SAD Lecture 9

Design Patterns (and principles)

Dave Stikkolorum

Some images are fetched from openclipart.org

Agile Modelling Workshop

Tomorrow we will be at **Alfons** in the morning slot:
8:45 - 12:00
(unfortunately time-edit was not updated)

Kahoot!

Lecture overview

- Recap: transition from Analalysis to Design
- Recap: design principles
- Design Patterns

But first

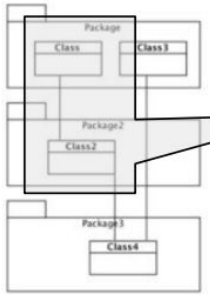
How would you setup a process of developing a piece of software including modelling?

Software development using modelling

Can it be done in an Agile way?

product backlog
domain model
requirements

initial
architecture



sprint planning
(sprint backlogs)

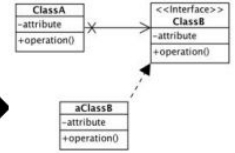
sprint 1

sprint ..

sprint n

sprint execution

detailed design
of increment n



Example Case

Monitoring Runners' Health

Case Description

Monitoring Health

A group of runners wants to monitor several personal statistics for analysing their health during exercising. There are four main matters they want to measure: heart rate, blood pressure, distance and time. Every runner owns a watch that can do the heart rate measurement and a pressure belt for the measurement of the blood pressure.

The group wants a system that combines the input from the watch and the belt. The system should be able to show distance, speed, heart rate, blood pressure during running or when having a pause. The system should be able to send health logs to a doctor (for support during professional advice).

Noun + Verb Identification

Monitoring Health

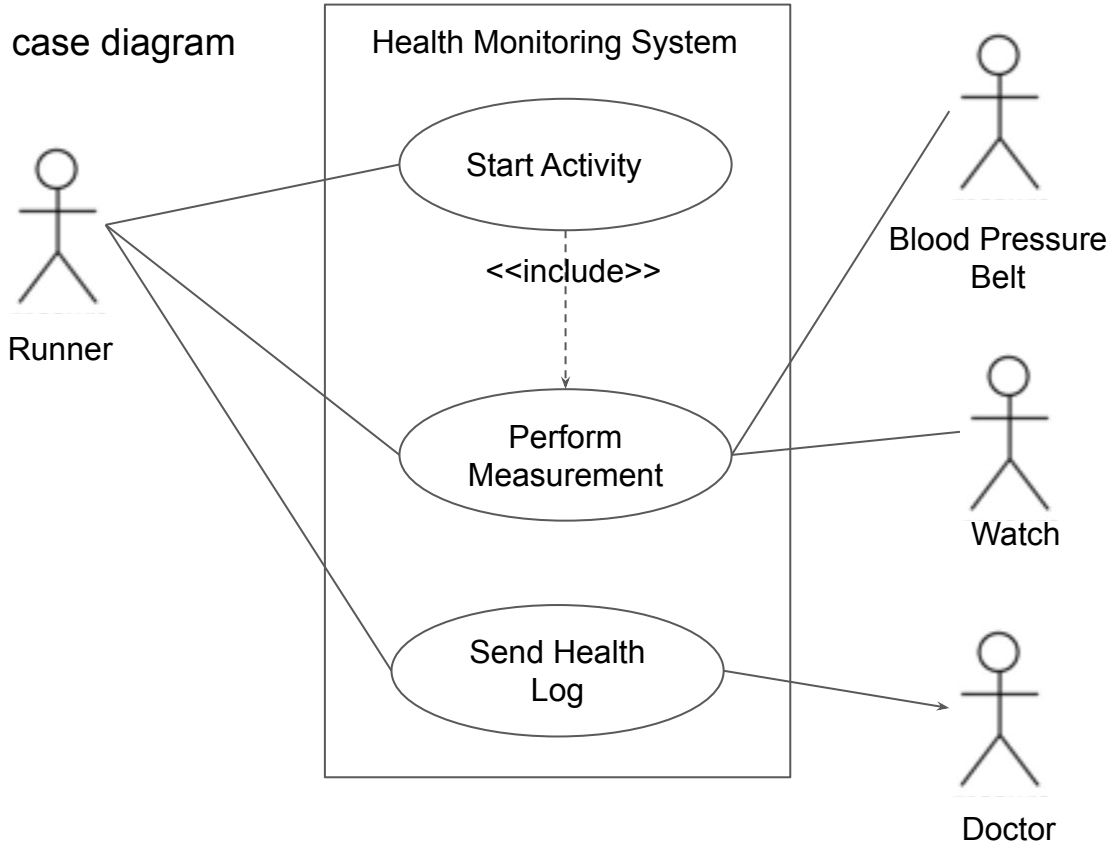
A group of runners wants to monitor several personal statistics for analysing their health during exercising. There are four main matters they want to measure: heart rate, blood pressure, distance and time. Every runner owns a watch that can do the heart rate measurement and a pressure belt for the measurement of the blood pressure.

The group wants a system that combines the input from the watch and the belt. The system should be able to show distance, speed, heart rate, blood pressure during running or when having a pause. The system should be able to send health logs to a doctor (for support during professional advice).

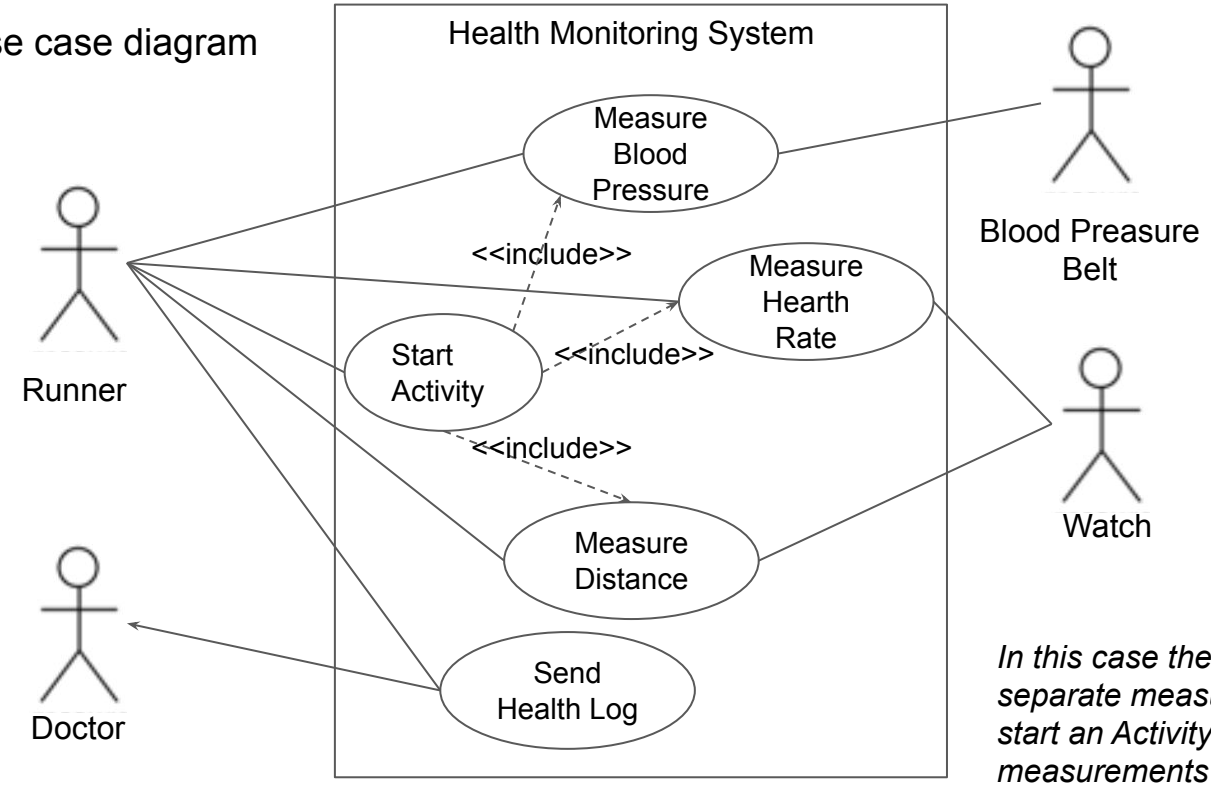
Verb Identification - selecting use cases

~~Wants~~ - vague, help verb
Monitor - Measure?
~~Exercising~~ - not an activity of the system
~~Are~~ - vague, help verb
to measure -> Use Case?: ***Measure*** ….. / ***Perform Measurement***?
owns -> could say something about relation
~~can~~ - vague, help verb
combines - related to analyse
should be able to show -> Use Case: ***Show Status***
should be able to send -> Use Case: ***Send Health Log***
~~Having~~ - vague, help verb
~~Send~~ - already covered

1st version use case diagram

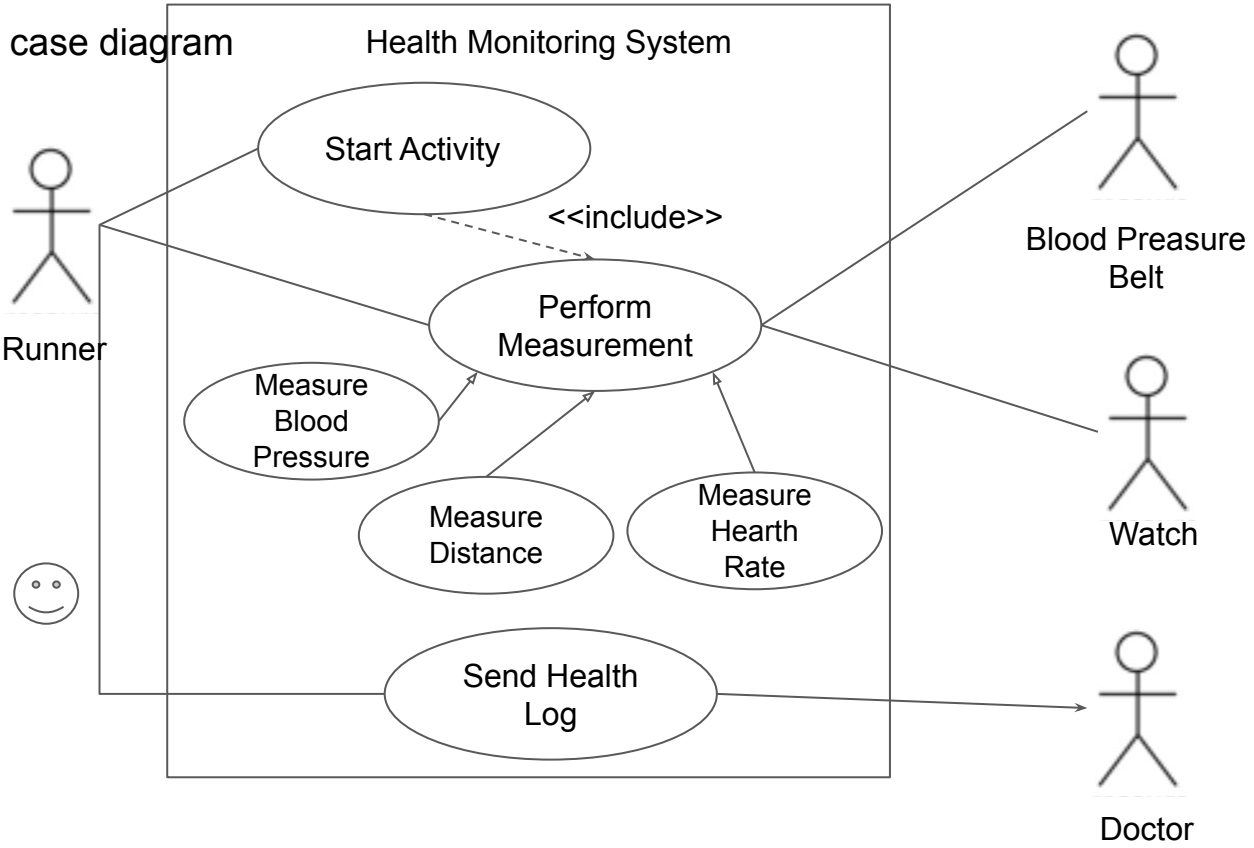


2nd version use case diagram



In this case the Runner can initiate the separate measurements. Also it can start an Activity that starts the measurements
Downside: a bit more complex

3rd version use case diagram



Also possible

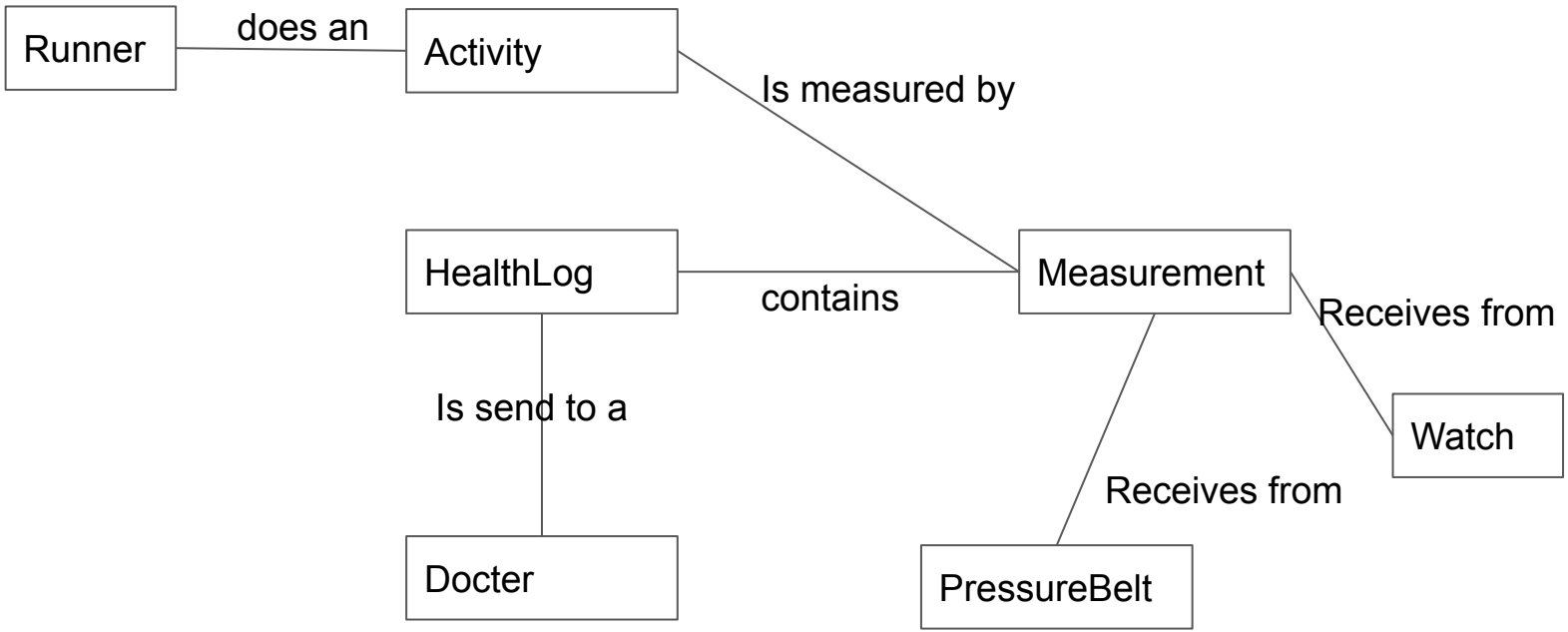
Noun identification - candidate classes and selection

~~Group~~ - collection of runners
Runners -> class : **Runner**
~~statistics~~ - vague, related to measurements
~~Health~~ - vague, related to measurements
~~matters~~ - just vague ;-)
heart rate - attribute of **Measurement**
blood pressure - attribute of **Measurement**
distance - attribute of **Measurement**
~~Time - class **Time**?~~ We assume time is available, so implicit
Watch - class: **Watch**
Measurement: class **Measurement**
pressure belt - class **PressureBelt**
blood pressure - attribute of **Measurement**
~~System~~ - very abstract term, end situation, not applicable now

