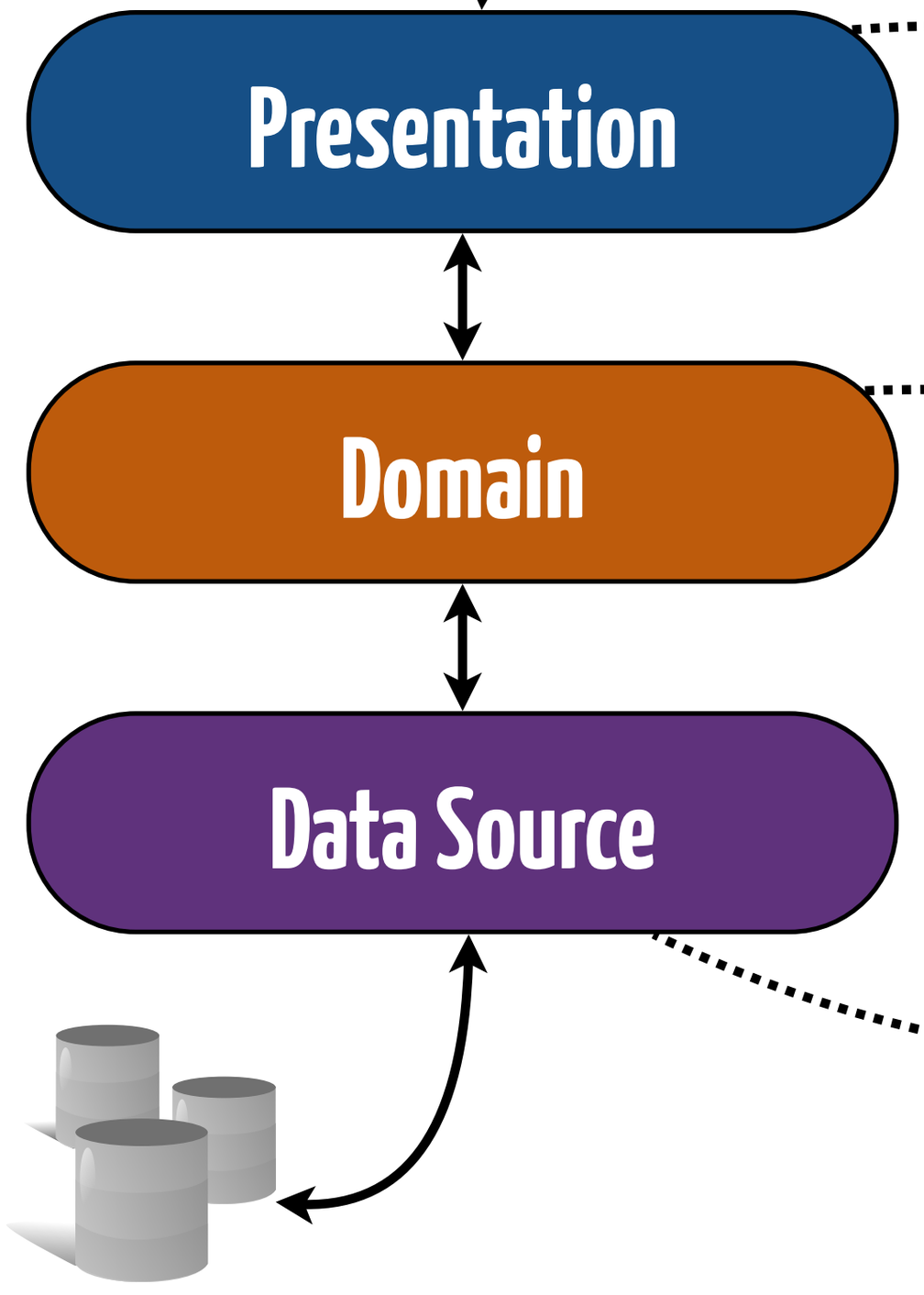
In theory, **N-tiers** architecture

In practice, N = 3

(or 5)