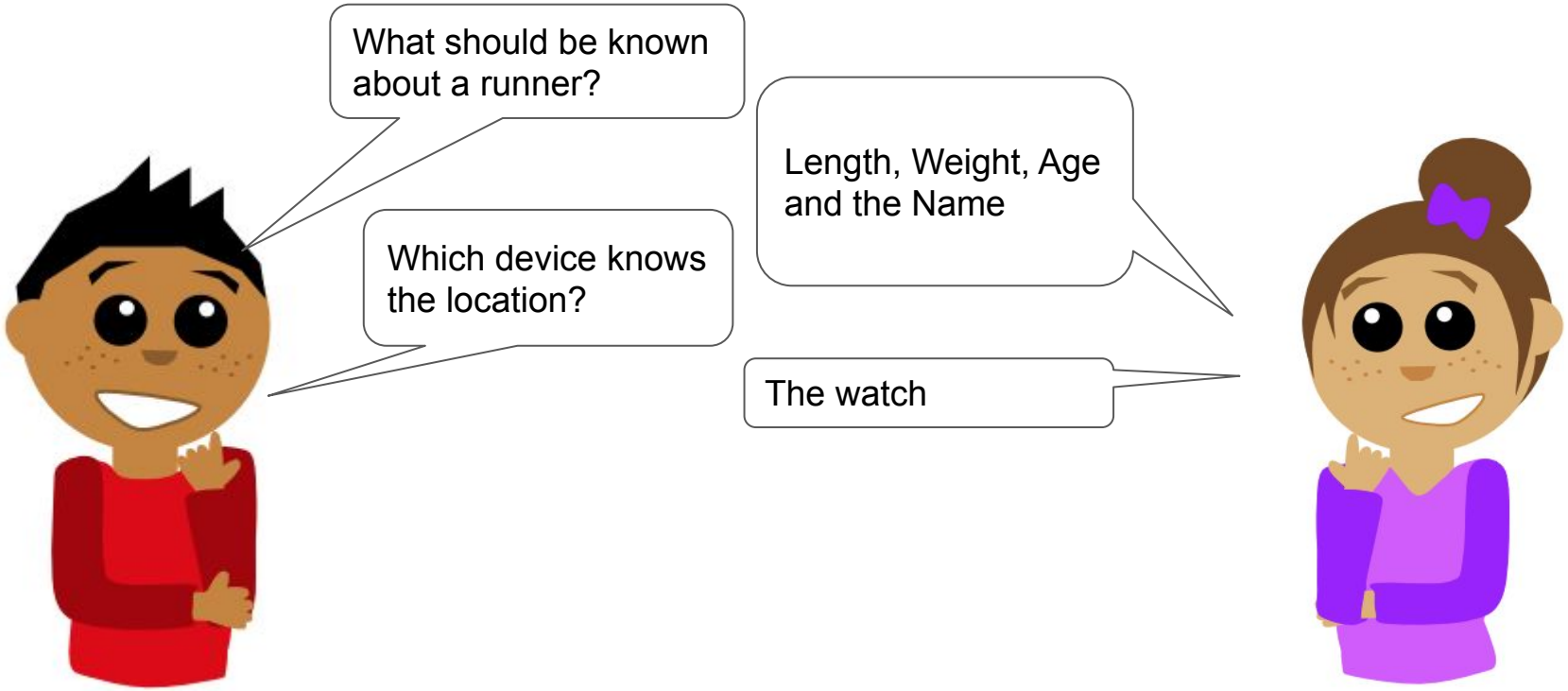
~~Input~~ - vague related measurement
Speed - attribute of **Measurement** (calculated)
Running - activity -> class **Activity**
~~Pause~~ - action of runner, not relevant
health logs -> class **HealthLog**
Doctor -> class: **Doctor**
~~Support~~ - outside of scope case

~~Advice~~ - outside of scope case

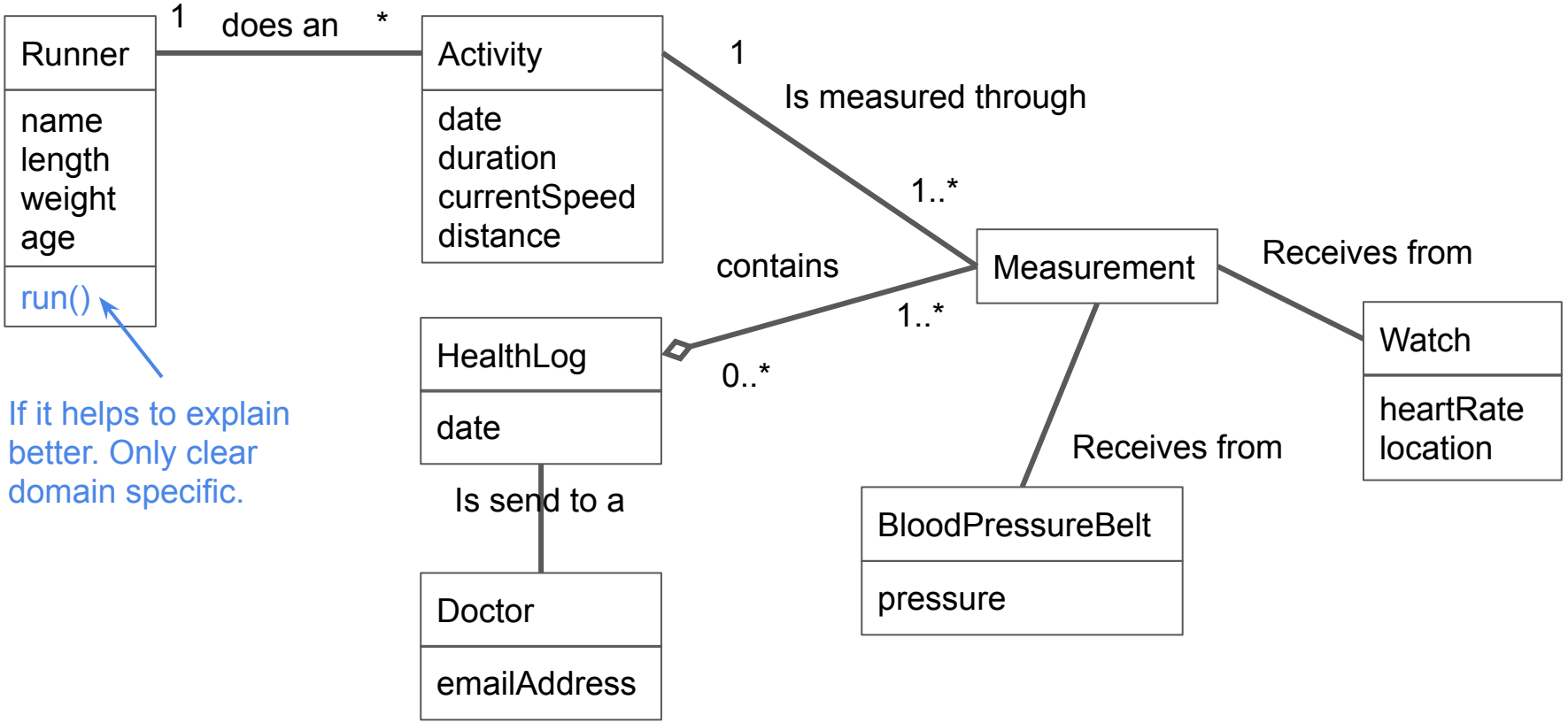
1st version Domain Model



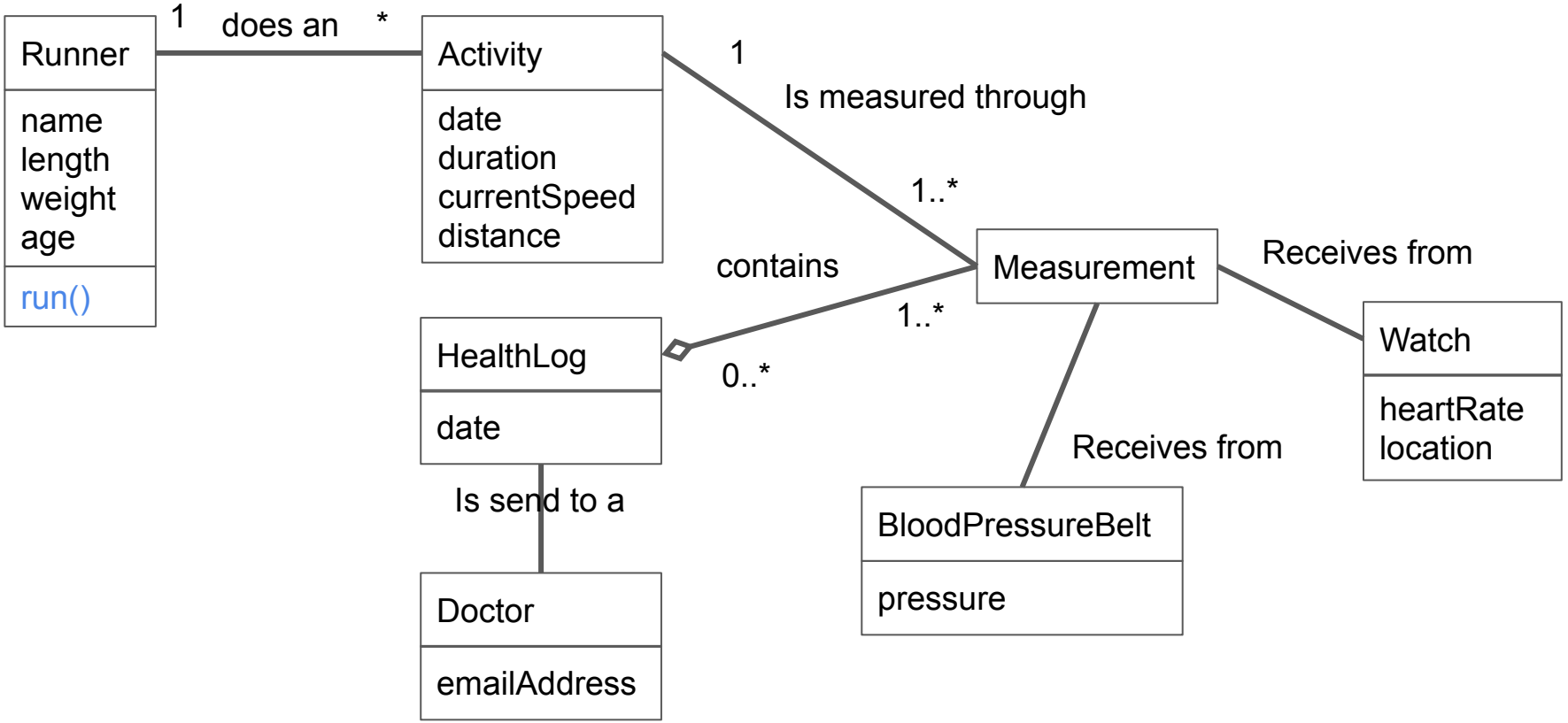
In the meantime, a discussion with the client

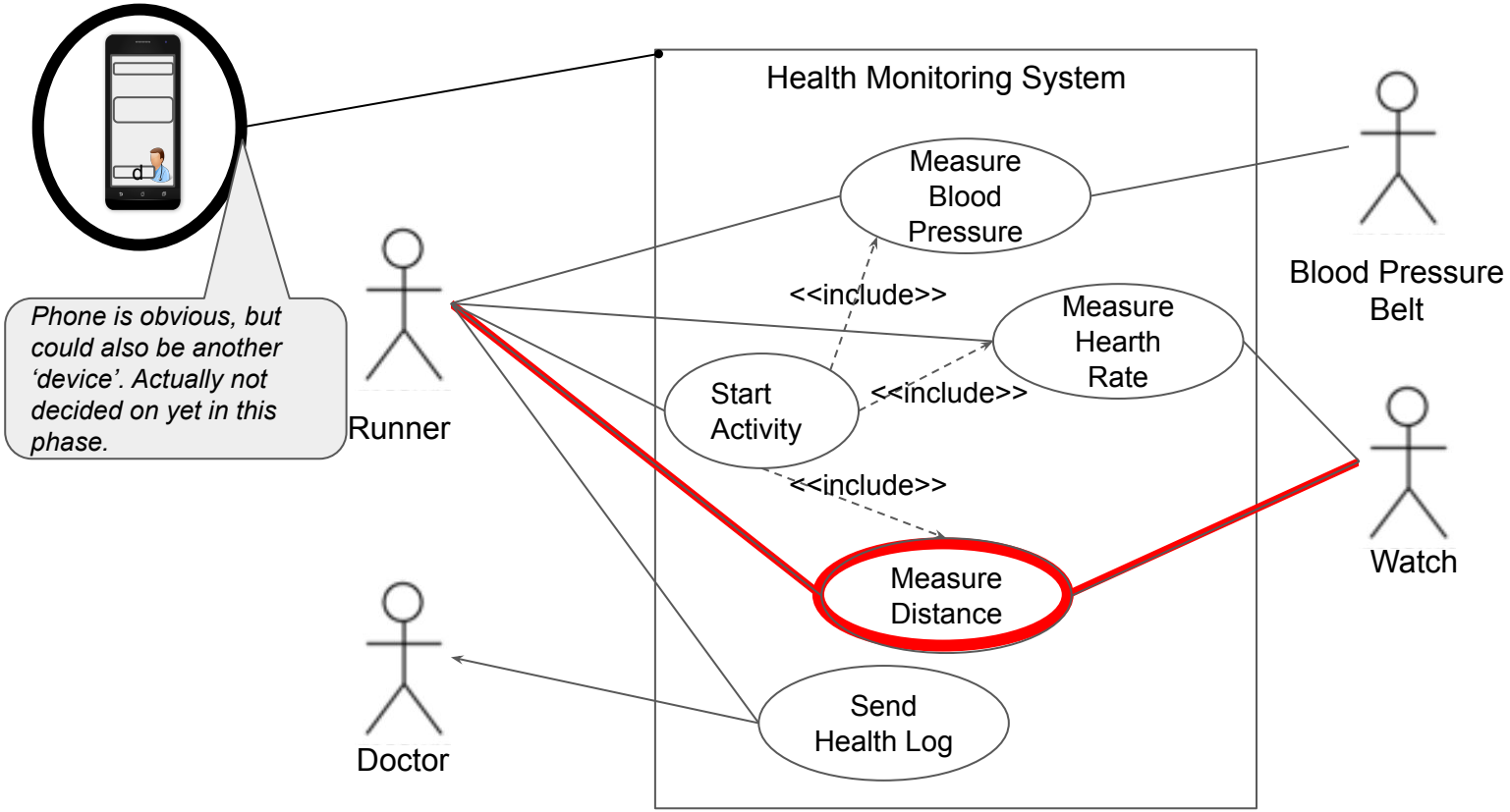


2nd version Domain Model



2nd version Domain Model




Health Monitoring System
Measure Blood Pressure
Blood Pressure Belt
<<include>>
Measure Hearth Rate
Start Activity
<<include>>
<<include>>
Runner
Watch
Measure Distance
Doctor
Send Health Log
Phone is obvious, but could also be another 'device'. Actually not decided on yet in this phase.

Use Case Specification

Use case: <u>Measure Distance</u>

Brief description: Runner wants to measure distance from A to B

Actors: Runner

Basic Flow
1.Use case begins when runner starts measuring and starts running. 2. System shows progress. 3.Runner runs on. 4. System shows distance 0 km. 5. Runner reaches B and stops the measuring. 6. System shows final distance. The use case ends.

Alternative Flows
1. At 1, the system is still on from a previous run. The runner ***resets trip value*** and the use case resumes from 2.

Exceptions
1. At 3, the runner stops for so long that the system turns off the measuring. Use case aborts.

Scenario's
S1. Basic Flow
S2. Basic Flow, A1
S2. Basic Flow, A2

Use Case Specification

(alternative notation)

Use case: Measure Distance

Brief description: Runner wants to measure distance from A to B

Actors: Runner

Basic Flow

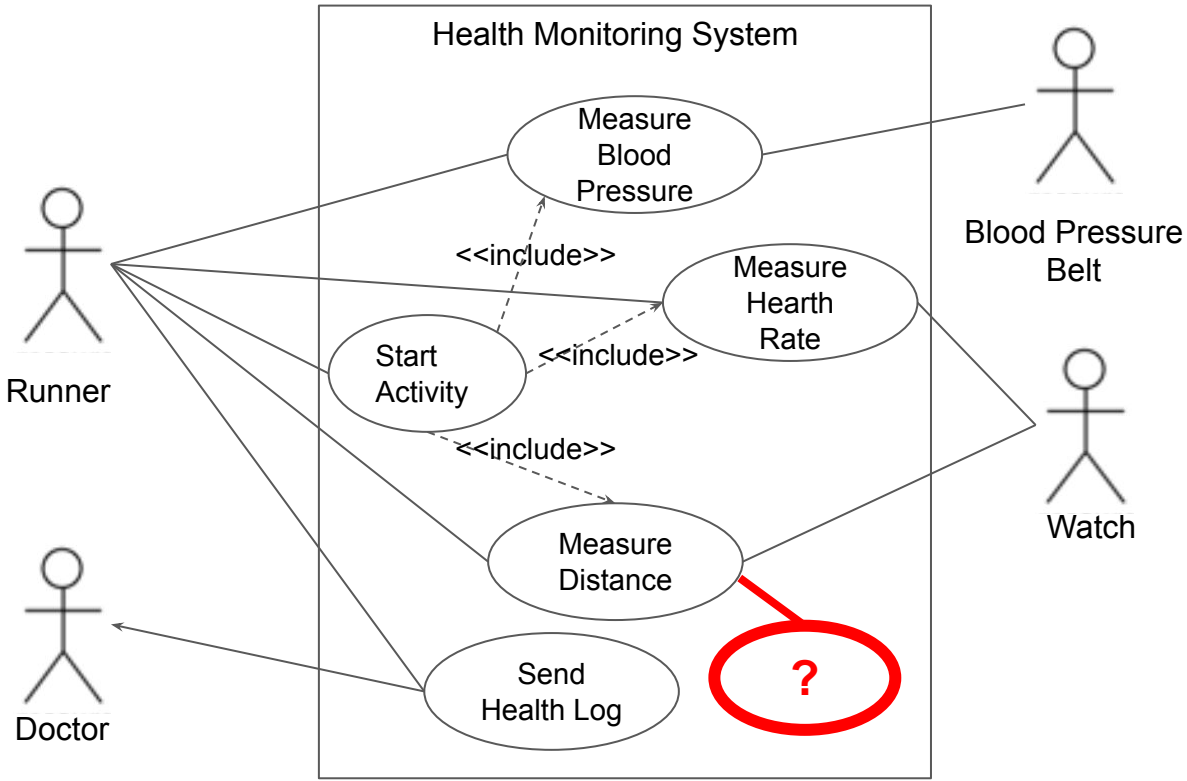
User	System
1.Use case begins when runner starts measuring and starts running. 3.Runner runs on. 5. Runner reaches B and stops the measuring.	2. System shows progress. 4. System shows distance 0 km. 6. System shows final distance.The use case ends.

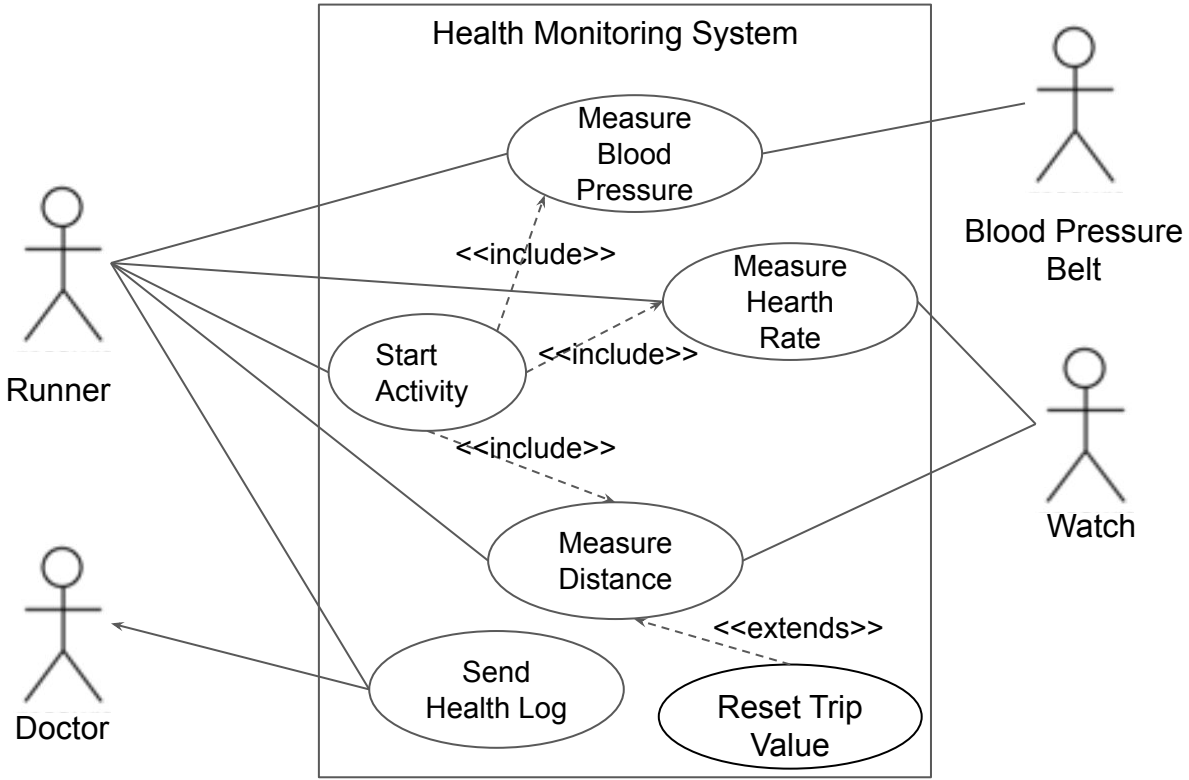
Alternative Flows

1. At 1, the system is still on from a previous run. The runner ***resets trip value*** and the use case resumes from 2.

Exceptions

1. At 3, the runner stops for so long that the system turns off the measuring. Use case aborts.


Health Monitoring System
Measure Blood Pressure
Measure Hearth Rate
Start Activity
Measure Distance
Send Health Log
<<include>>
<<include>>
<<include>>
?
Runner
Doctor
Blood Pressure Belt
Watch


Health Monitoring System
Measure Blood Pressure
Measure Hearth Rate
Start Activity
Measure Distance
Send Health Log
Reset Trip Value
<<include>>
<<include>>
<<include>>
<<extends>>
Runner
Doctor
Blood Pressure Belt
Watch

Reset trip value

Runner wants to reset the trip value in the computer

Basic Flow

1. Use case begins when the runner decides to reset the computer's trip value.
2. The runner selects the reset function. The system asks for confirmation to reset all trip values.
3. The runner confirms. The system notifies the runner that all trip values have been reset.
4. After 3 seconds, the system returns to its main display. The use case ends.

Alternate flow

A1. At 3, the user rejects resetting all trip values. The system immediately returns to the main display. The use case ends.

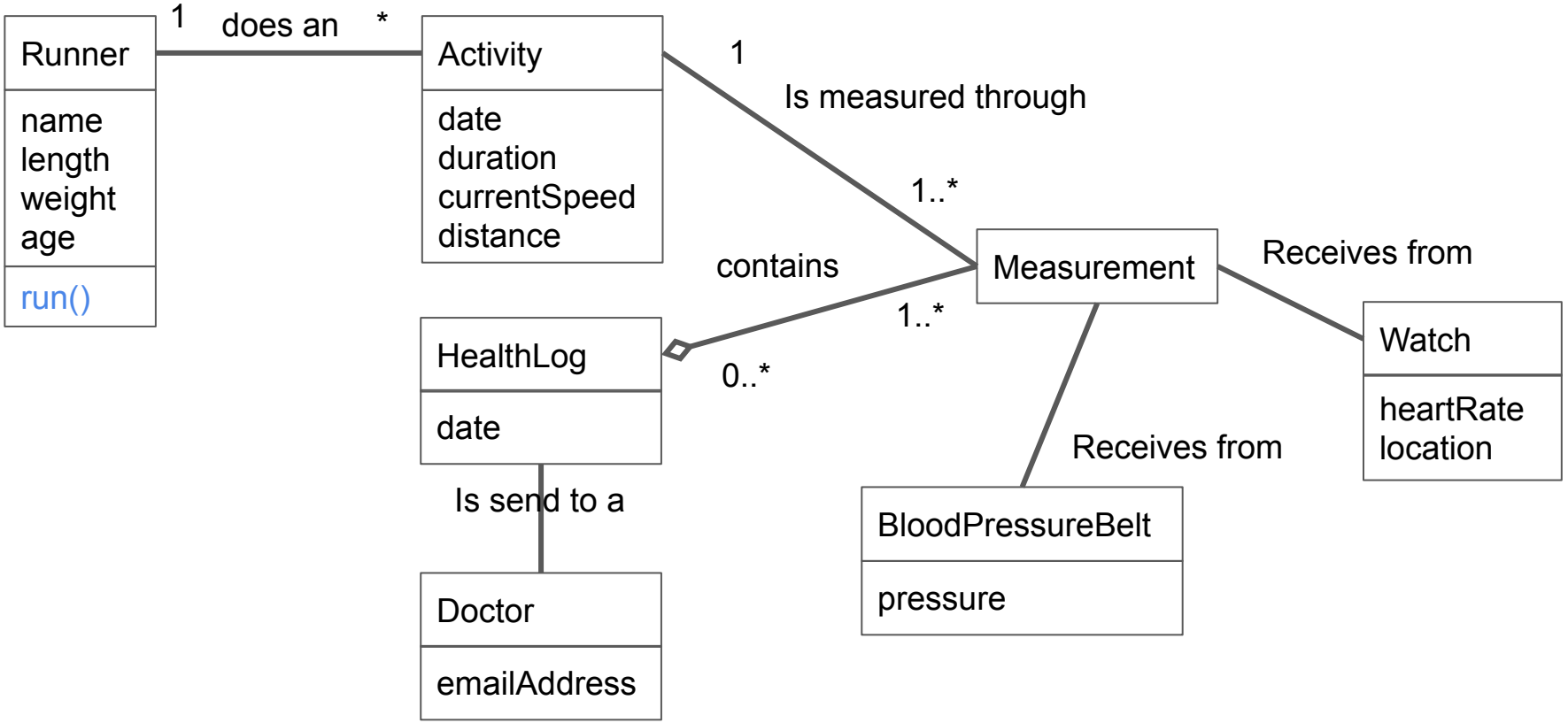
Scenarios

S1. BF

S2. BF, A1

2nd version Domain Model

already satisfied with this one?
Does it reflect the problem?



Refinement step 1/2

USE CRC Cards to complete Complete the diagram

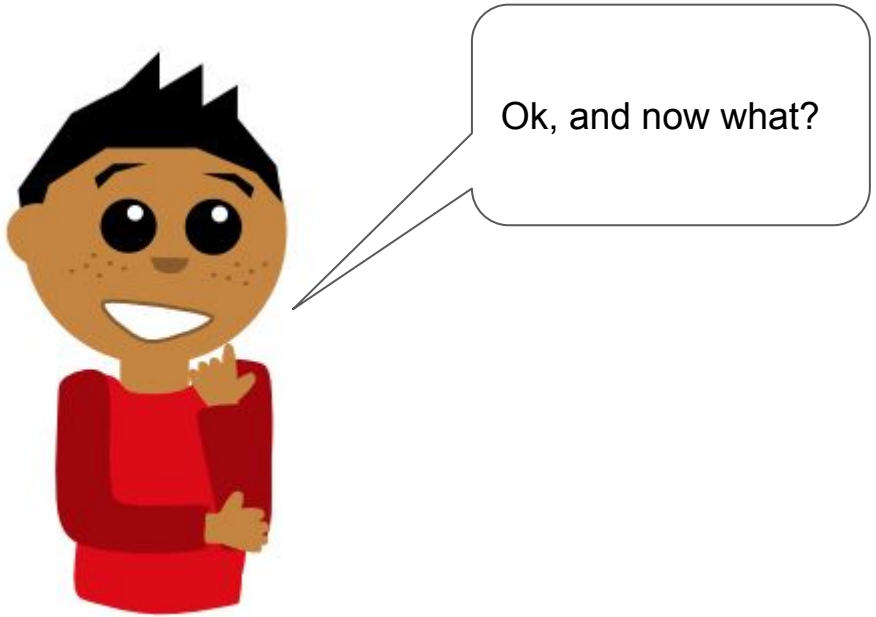
1. Complete the use case descriptions (which one is missing?)
2. Search for candidate classes
    a. Text analysis
    b. Discussion
3. Make CRC cards
4. Validate by simulating the use cases
5. Complete the analysis class diagram

Refinement step 2/2

Use a sequence diagram to show the interaction between domain objects

1. Have a look at the domain model (analysis class diagram)
2. For every use case, make a sequence diagram

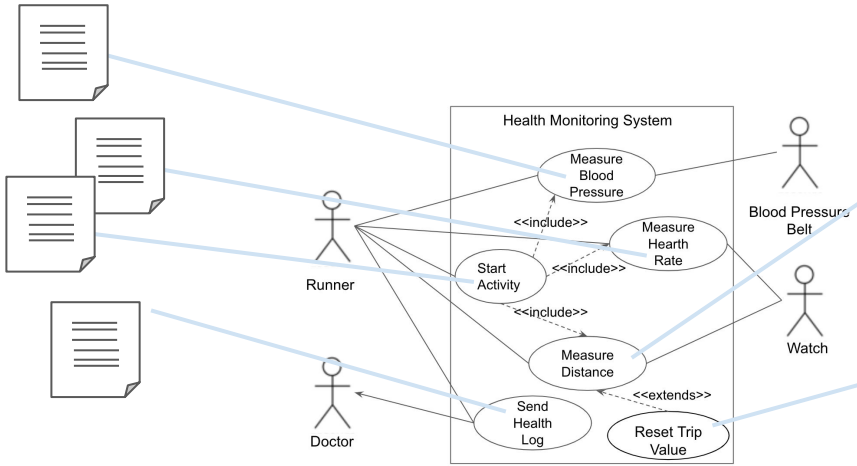
Initial domain analysis done



Requirements

To make the step into the design phase, we need to know the **functional** and **quality** (also called: non-functional) requirements.

Functional requirements we have:



Use Case Specification

Use case: Measure Distance

Brief description: Runner wants to measure distance from A to B

Actors: Runner

Basic Flow
1.Use case begins when runner starts measuring and starts running. 2. System shows progress. 3.Runner runs on. 4. System shows distance 0 km. 5. Runner reaches B and stops the measuring. 6. System shows final distance. The use case ends.

Alternative Flows
1. At 1, the system is still on from a previous run. The runner ***resets trip value*** and the use case resumes from 2.

Exceptions
1. At 3, the runner stops for so long that the system turns off the measuring. Use case aborts.

Scenario's
S1. Basic Flow
S2. Basic Flow, A1
S2. Basic Flow, A2

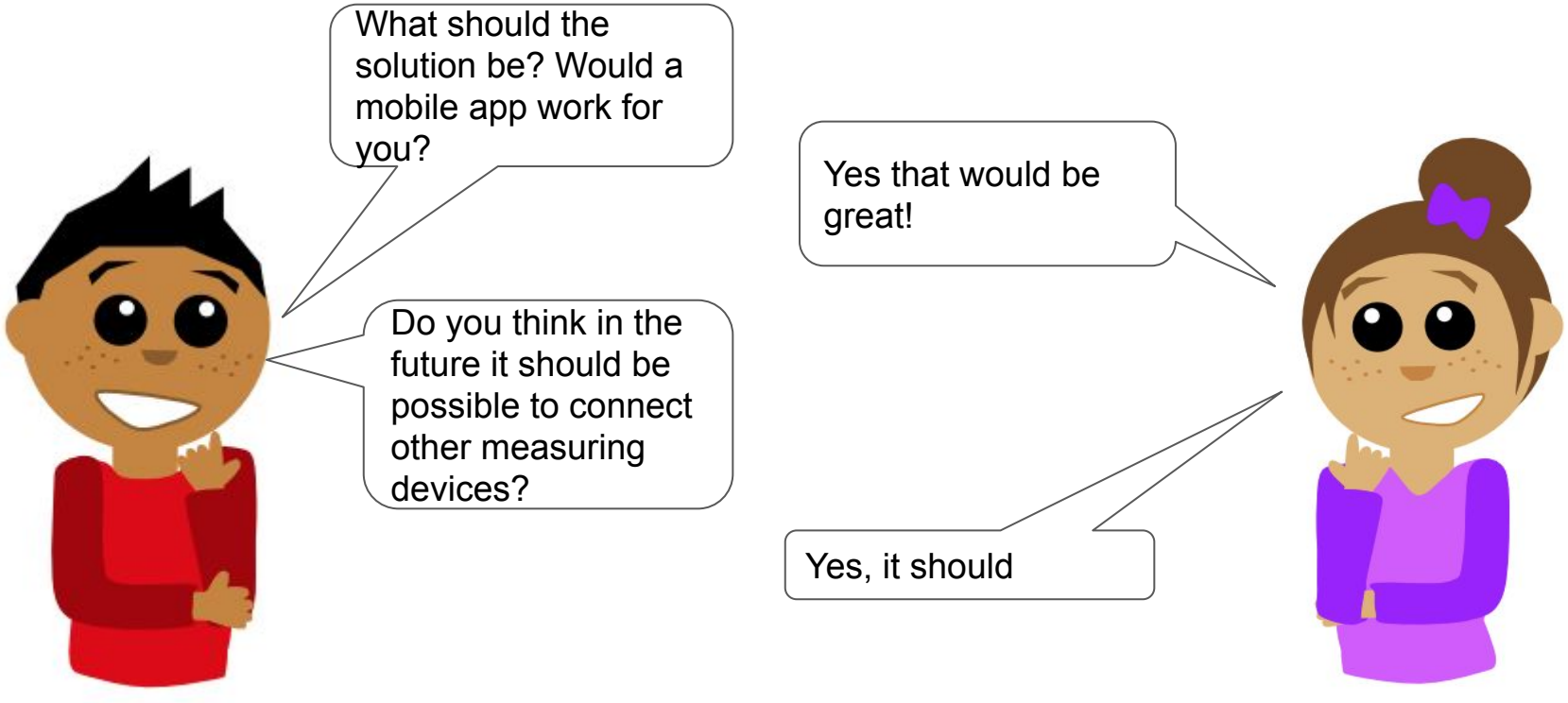
Reset trip value

Runner wants to reset the trip value in the computer
Basic Flow
1. Use case begins when the runner decides to reset the computer's trip value.
2. The runner selects the reset function. The system asks for confirmation to reset all trip values.
3. The runner confirms. The system notifies the runner that all trip values have been reset.
4. After 3 seconds, the system returns to its main display. The use case ends.

Alternate flow
A1. At 3, the user rejects resetting all trip values. The system immediately returns to the main display. The use case ends.