3-tiers architecture

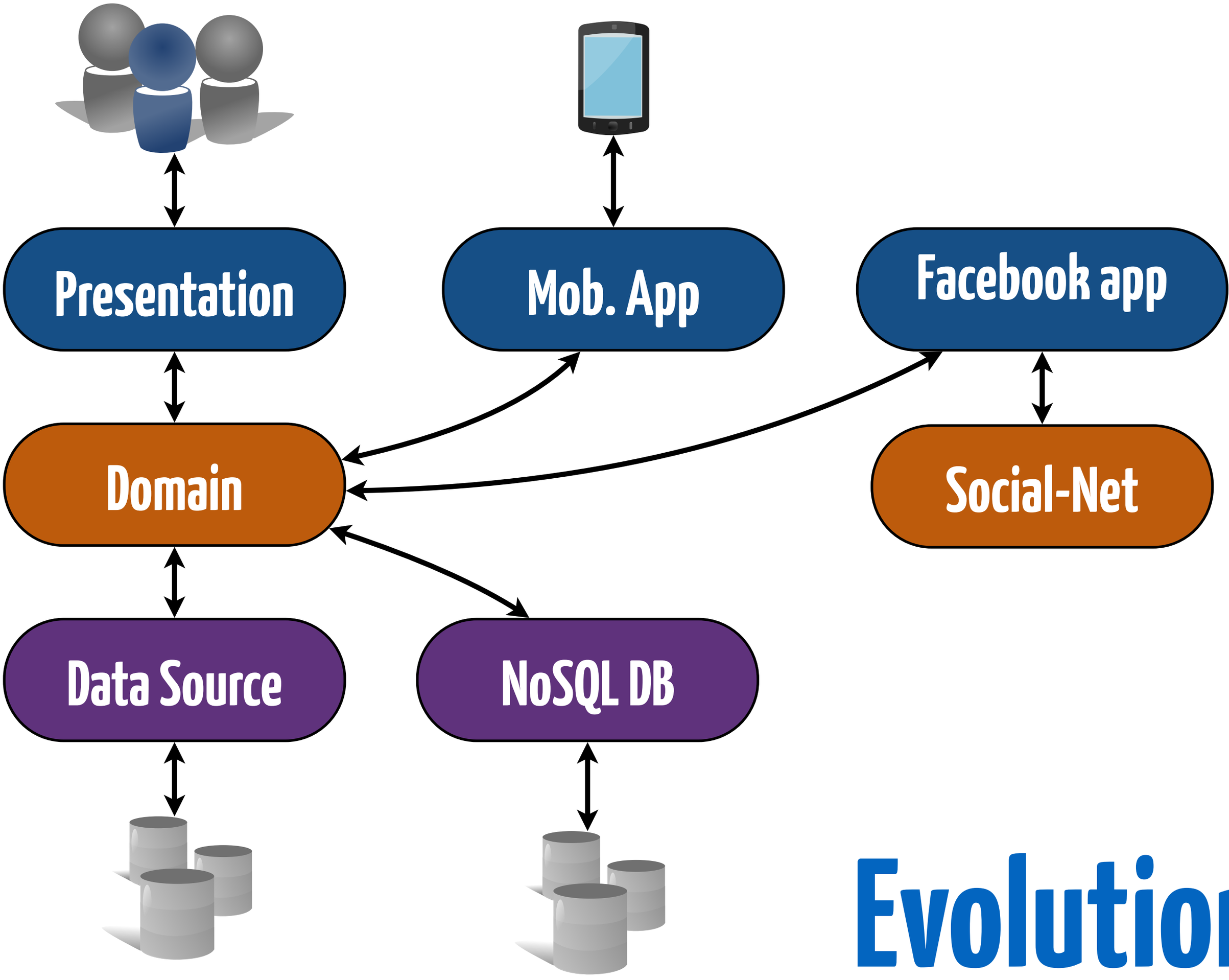




Handle users

Business functionality

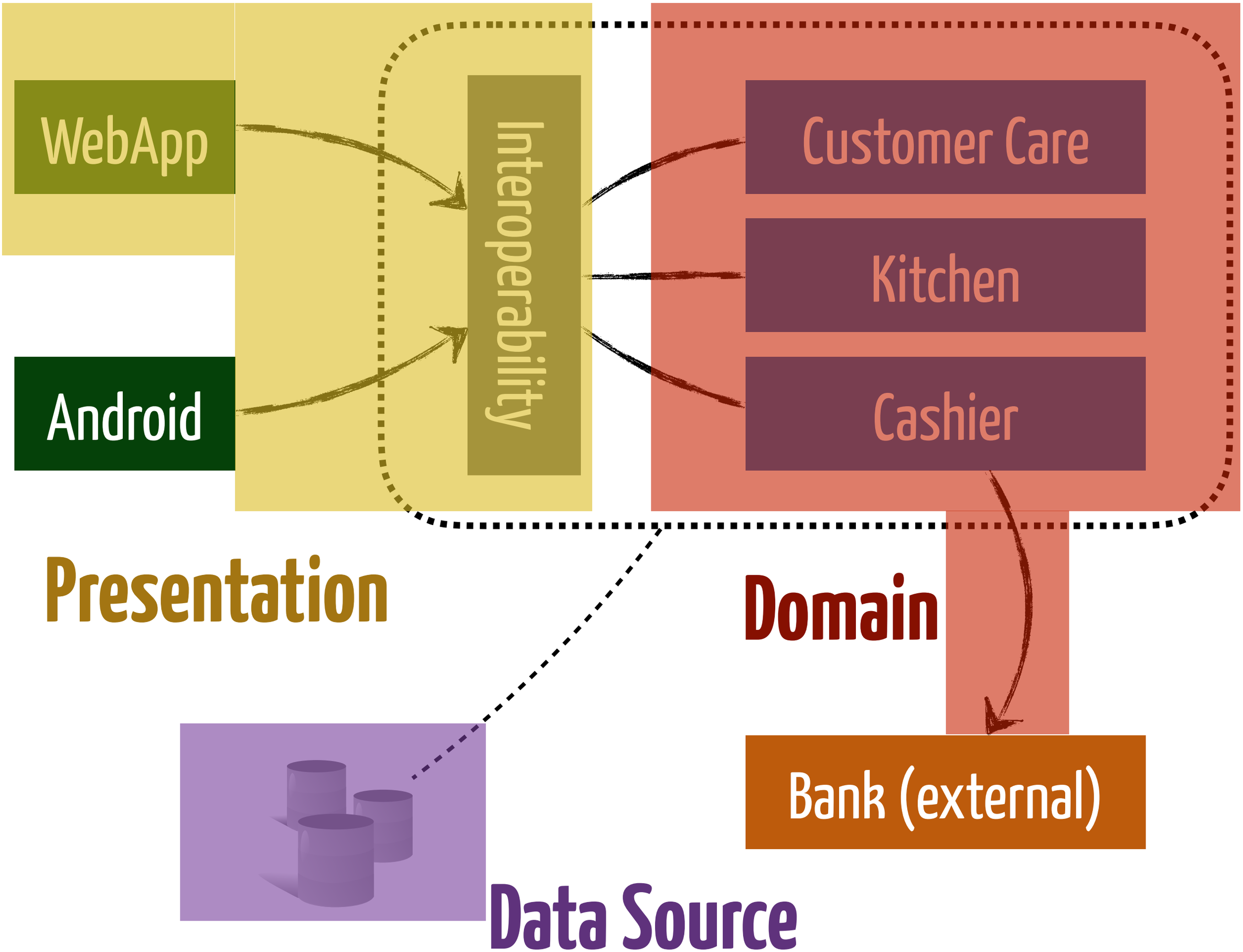
Handle data storage


Presentation
Mob. App
Facebook app
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

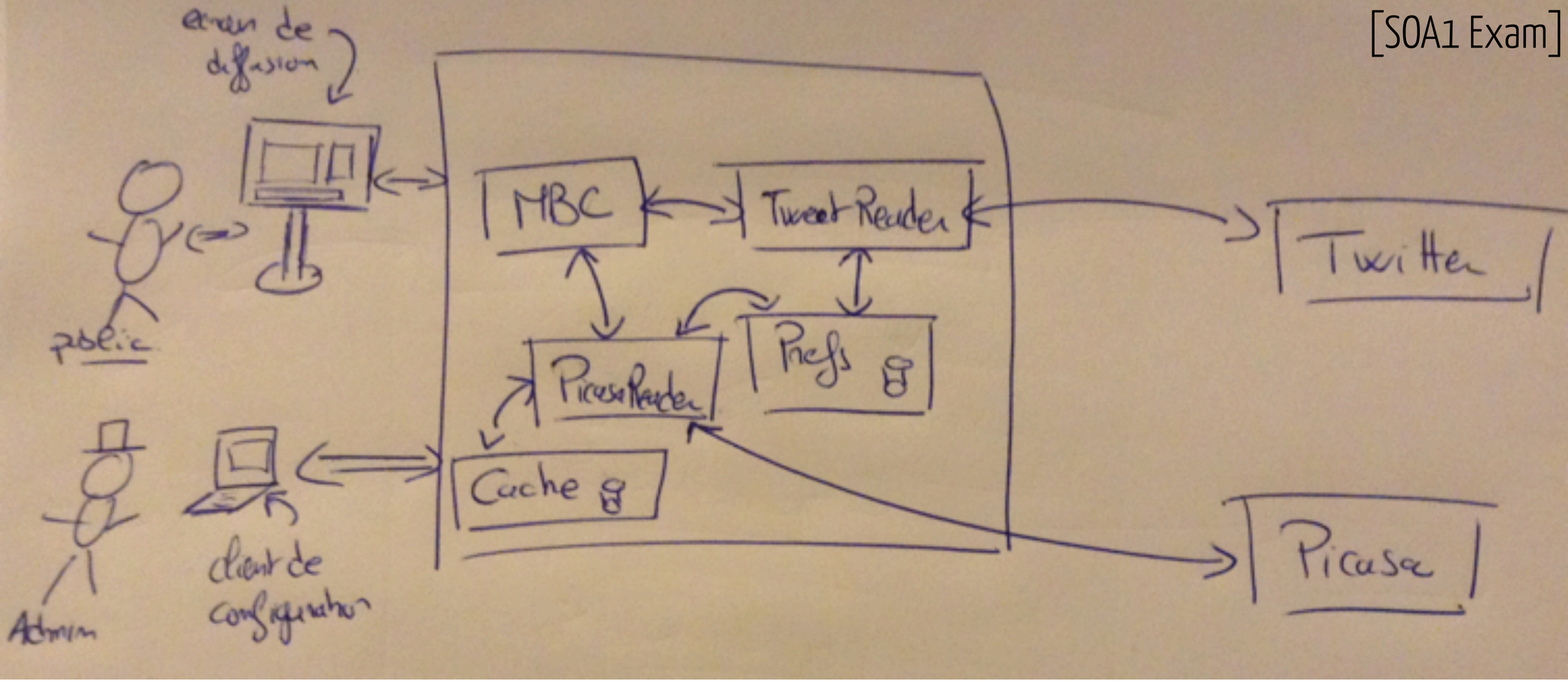
UNIFIED MODELING LANGUAGE™

The **UML** is just a

standard syntax

for modeling

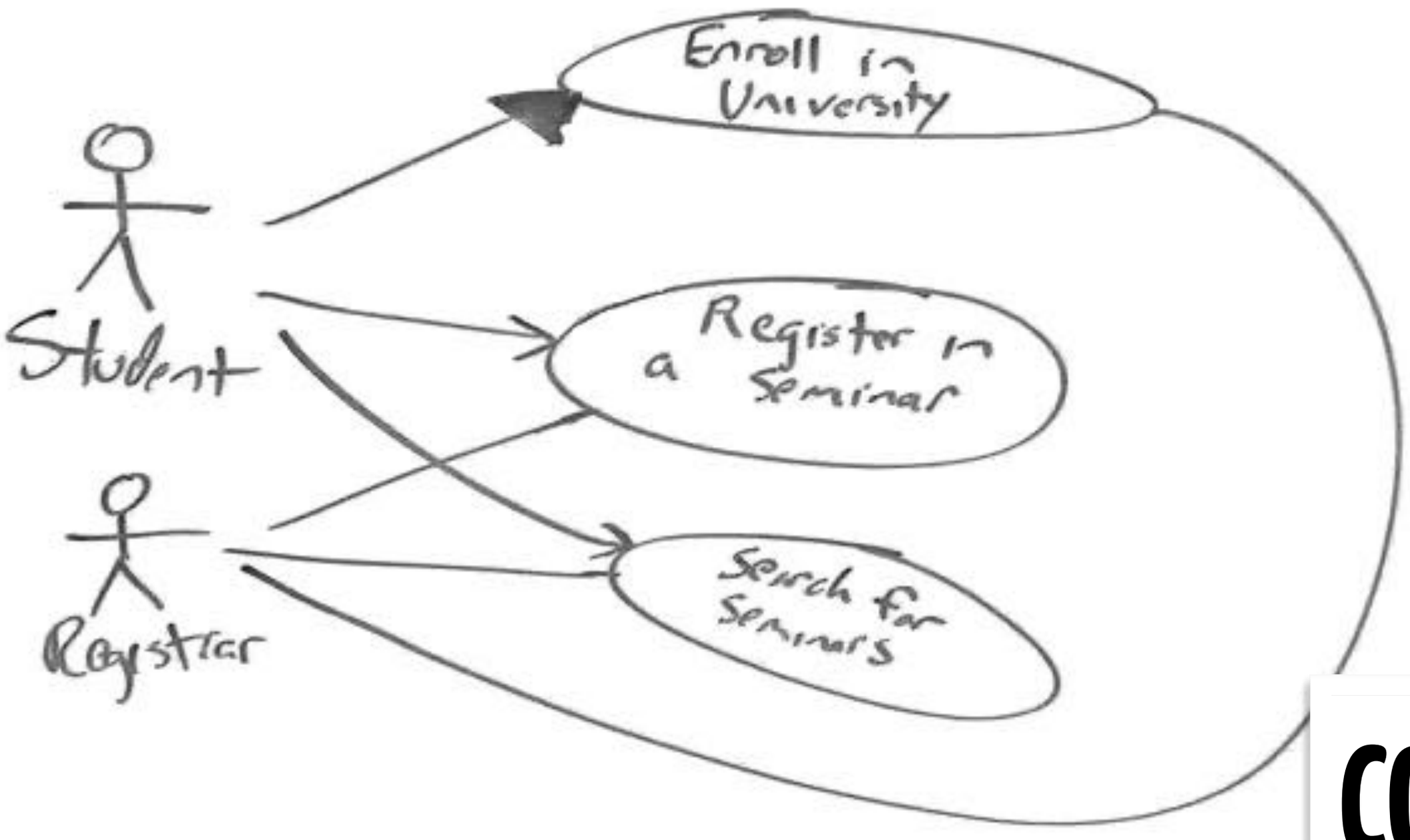
[SOA1 Exam]