Scenarios
S1. BF
S2. BF, A1

Quality Requirements



Quality Requirements Mobile Health App

- The language should be English, but in a later stage languages should be added
- The interface should be modern and fit to the phone’s style
- Measurement units and formats that should be used
  - Blood pressure in mmHg
  - Heart Rate in bpm
  - Time in hours, minutes and seconds (hh:mm:ss)
  - Distance in km and meters
- In the future it should be possible to add new measuring devices
- For supporting measuring devices, bluetooth should be used

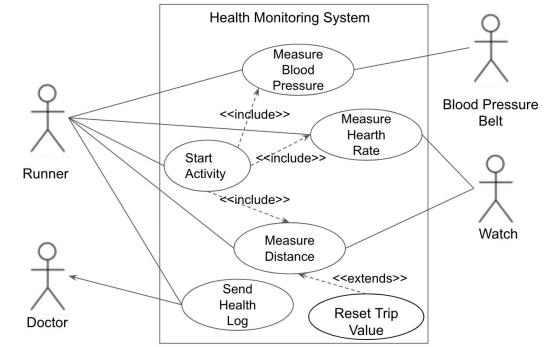
Quality Requirements Mobile Health App

Other requirements after discussing with the client and making some decisions:

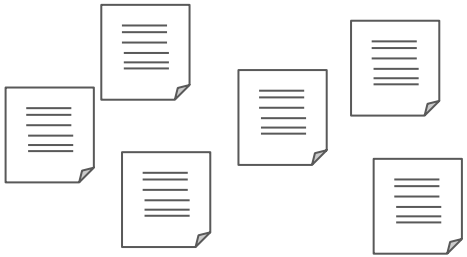
- The first version of the mobile app will be running on Android (this was a money issue)
    - Version: Android 5 - 9
    - Programming language: Java
- Development is done with an agile aproach (such as scrum)

Transitioning to design

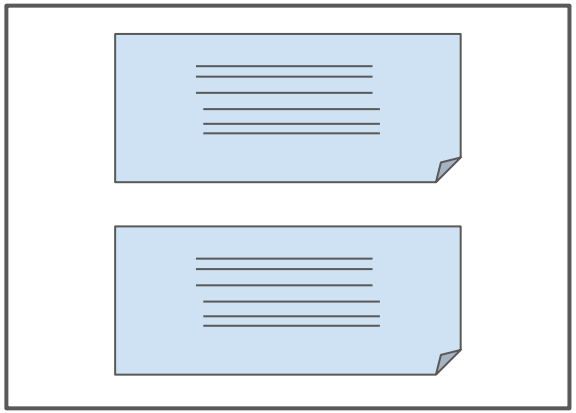
Functional requirements



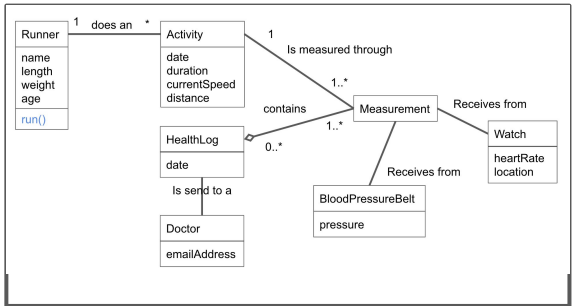
Use cases (diagram+specs)



Quality requirements



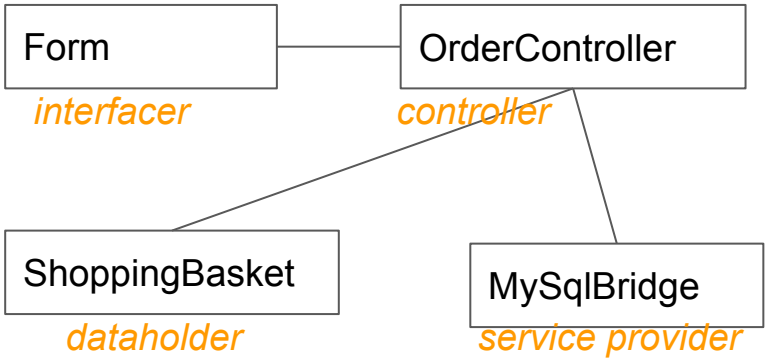
Domain model

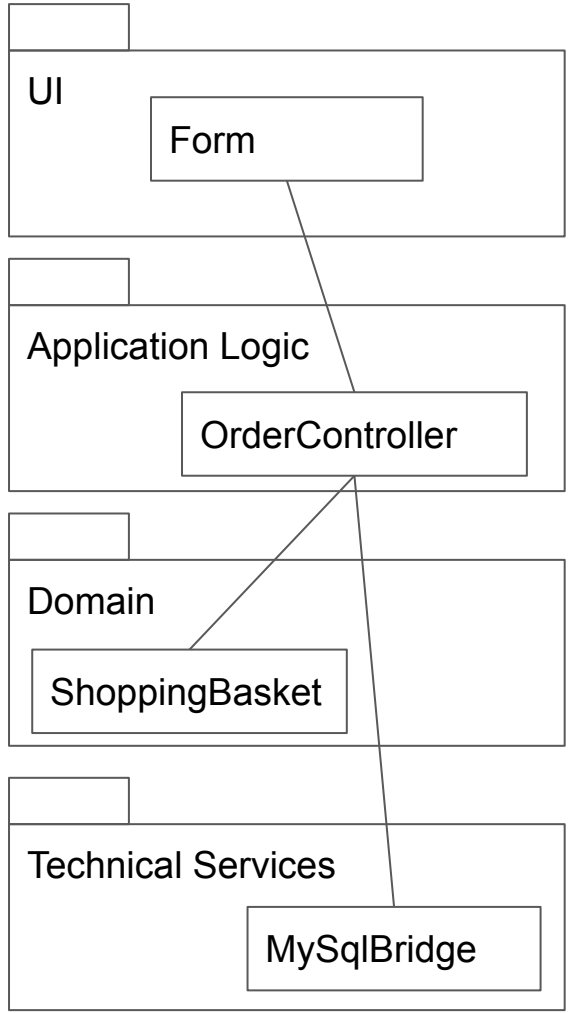


Design

Architecture

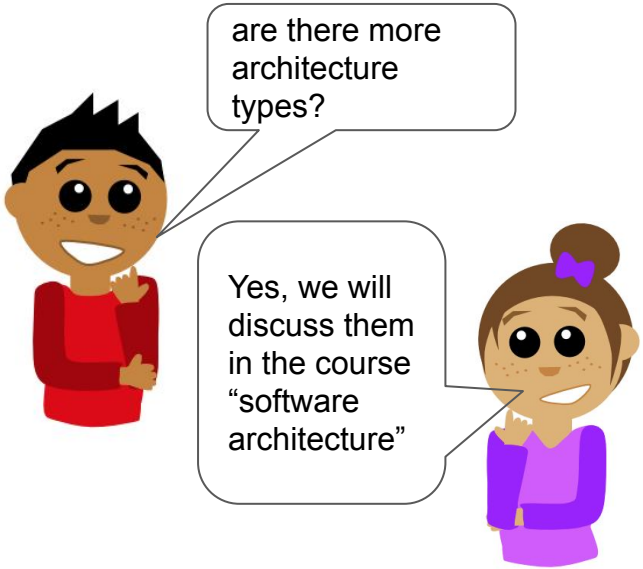
The software classes in a design are distributed over layers with a specific functionality or role: the **software architecture**

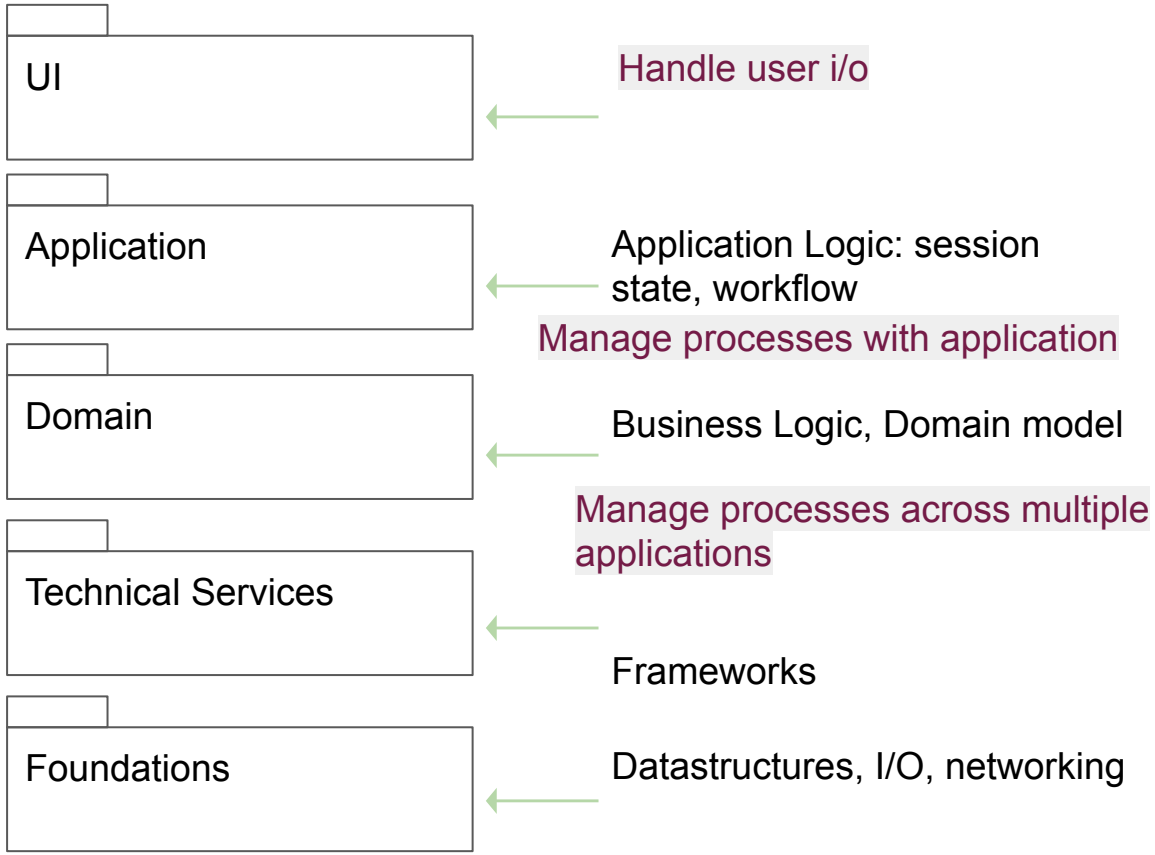




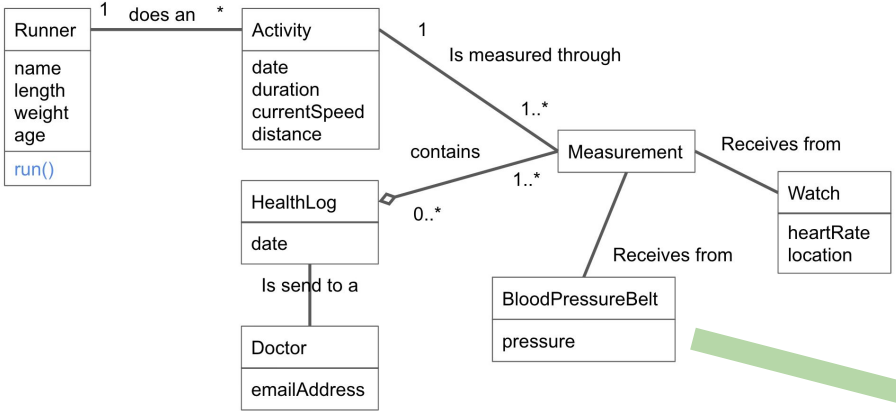
Classical Layering

A typical classical software architecture is the classical n - tier (layer) architecture.





Initial Architecture Health App

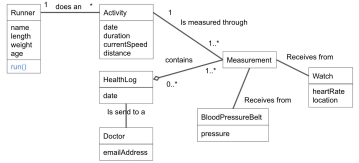


- The first version of the mobile app will be running on Android (this was a money issue)
    - Version: Android 5 - 9
    - Programming language: Java

UI

Application Logic

Domain

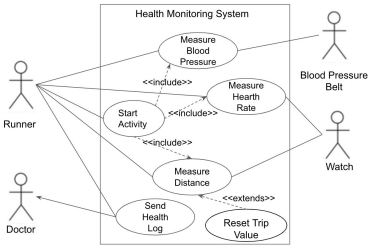


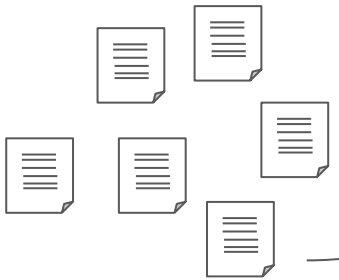
Technical Services

Android

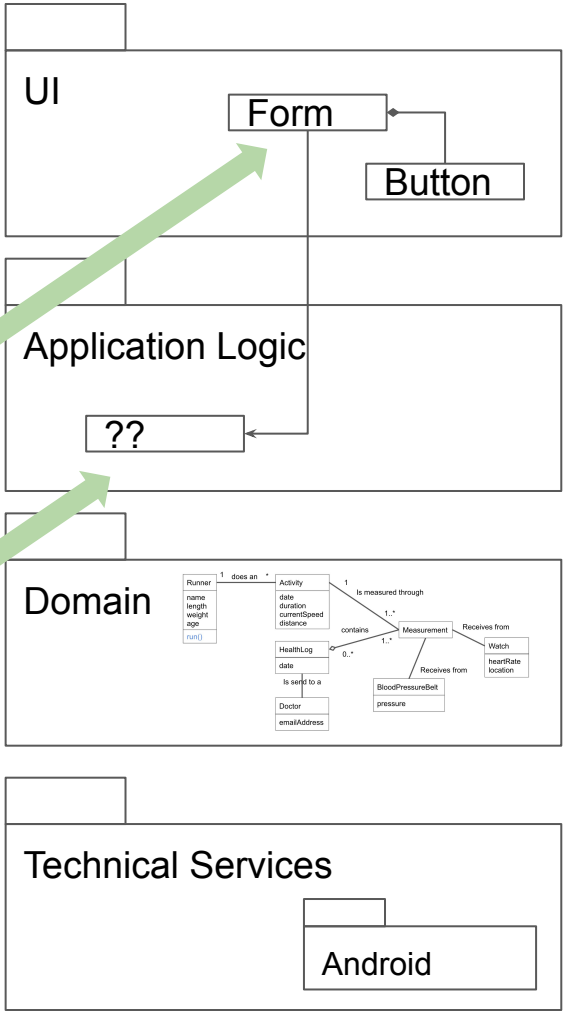
Initial Architecture Health App

For refinement and completing the design (a filled in architecture) further input is need from the functional requirements.





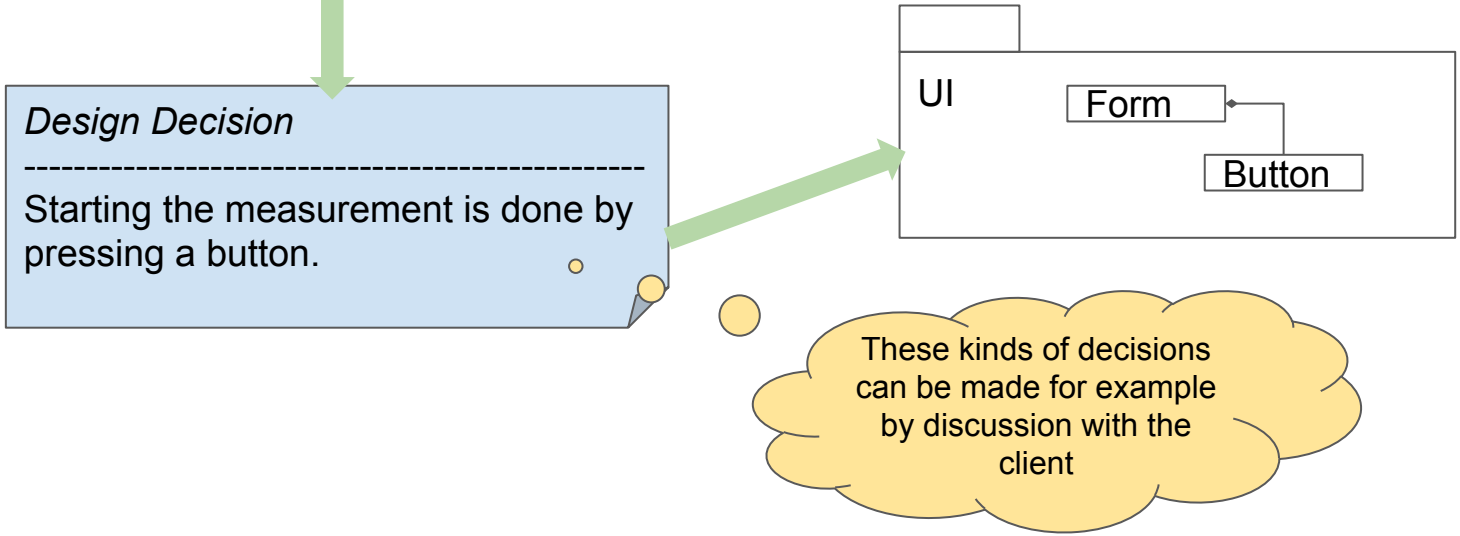
Design Decisions



Design Decision

Basic Flow

1.Use case begins when runner starts measuring and starts running. 2. System shows progress. 3.Runner runs on. 4. System shows distance 0 km. 5. Runner reaches B and stops the measuring. 6. System shows final distance. The use case ends.