One can design a Software Architecture

without the UML

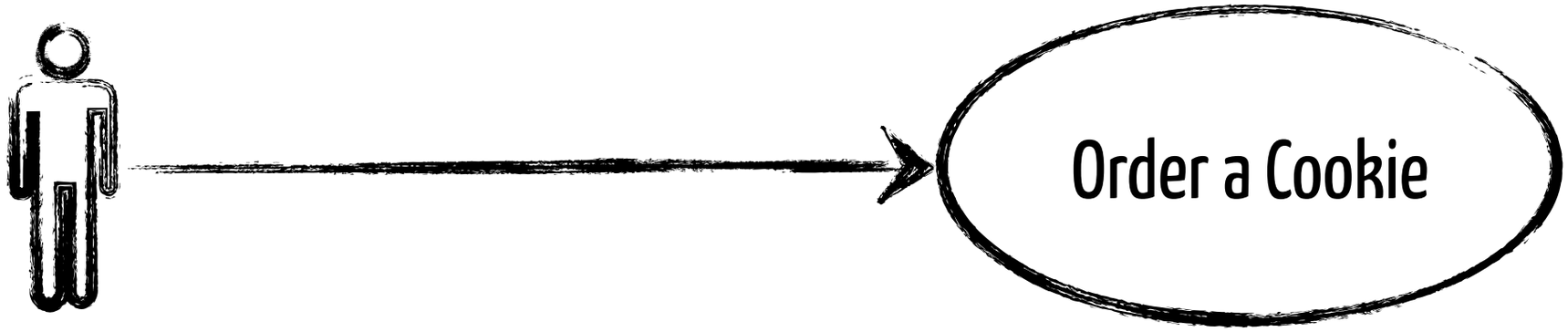
Use case diagrams



COO

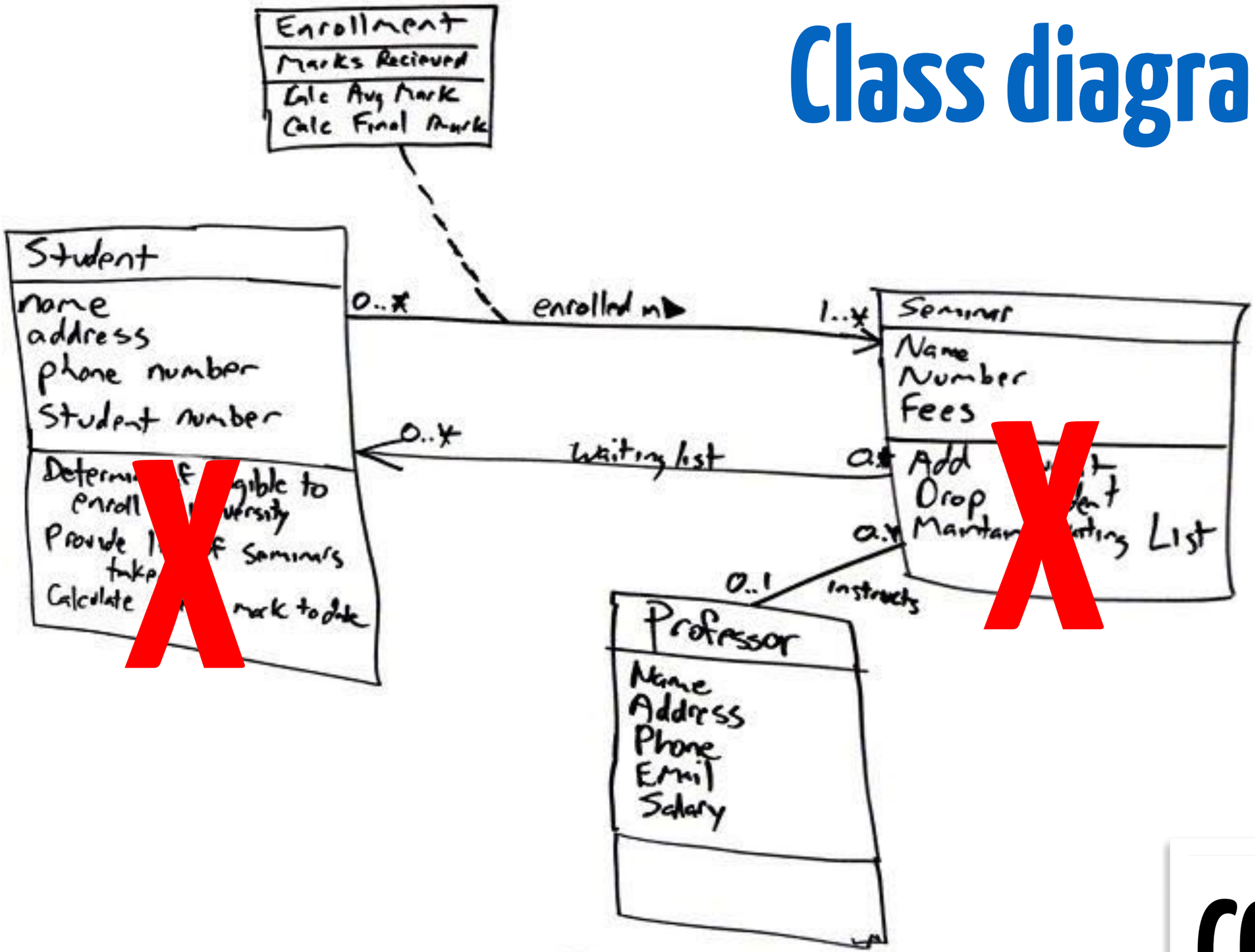
TCF MVP

Minimal & Viable Use Case ?



customer

Class diagrams



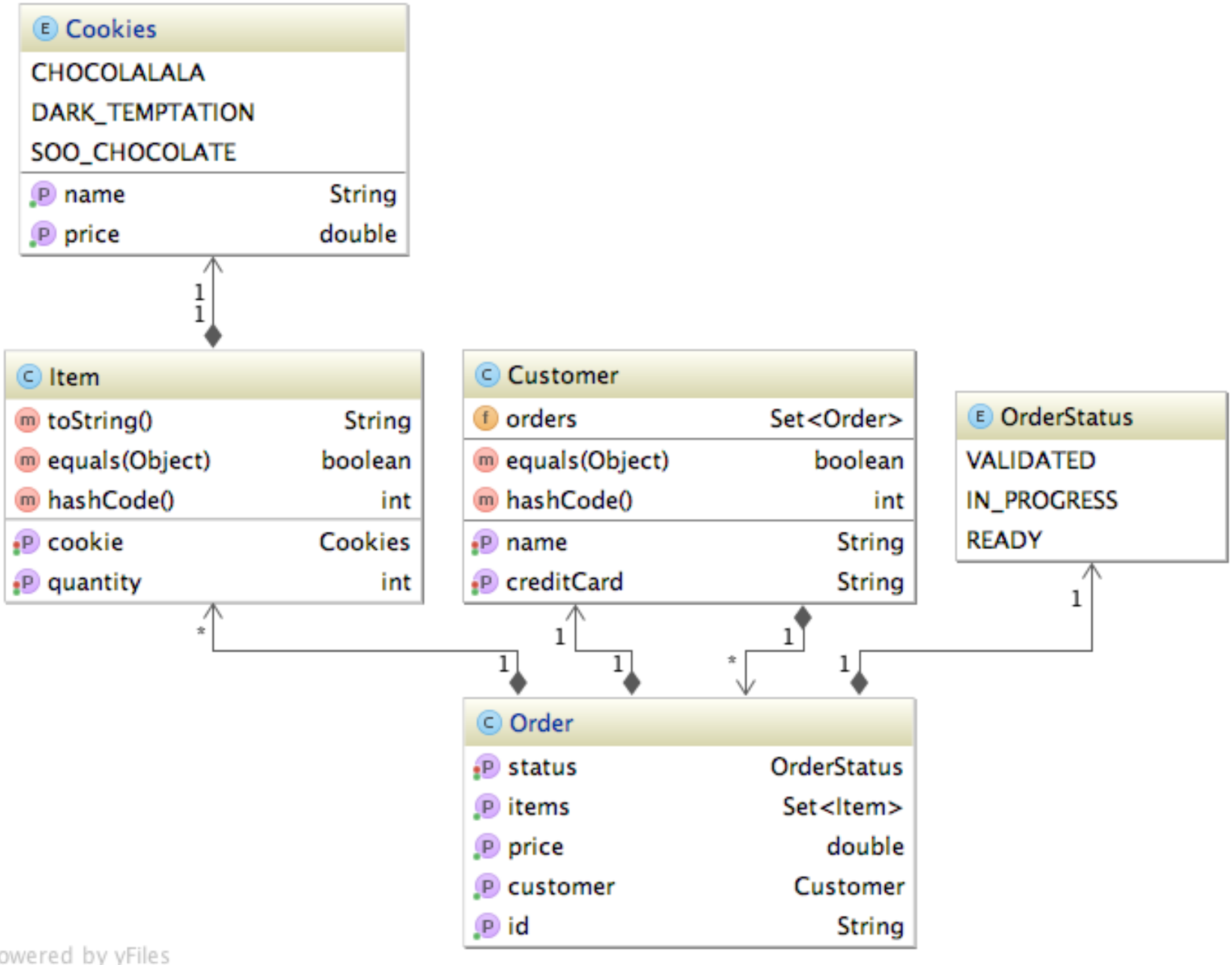
Business objects => **no methods**

COO

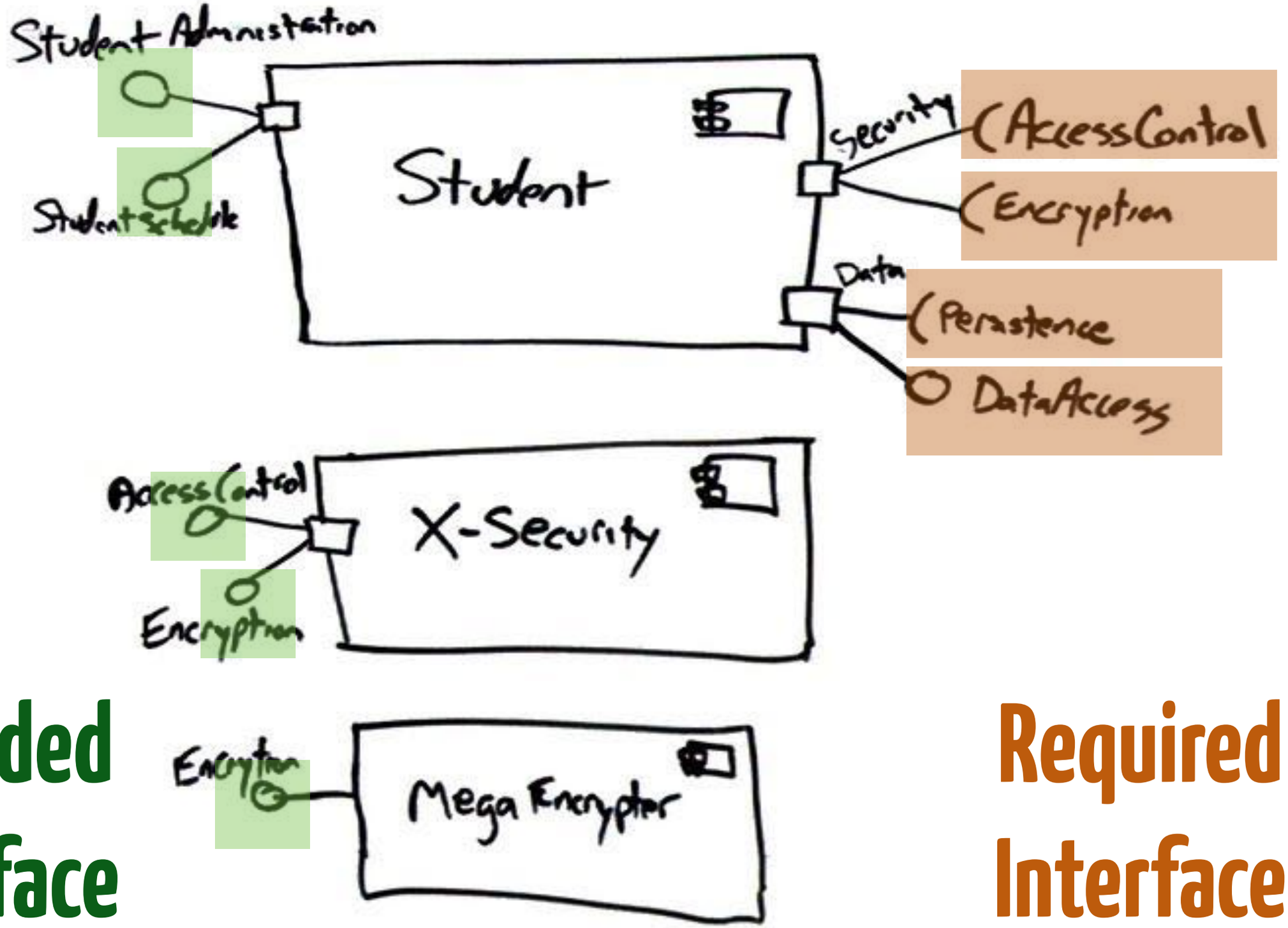
http://www.agilemodeling.com/artifacts/classDiagram.htm

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
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