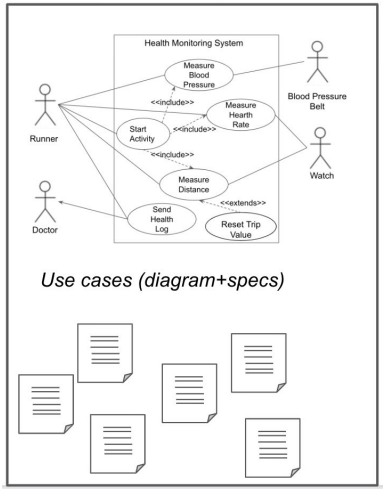


Further Development

Analysis

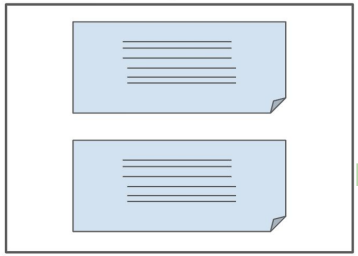
Analyse the problem and deliver and initial architecture

Functional requirements

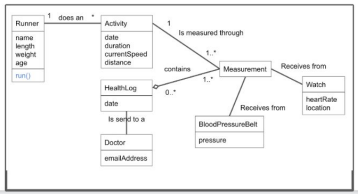


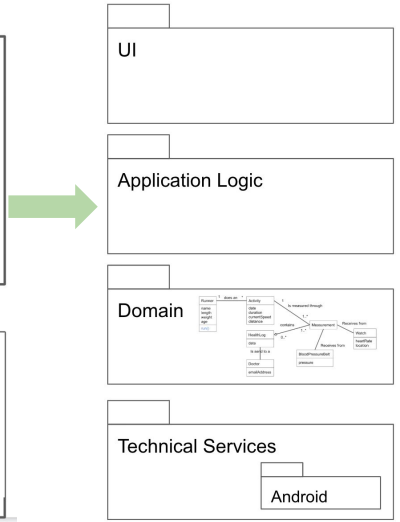
Use cases (diagram+specs)

Quality requirements



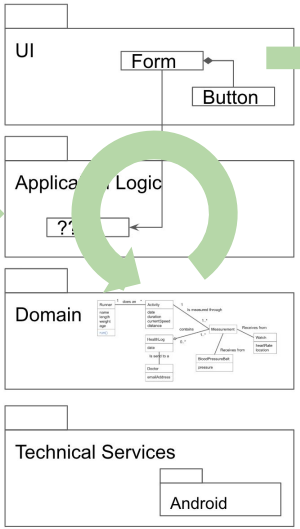
Domain model





Design

Iteratively refine the design

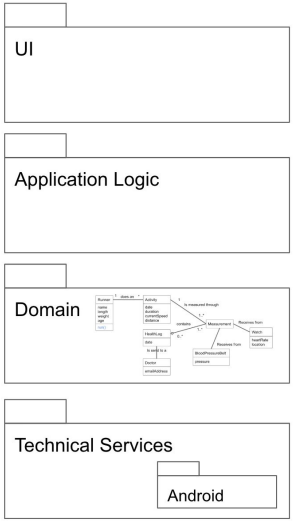


Realize

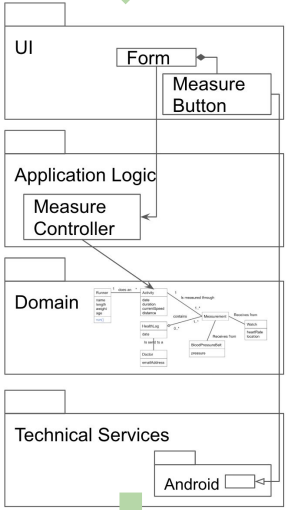
```
Button measureButton =
(Button)
findViewById(R.id.clickButto
n);

measureButton.setOnClickList
ener(new OnClickListener() {
 @Override public void
 onClick(View v){
 }
});
```

Further Development - an Agile Approach

Initial Architecture
after analysis



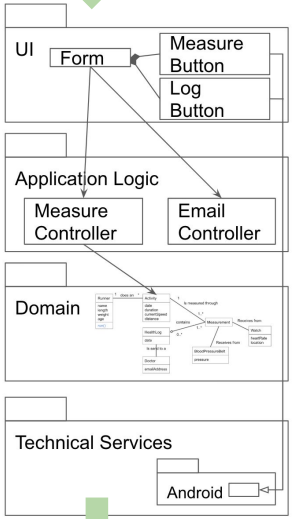
Measure
Distance



Software v1

```
Button measureButton =
(Button)
findViewById(R.id.measureBut
ton);
```

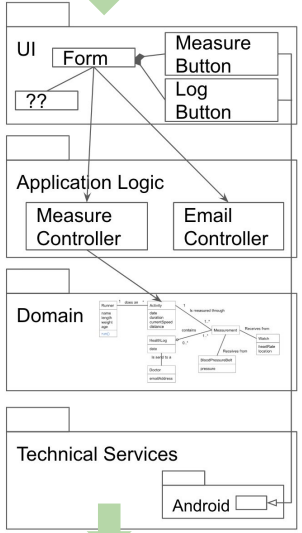
Send
HealthLog



Software v2

```
Button logButton = (Button)
findViewById(R.id.logButton)
;
```

Use Case X

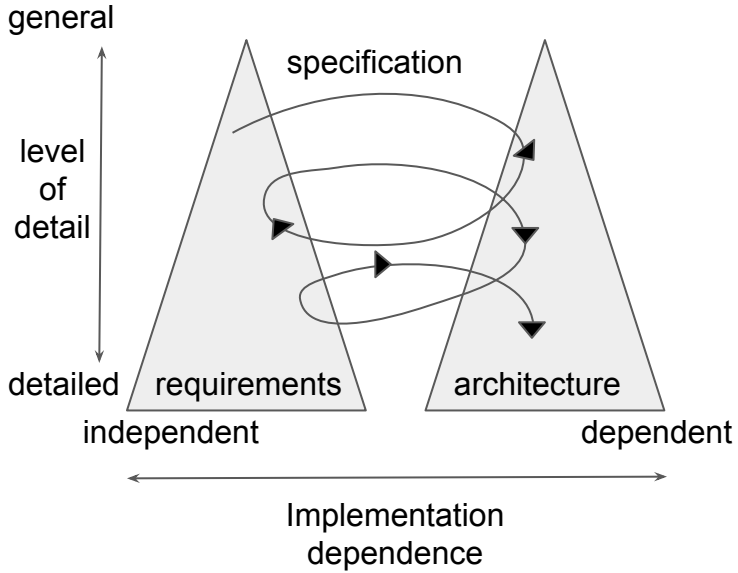


Software vn

```
Aaaaa bbbbb = (ccccc)
dddddd(R.id.eeeee);
```

General Overall Software Development Approach

“A more robust and realistic development process allows both requirements engineers and system architects to work concurrently and iteratively to describe the artifacts they wish to produce. This process allows developers to better understand problems through consideration of architectural constraints, and they can develop and adapt architectures based on requirements.” [1]



Bashar Nuseibeh, 2001

References

[1] Nuseibeh, Bashar. "Weaving together requirements and architectures." *Computer* 34.3 (2001): 115-119.

Recap: Design Principles

Design Principle

Guide or rule to follow for building a solid design

Design Principle - examples 1/2

Norman’s Principles (1988), interface design:

..

2. Simplify the structure of tasks

..

6. Design for error

..

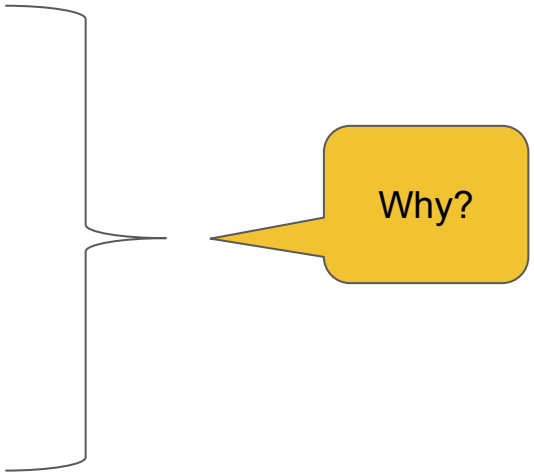
Design Principle - examples 2/2

A outside door should go inwards when opening (house construction)

Software Design Principles 1/4 [RECAP]

Booch [1] et al.

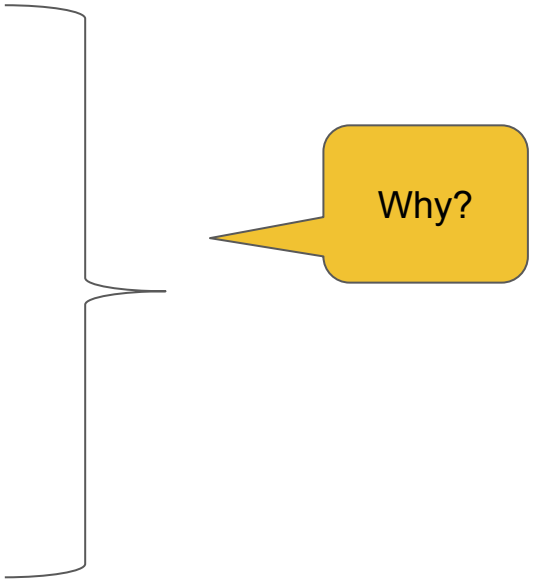
Abstraction
Encapsulation
Modularity
Hierarchy
Typing
Concurrency
Persistence



Software Design Principles 2/4 [RECAP]

General Responsibility Assignment Software Principles - GRASP, Larman [2]

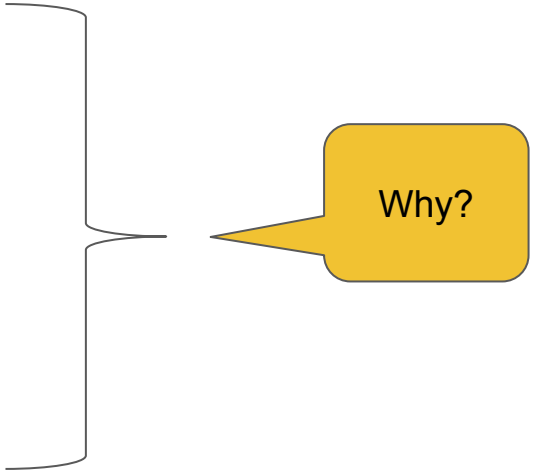
Creator
Information Expert
Low Coupling
Controller
High Cohesion
Indirection
Polymorphism
Protected Variations
Pure Fabrication



Software Design Principles 3/4 [RECAP]

SOLID Robert C. Martin [3]

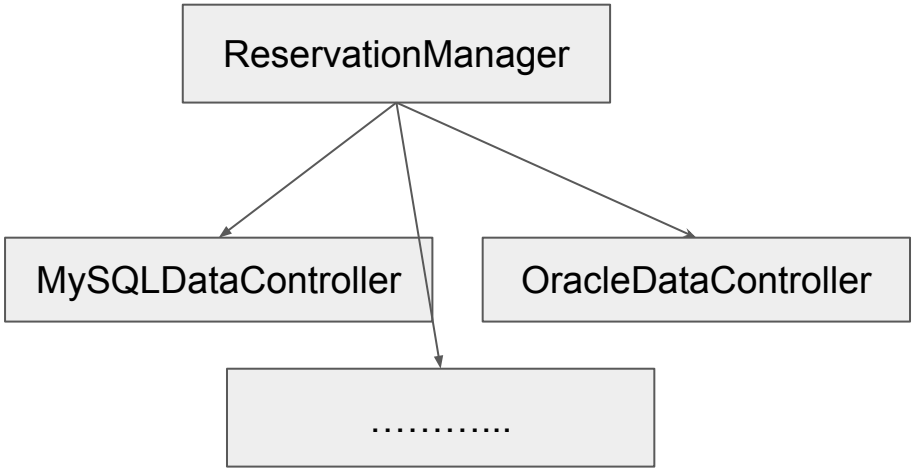
- S – Single-responsibility principle
- O – Open-closed principle
- L – Liskov substitution principle
- I – Interface segregation principle
- D – Dependency Inversion Principle

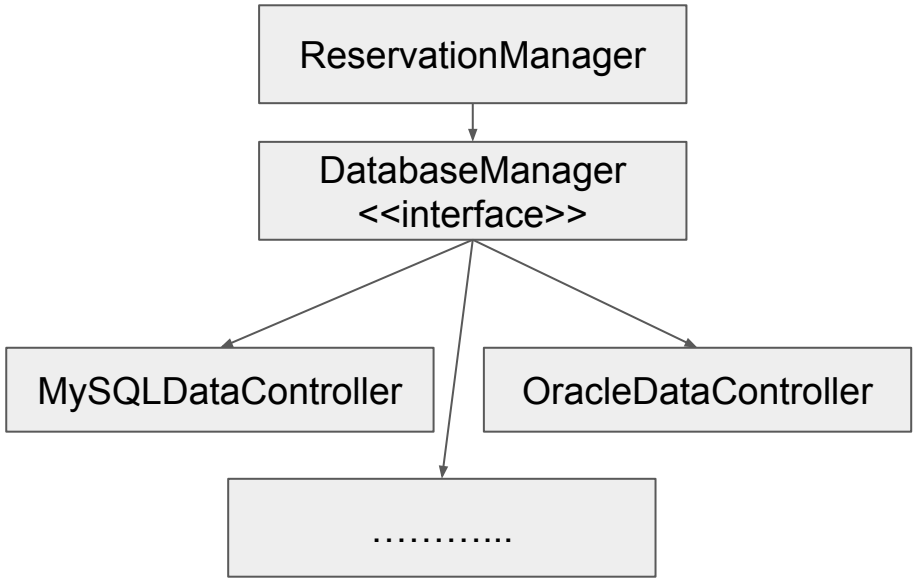


Software Design Principles 4/4 [RECAP]

Information hiding (Parnas)

Reduce the coupling between a specific implementation and the parts of the system that use it.





Design Pattern

a reusable solution for a common problem

Design Principles vs Design Patterns

Principle ← → Pattern

Guideline ← → Solution

Guide or rule to follow for building a solid design

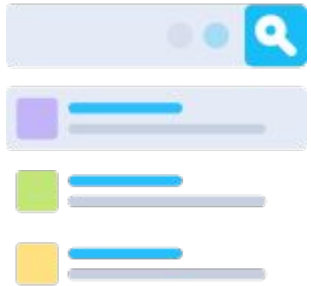
a reusable solution for a common problem

Design Patterns example 1/2

GUI design

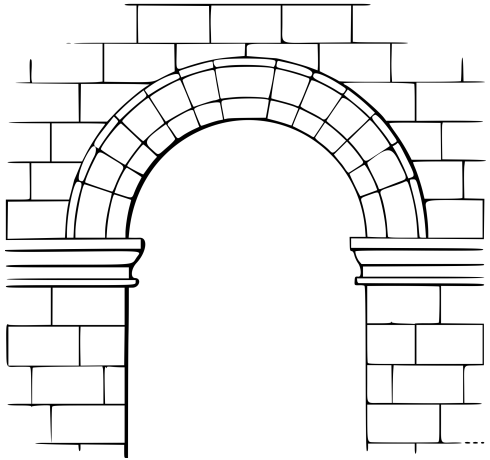
- Forms

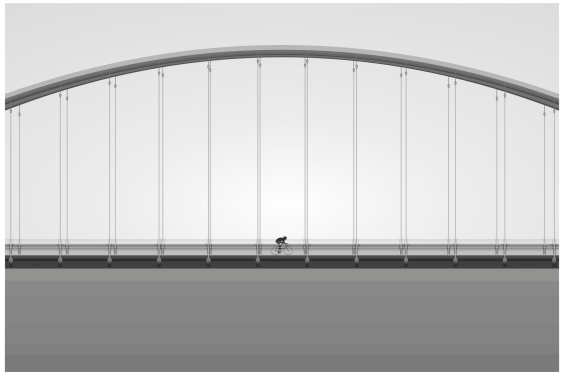
- Search



Design Patterns example 2/2

Construction





Software Design Patterns

What is a Design Pattern?

- General **solution** to a common problem.
- Formalized best practice.

Different common problems

- Behavior
- Creation
- Structured
- Architecture

State Pattern [behavioral]

a pizza delivery process

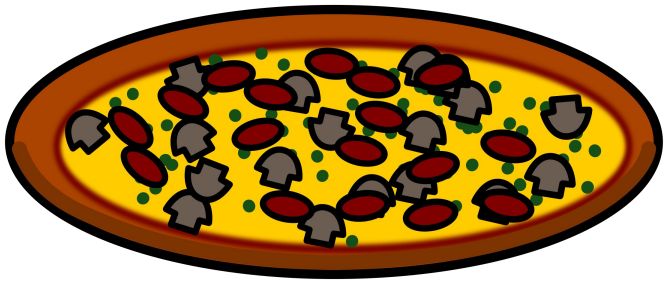
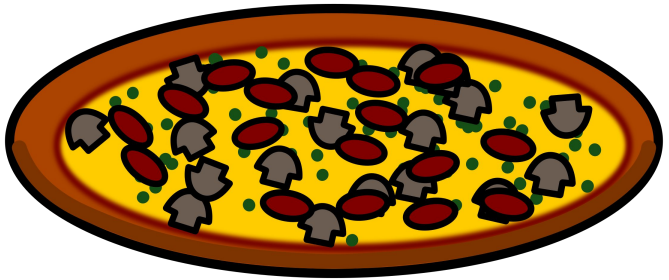


image : www.openclipart.org

State Pattern

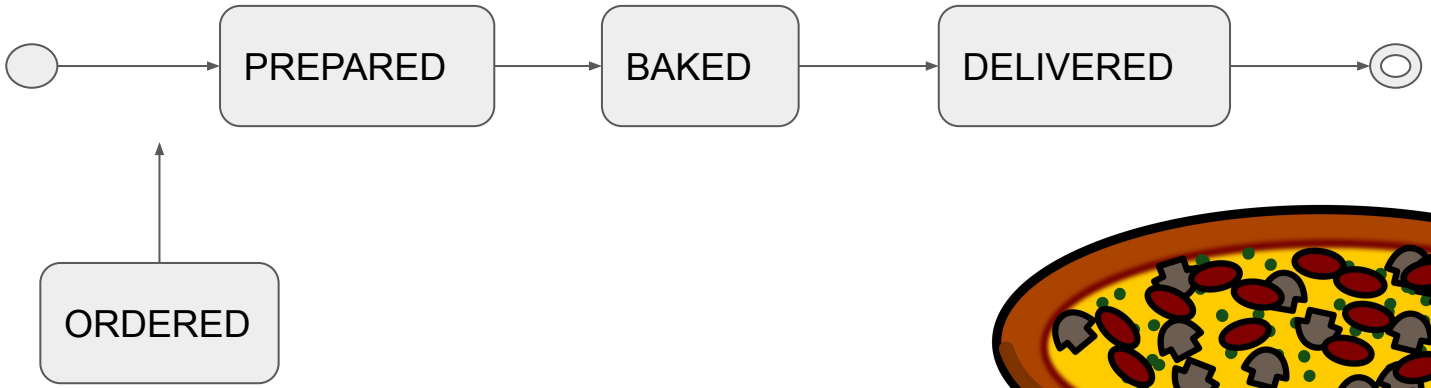
Problem:

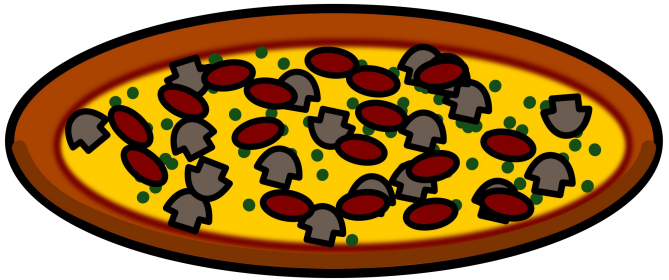
- Behaviour changes when state of object changes
- Add new behaviour without changing the existing behaviour



State Pattern

What if we want to add : ordered?