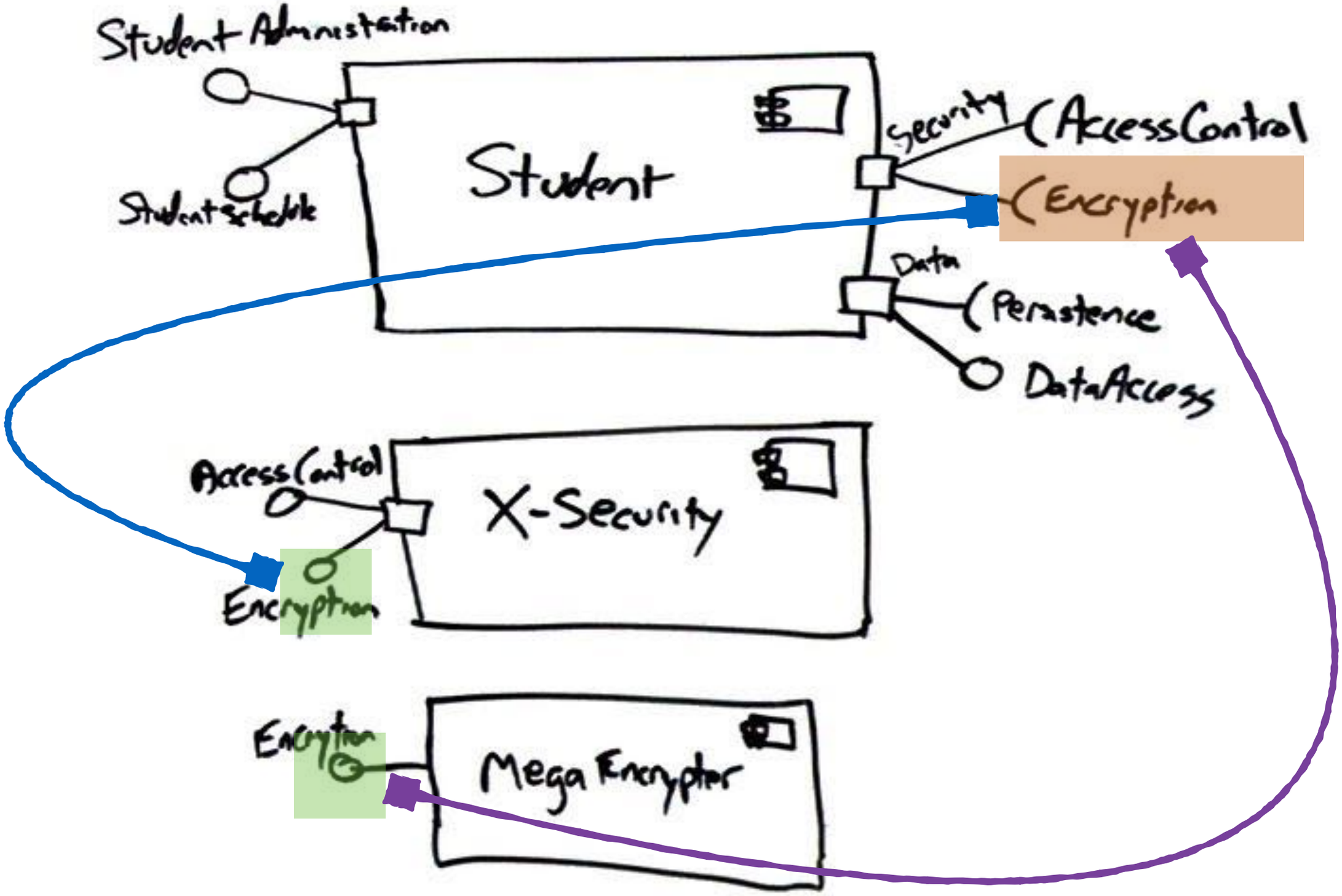
3A

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

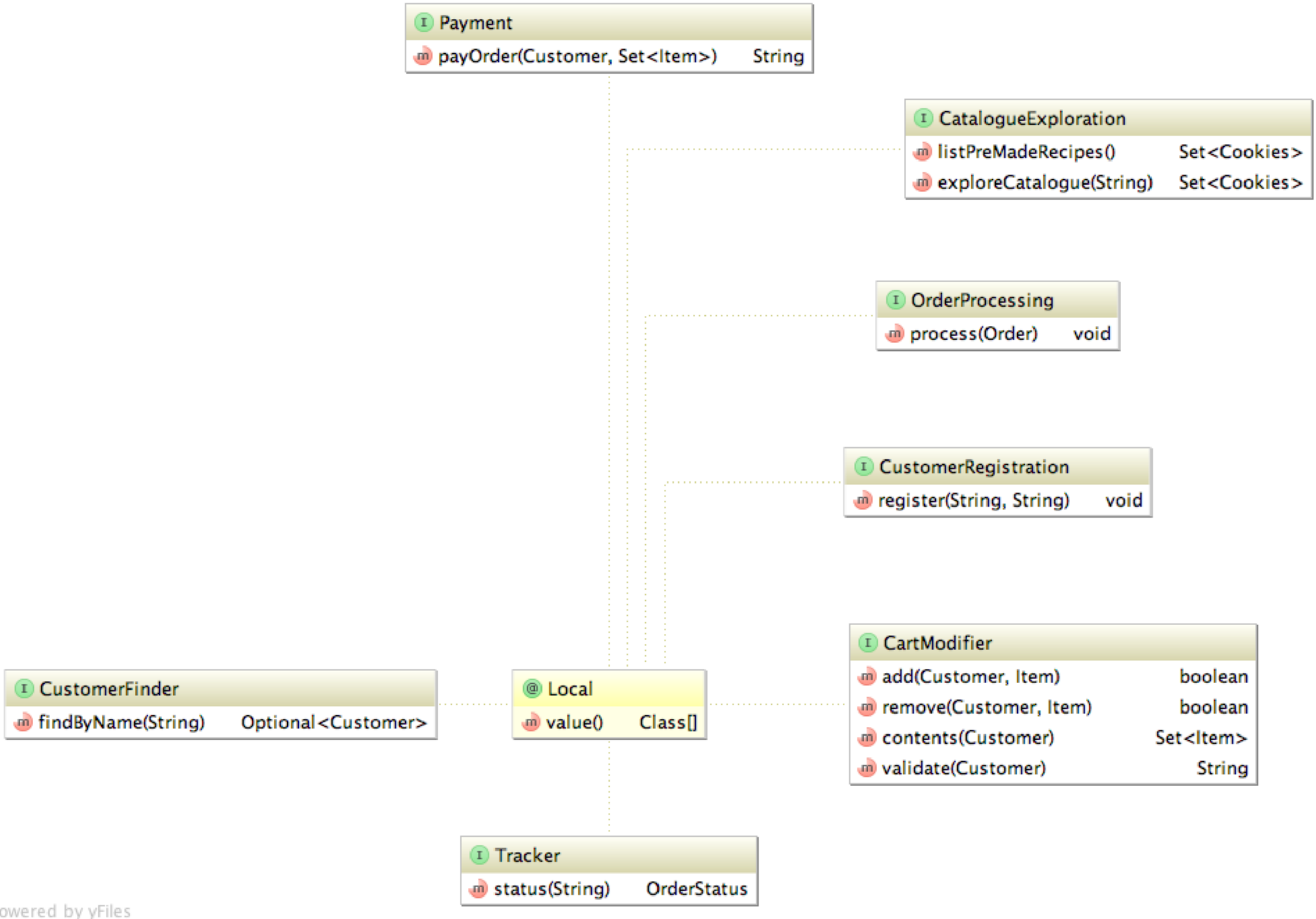
```
class Student implements
        StudentAdministration, StudentSchedule {

  @Inject
  private Encryption e;

}
```