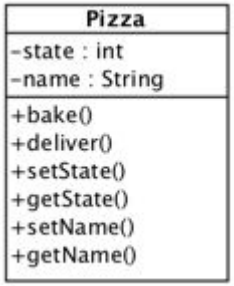




State Pattern in action (for later study)

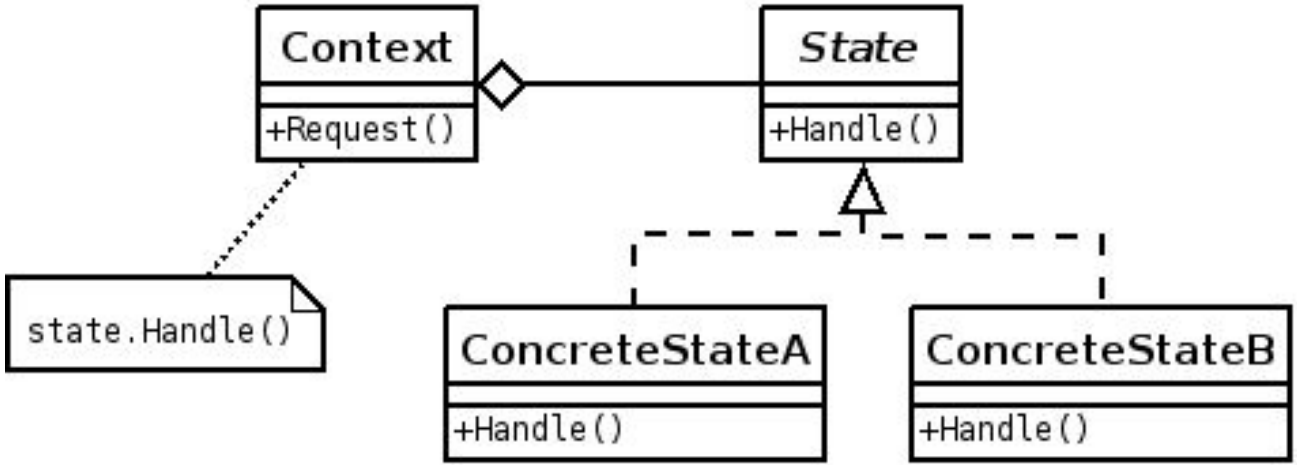
see:

http://onjavahell.blogspot.se/2009/05/simple-example-of-state-design-pattern.html

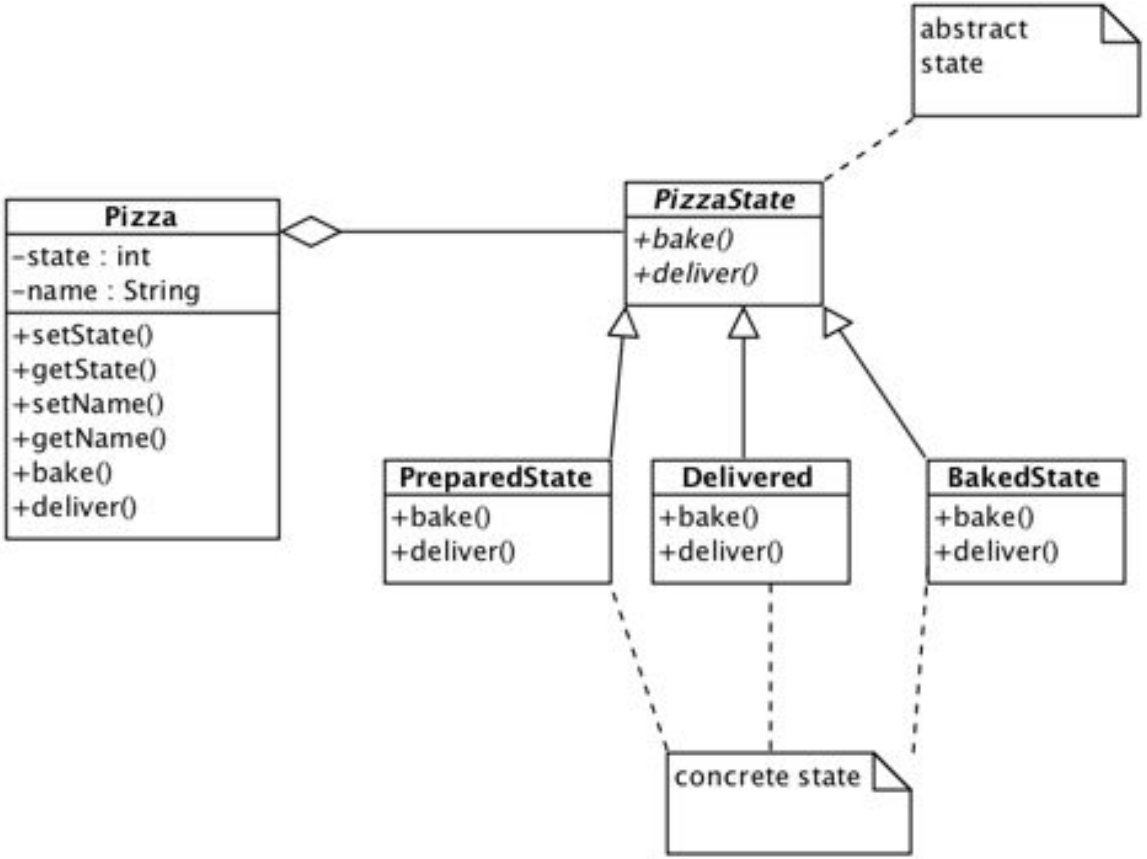


Another (C++): http://www.ai-junkie.com/architecture/state_driven/tut_state1.html

State Pattern in UML



State Pattern



Singleton

Problem:

- Only one instance of an object is need
  - Logging
  - GameManager
  - API

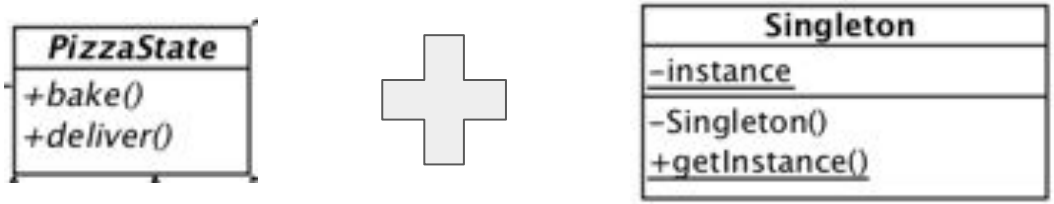
Singleton


```
public class ClassicSingleton {
  private static ClassicSingleton instance = null;
  private ClassicSingleton() {
    // Exists only to defeat instantiation.
  }
  public static ClassicSingleton getInstance() {
    if(instance == null) {
      instance = new ClassicSingleton();
    }
    return instance;
  }
}
```

```
public static void main(String[] args) {
      ClassicSingleton tmp = ClassicSingleton.getInstance( );

    //ClassicSingleton tmp = new ClassicSingleton( ); //not possible
}
```

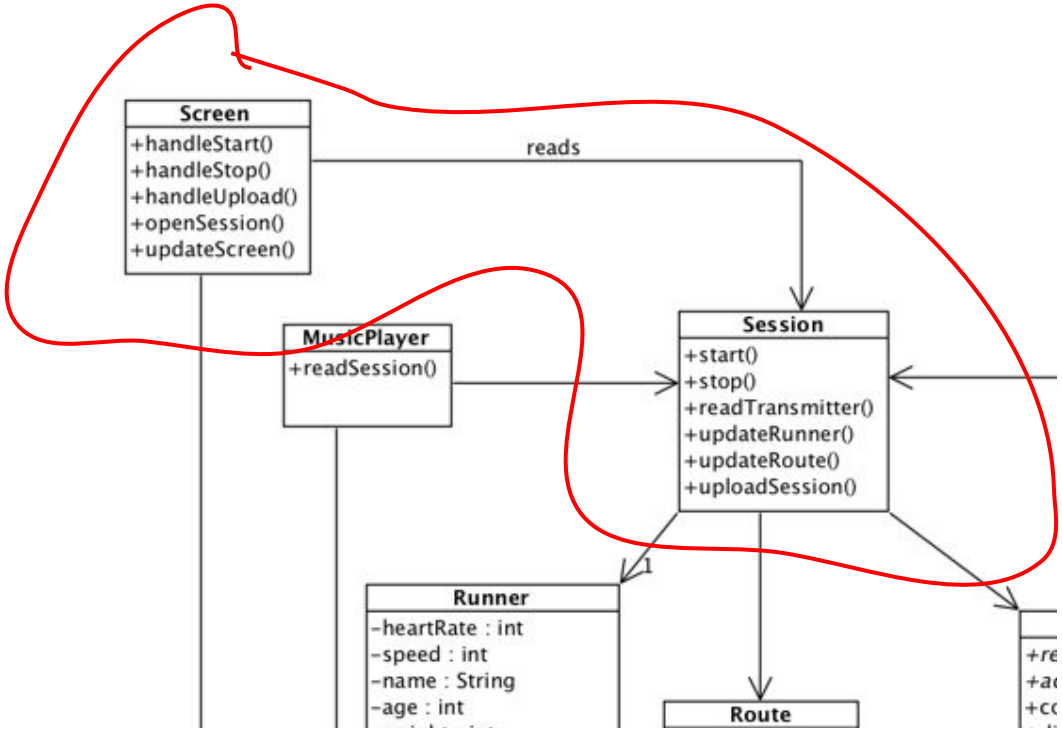
Singleton [creational]

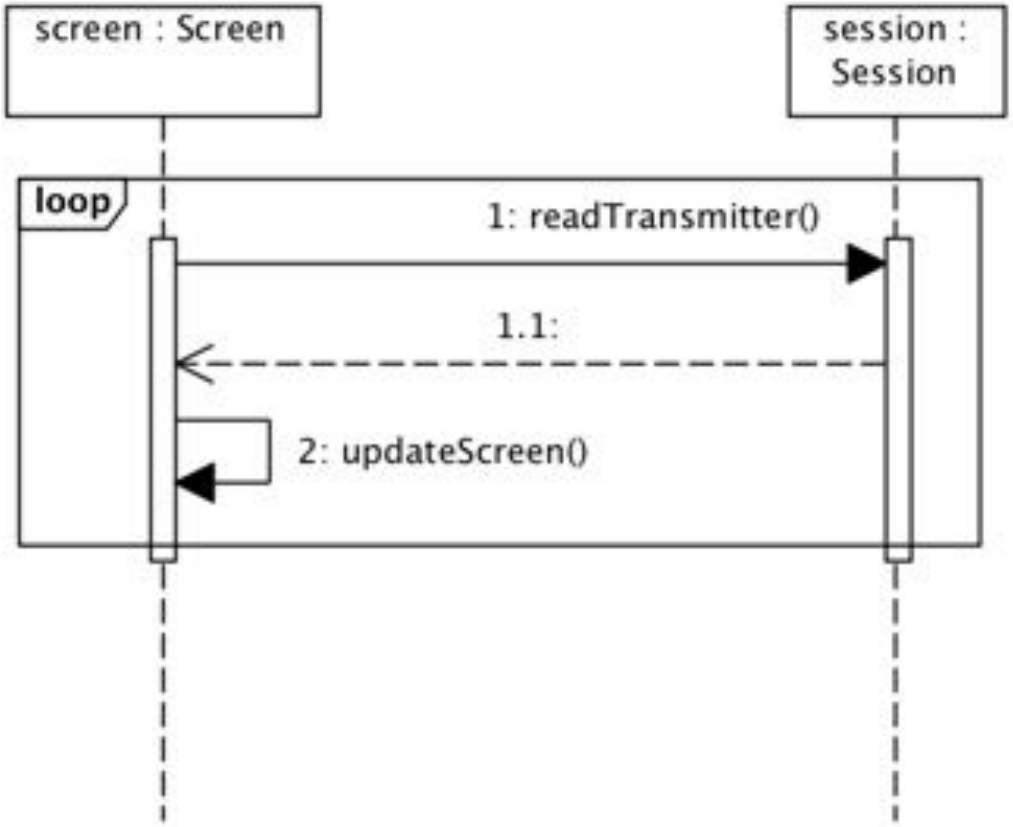
Abstract state often implemented as a singleton



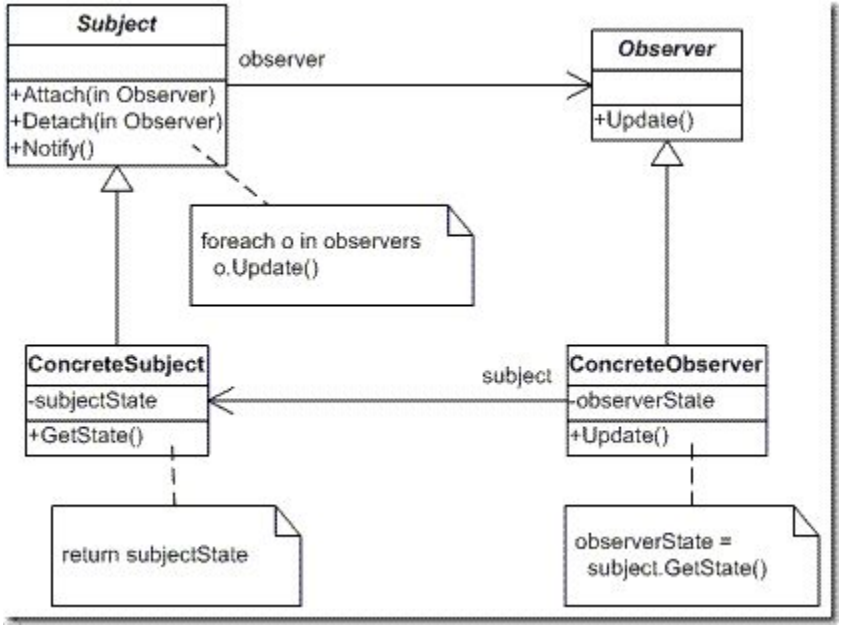
Challenge

What are possible issues here?




screen : Screen
session : Session
loop
1: readTransmitter()
1.1:
2: updateScreen()

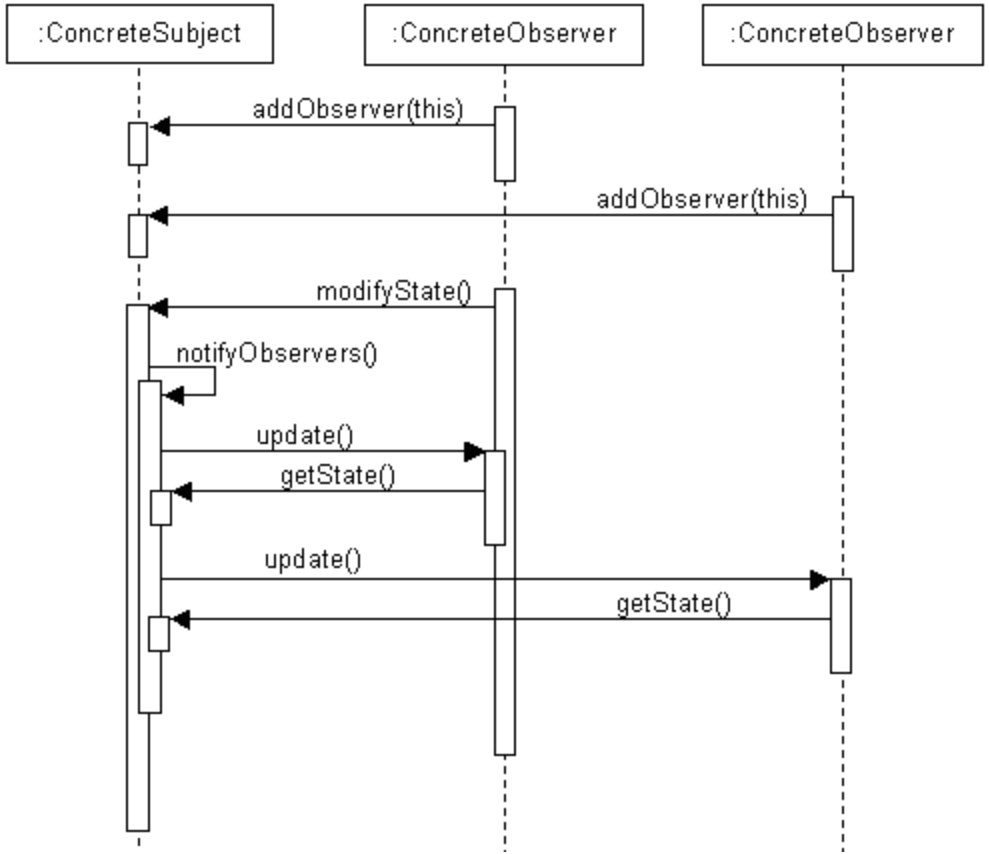
Solution - Observer Pattern



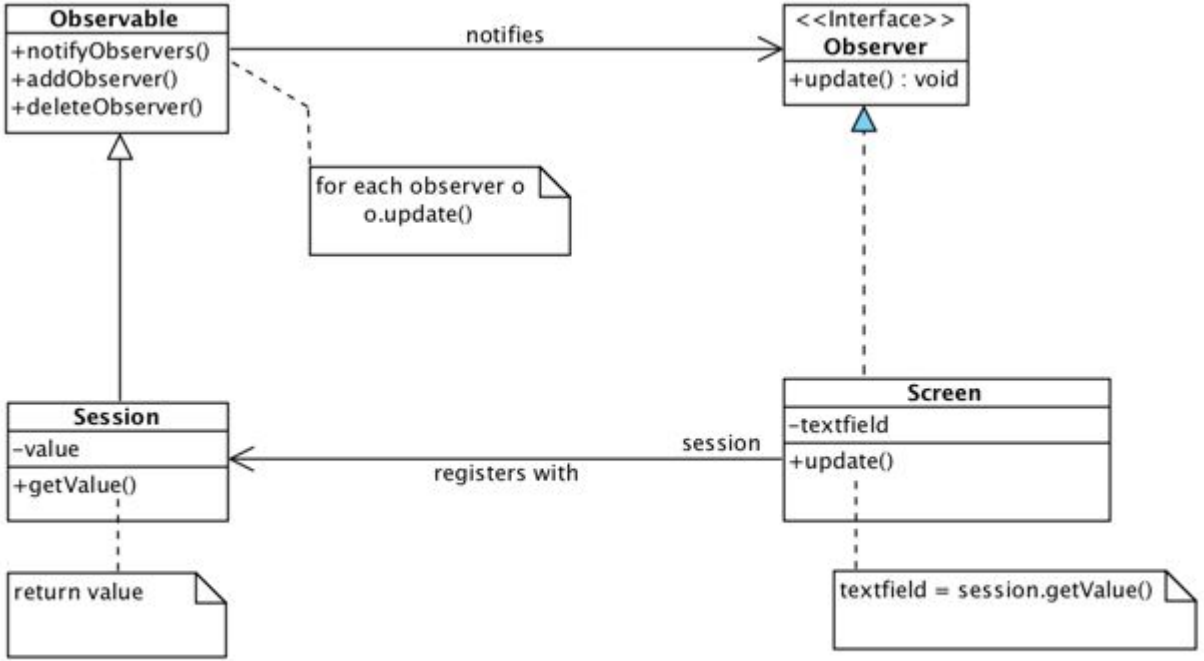
Solution - Observer Pattern

Problem:

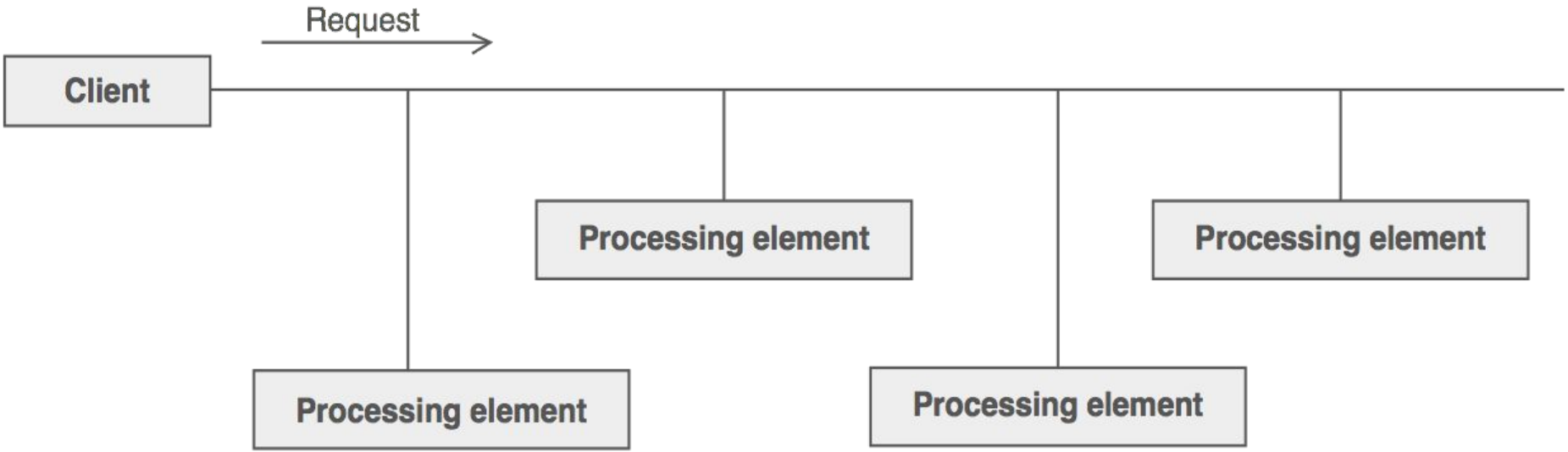
- Calling object has to wait for reply
- Multiple objects need information of one object after update


:ConcreteSubject
:ConcreteObserver
:ConcreteObserver
addObserver(this)
addObserver(this)
modifyState()
notifyObservers()
update()
getState()
update()
getState()

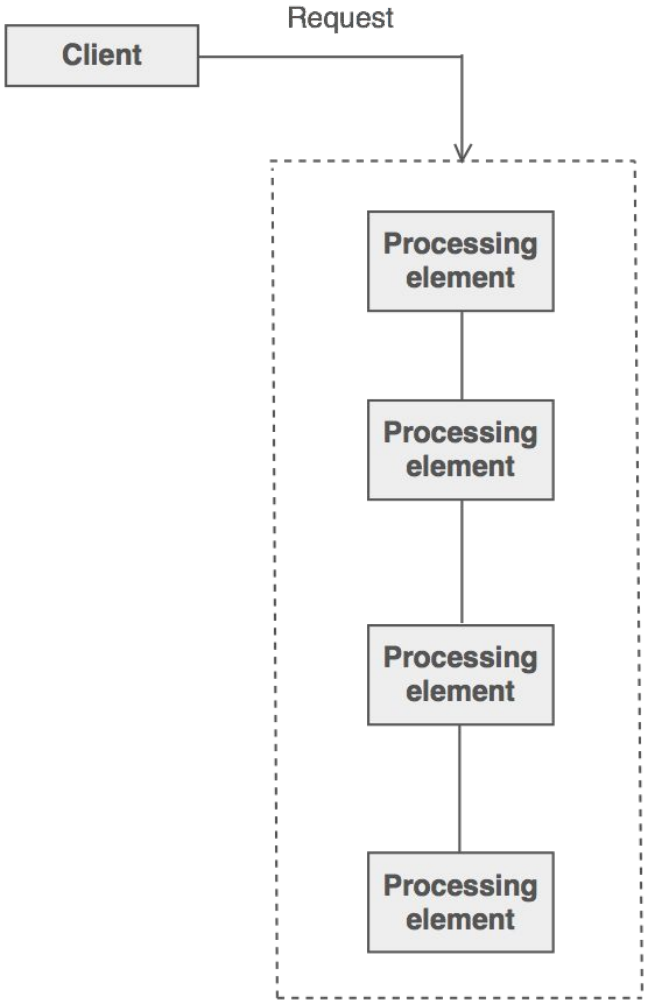
Observer Pattern [behavioral]



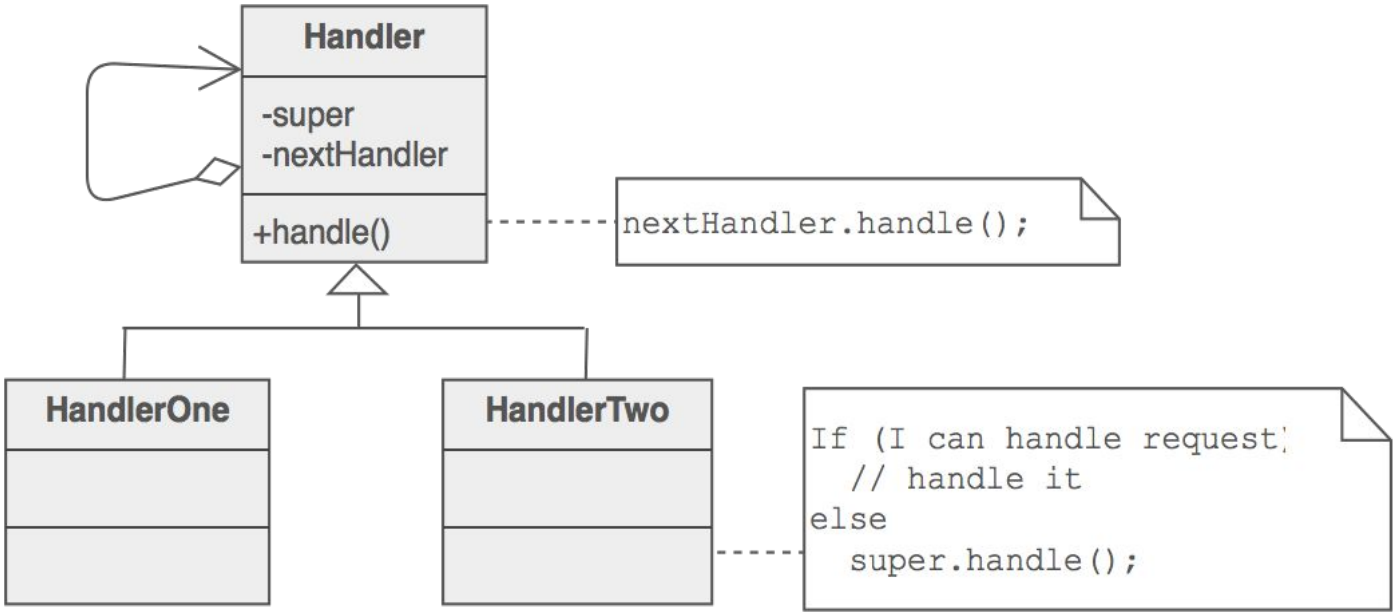
Chain of Responsibility 1/3



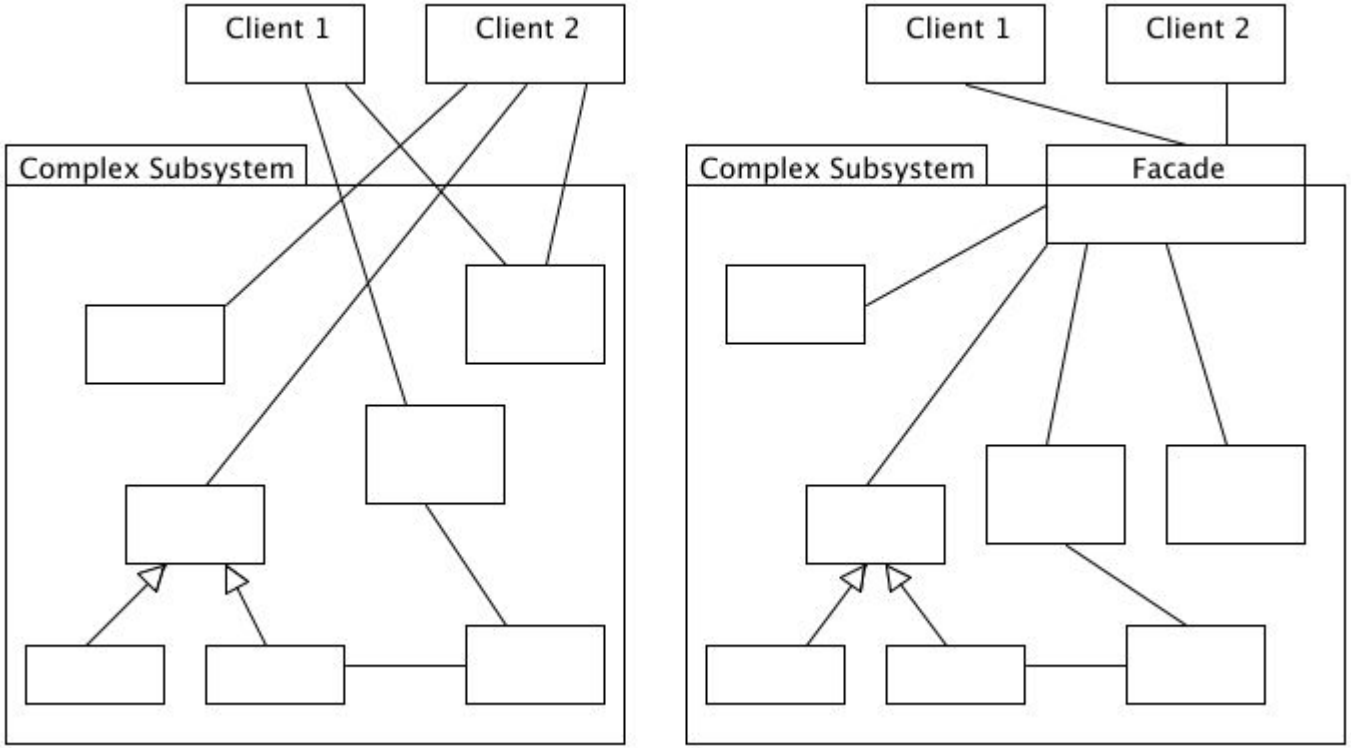
Chain of Responsibility 2/3



Chain of Responsibility 3/3 [behavioural]



Facade [structural]

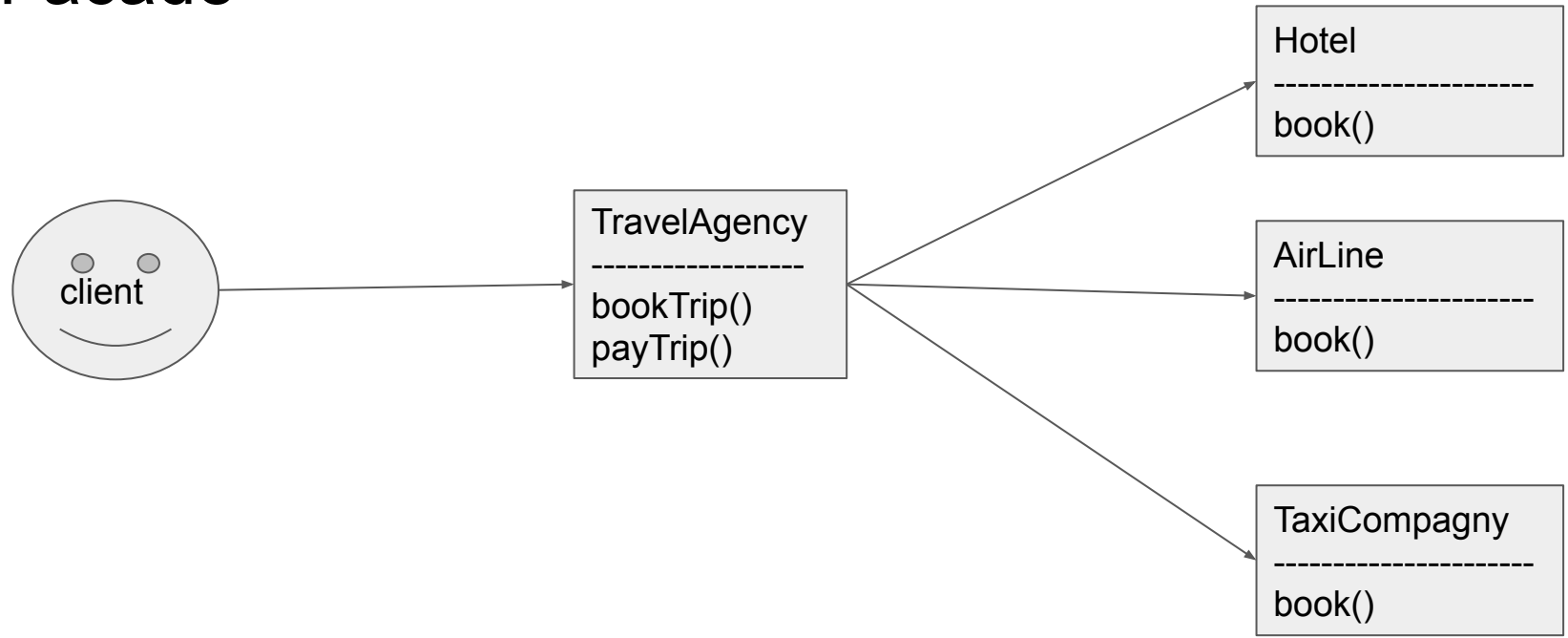


Facade [structural]

Problem:

- Make a subsystem easier to use
- Minimize dependencies on a subsystem

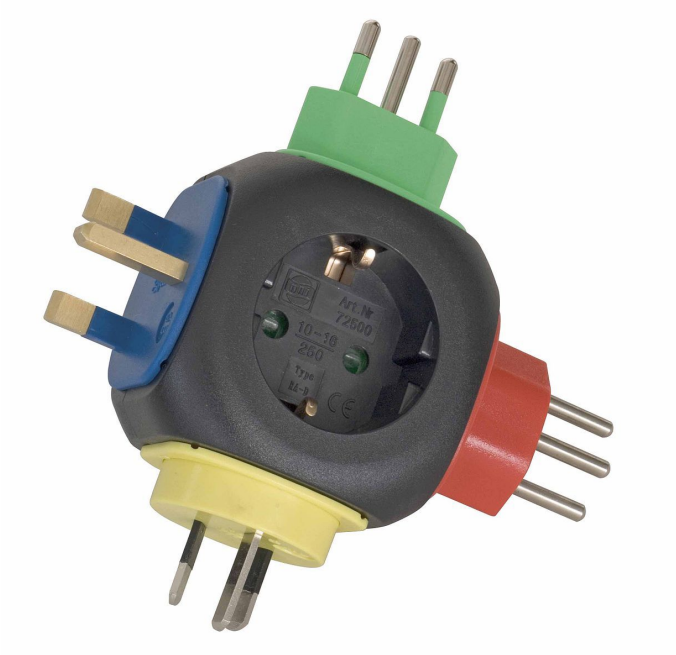
Facade


client
TravelAgency
bookTrip()
payTrip()
Hotel
book()
AirLine
book()
TaxiCompagny
book()

Adapter [structural]