Required Interface

TCF MVP

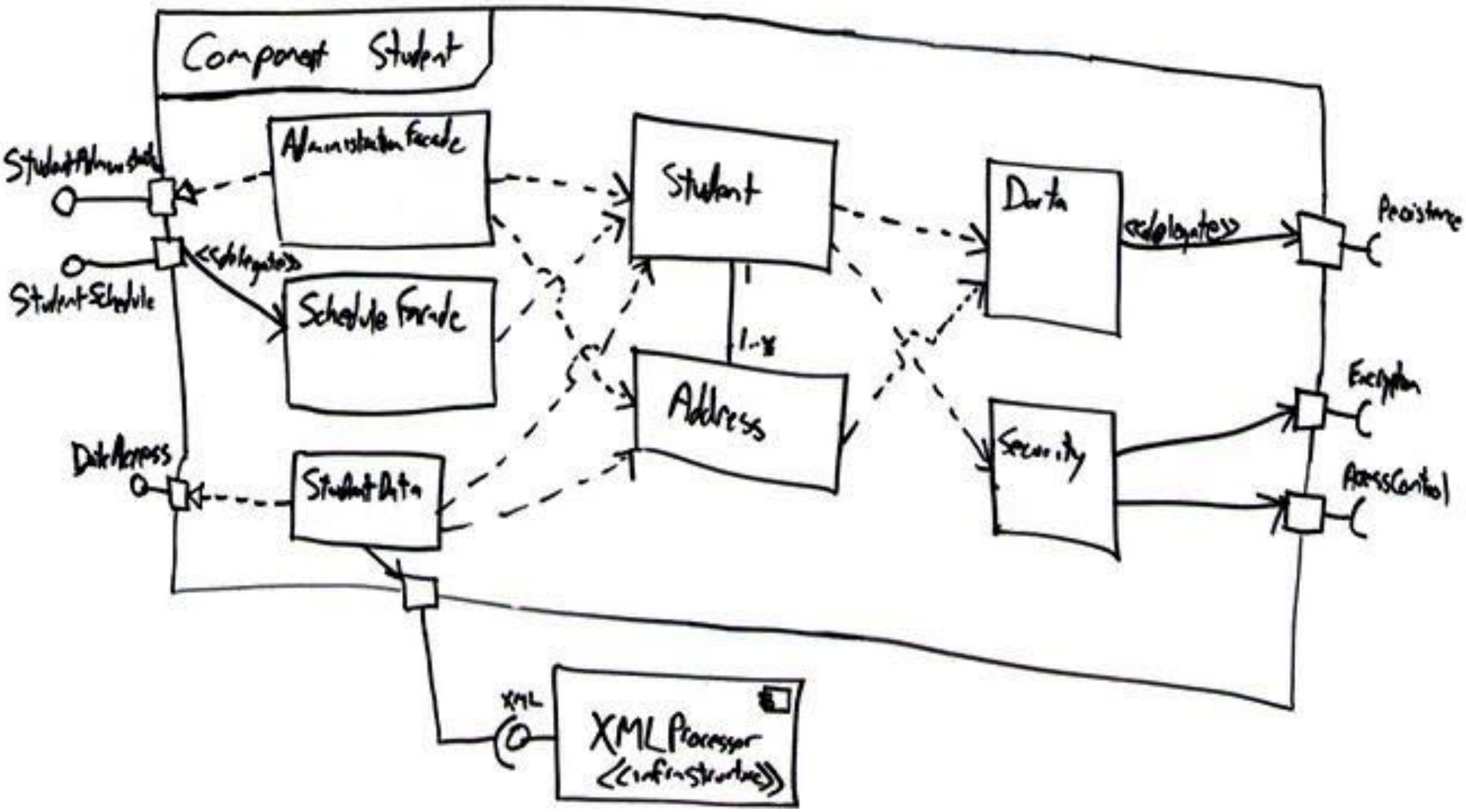
Functional Interfaces for TCF ?

Components ?



owered by yFiles

Components Assembly

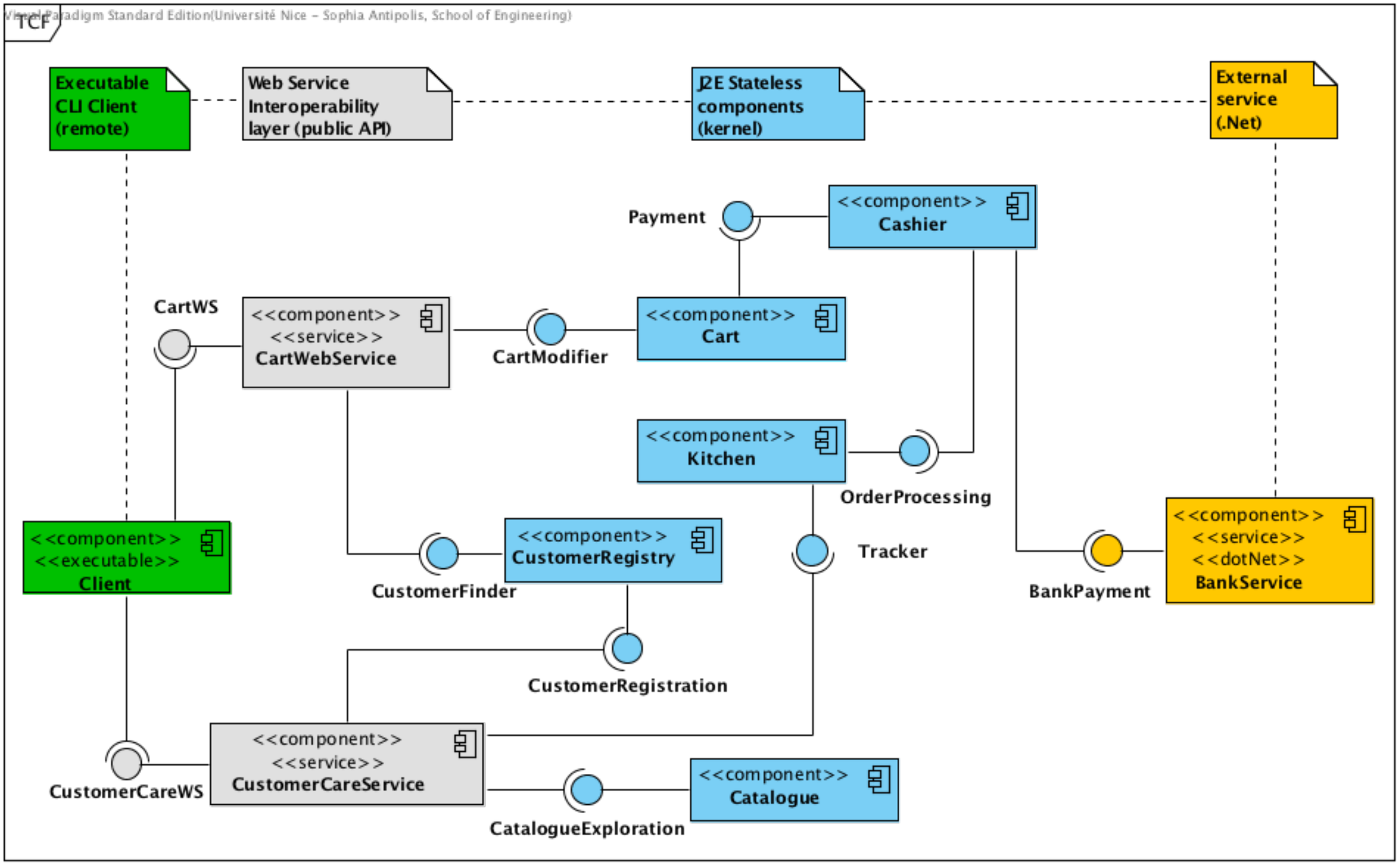


http://www.agilemodeling.com/artifacts/componentDiagram.htm

TCF MVP

Component Assembly ?

Visual Paradigm Standard Edition(Université Nice – Sophia Antipolis, School of Engineering)



Architecture Report (first report)

You must deliver a PDF file at the root of your main repository on Github, named architecture.pdf.

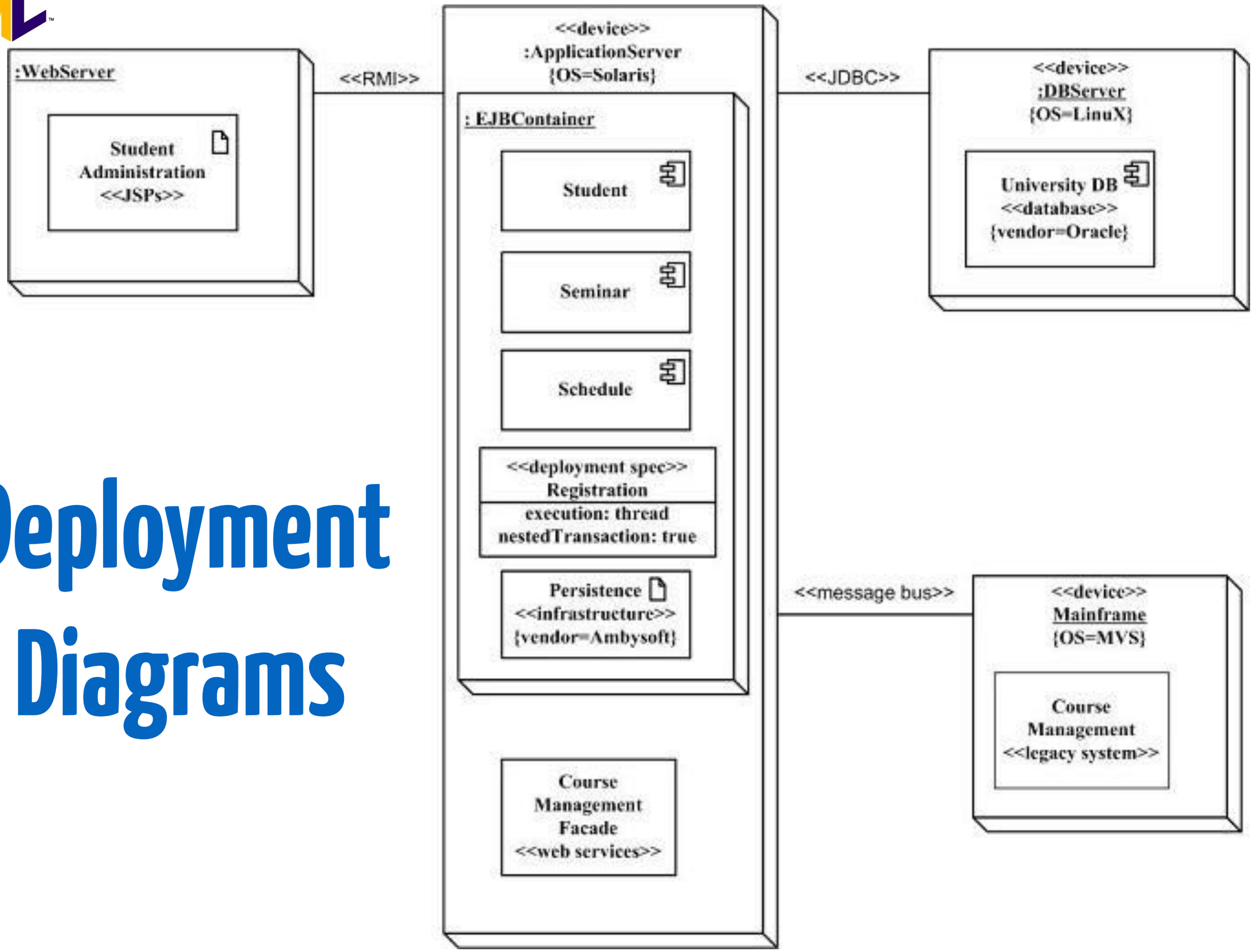
There is no page limit but concision is an evaluation criterion (and your report should be ~10 pages long):
Sunday 14th, February, 8:00pm

It must contain the following architecture description:

- Use cases diagrams;
- Business objects definition as class diagram;
- Interfaces pseudo-code definition (e.g., Java like);
- Components described by a component diagram;

Each artefact must be justified with respect to its relevance in your architecture.

Deployment Diagrams



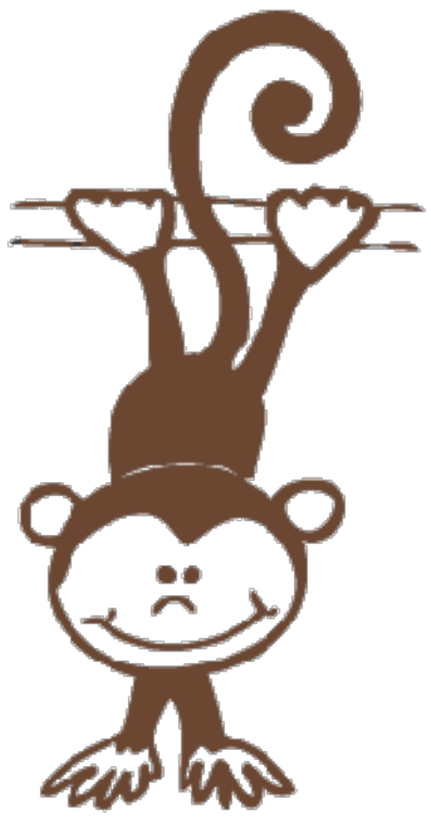
http://www.agilemodeling.com/artifacts/deploymentDiagram.htm

TCF MVP

How to deploy TCF ?

https://github.com/collet/4A_ISA_TheCookieFactory

monkey see



monkey do