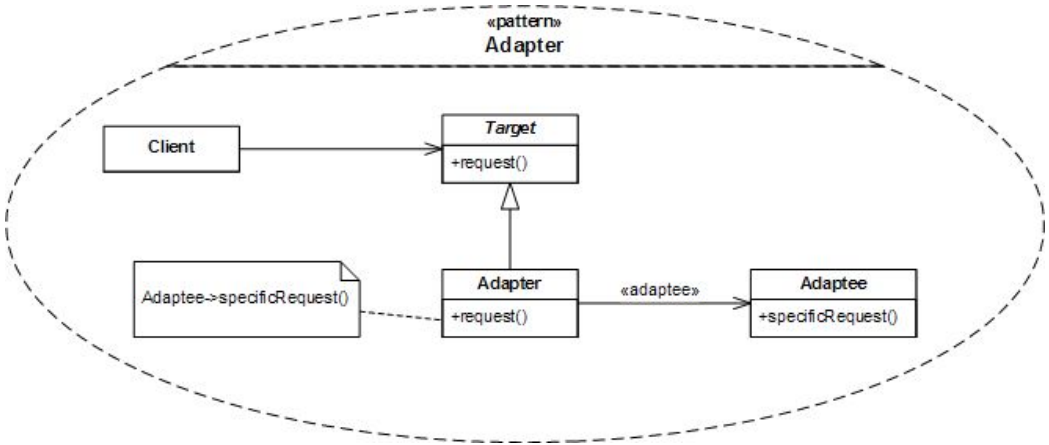
Problem:

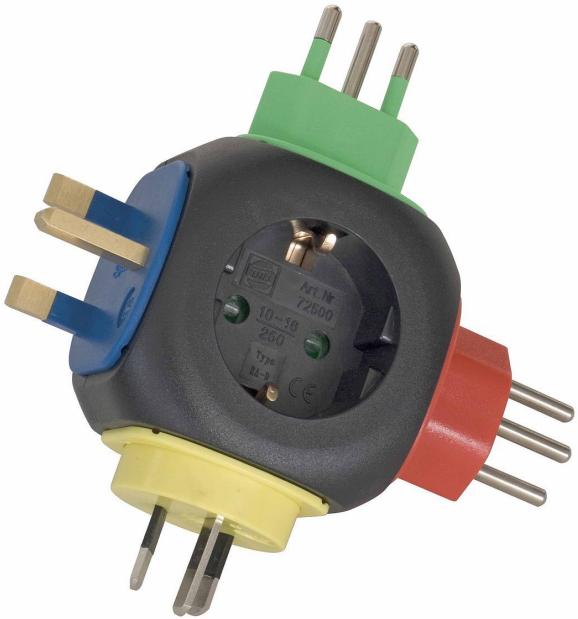
- Fit a class to a required interface



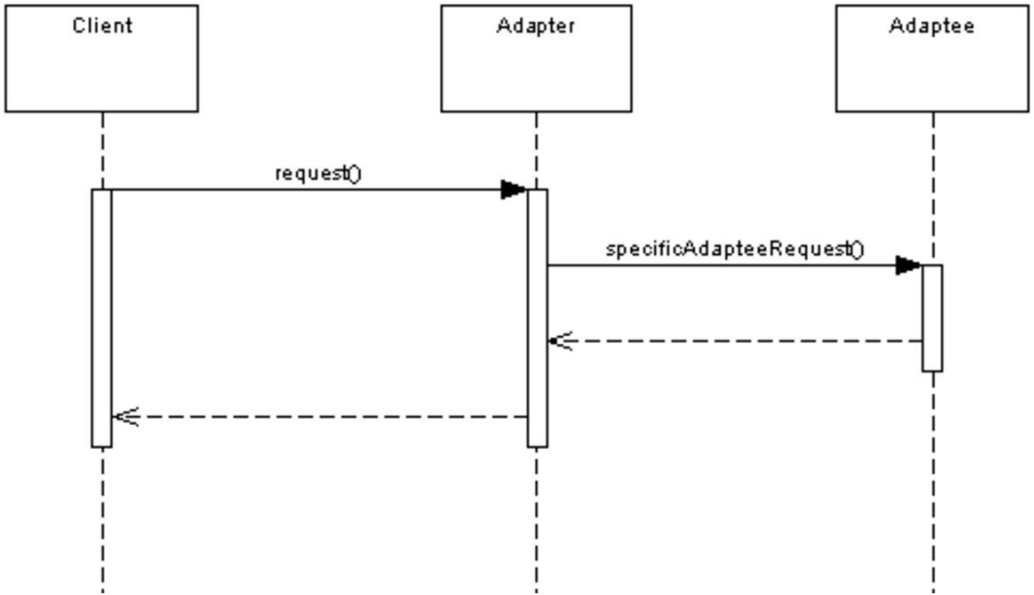
By Cephira [GFDL (http://www.gnu.org/copyleft/fdl.html), CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0) or CC BY-SA 3.0 de (https://creativecommons.org/licenses/by-sa/3.0/de/deed.en)], from Wikimedia Commons

Adapter [structural]





Adapter



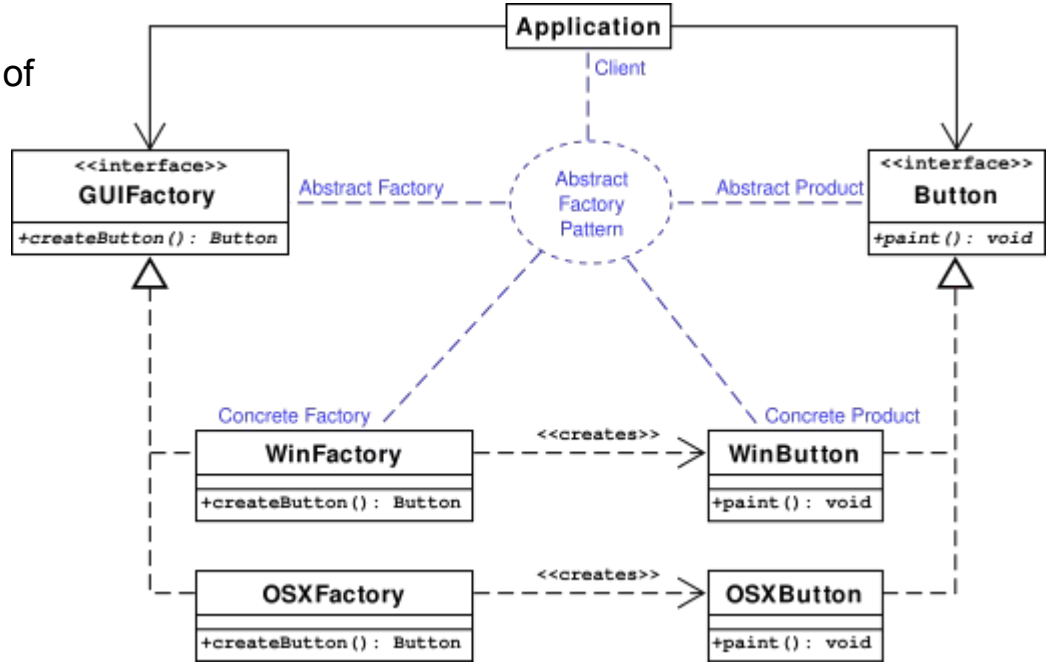
Adapter

Usage : for example (Eclipse) plugins

Abstract Factory [creational]

Problem:

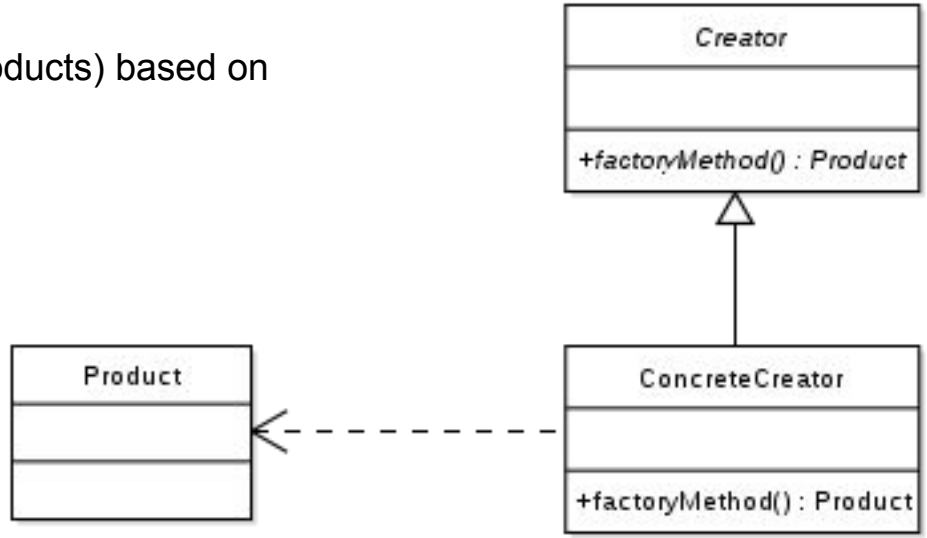
- Application should be independent of creation of its own objects



Factory Method [creational]

Problem:

- Need to create different objects (products) based on situation

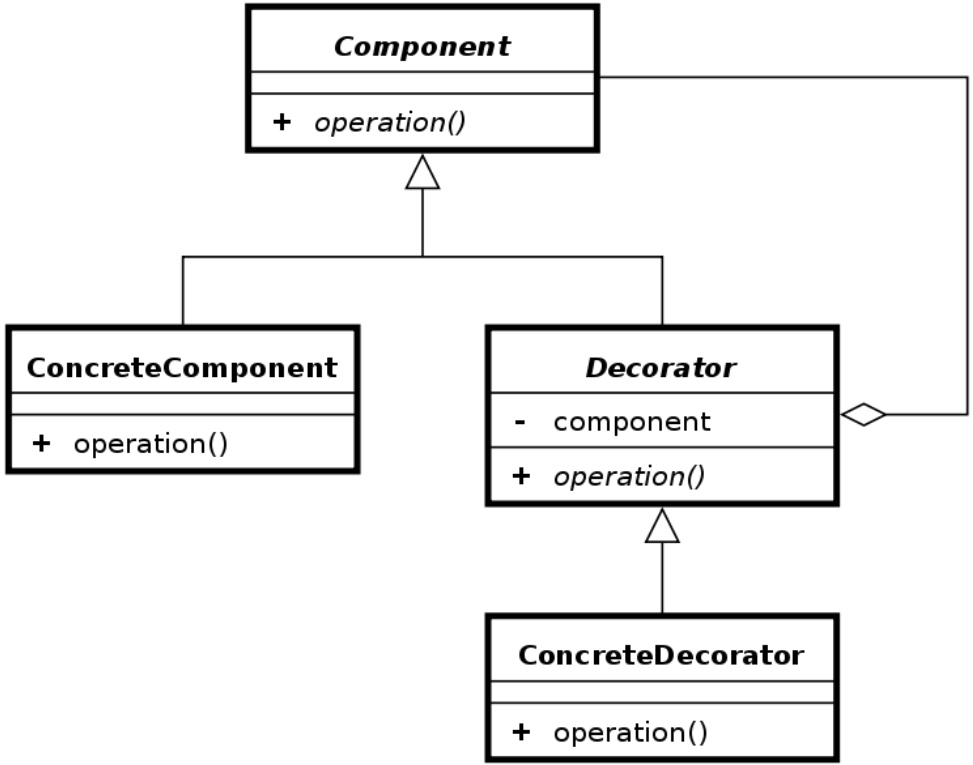


Real life example: hotel front desk

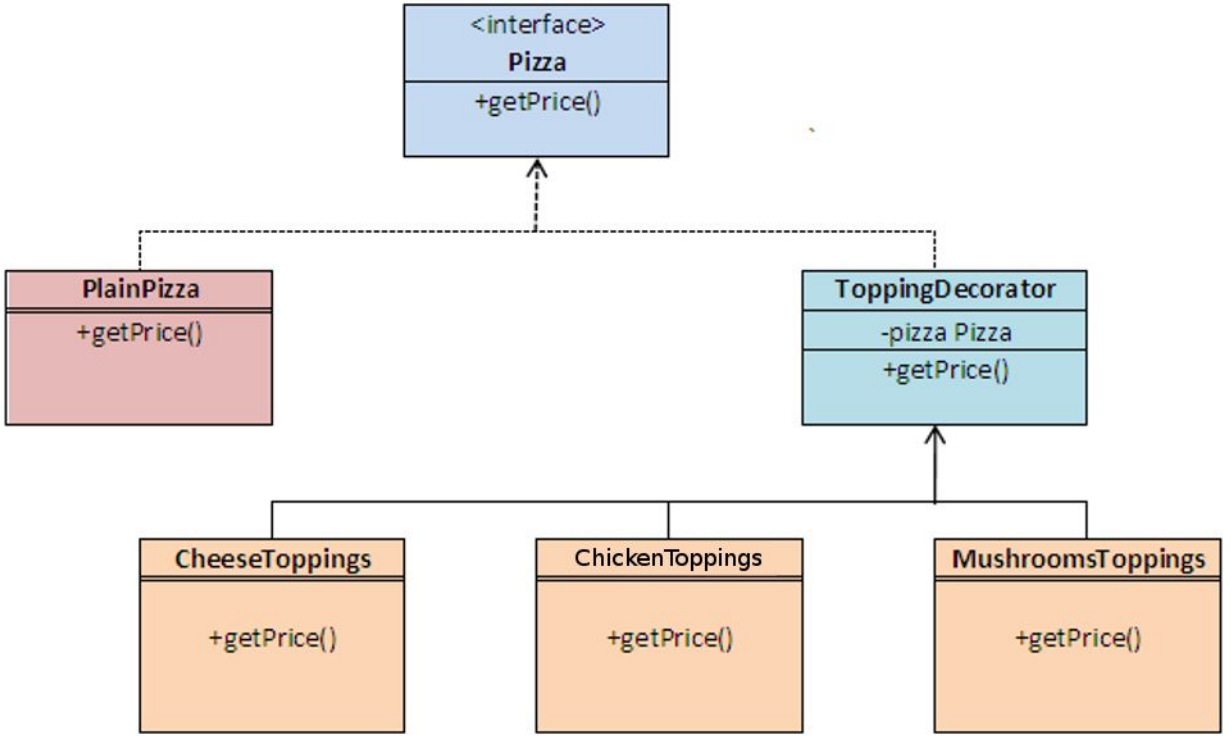
Decorator [structural]

Problem:

- Want to add responsibility at runtime



Decorator


<interface>
Pizza
+getPrice()
PlainPizza
+getPrice()
ToppingDecorator
-pizza Pizza
+getPrice()
CheeseToppings
+getPrice()
ChickenToppings
+getPrice()
MushroomsToppings
+getPrice()

Question

What patterns could be used in the running health app?

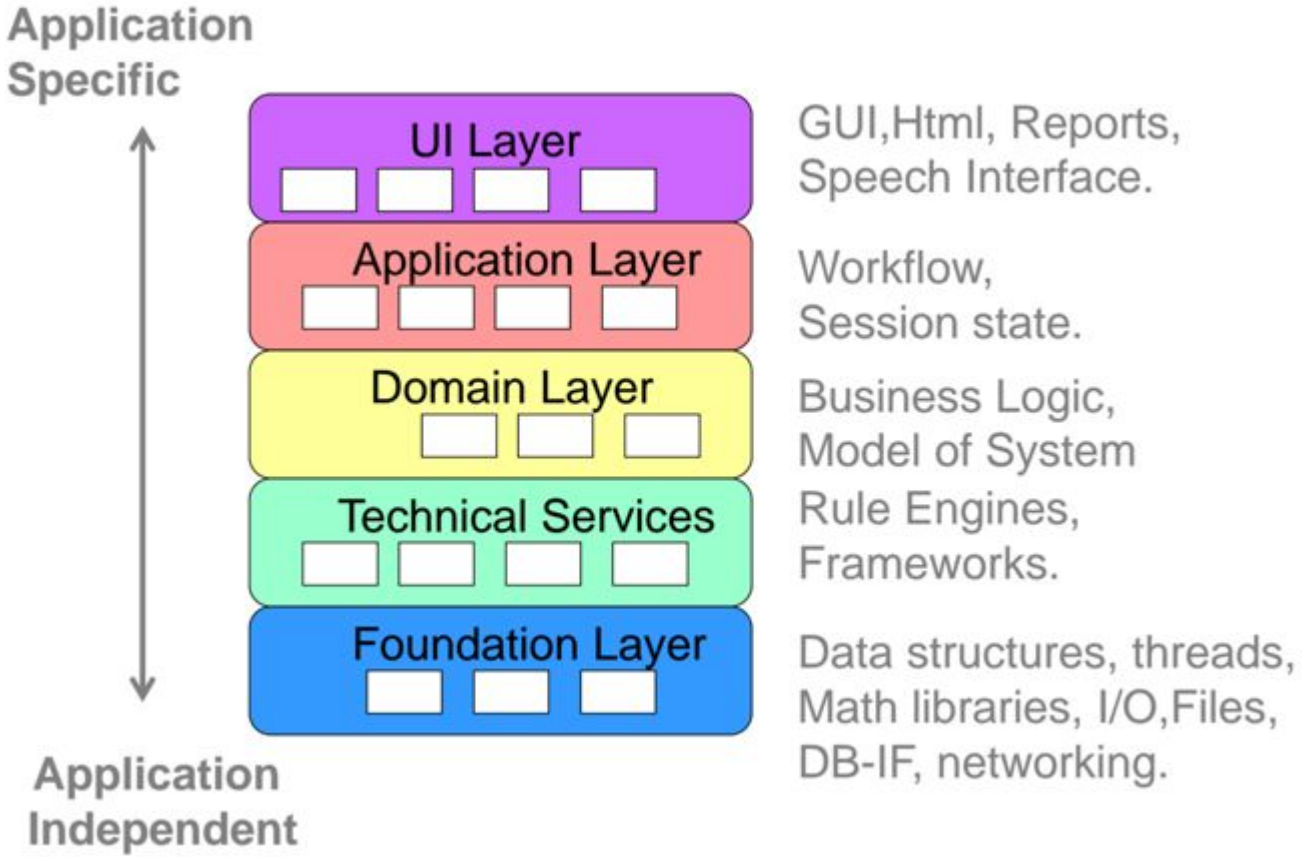
Architectural Patterns

Architectural Patterns

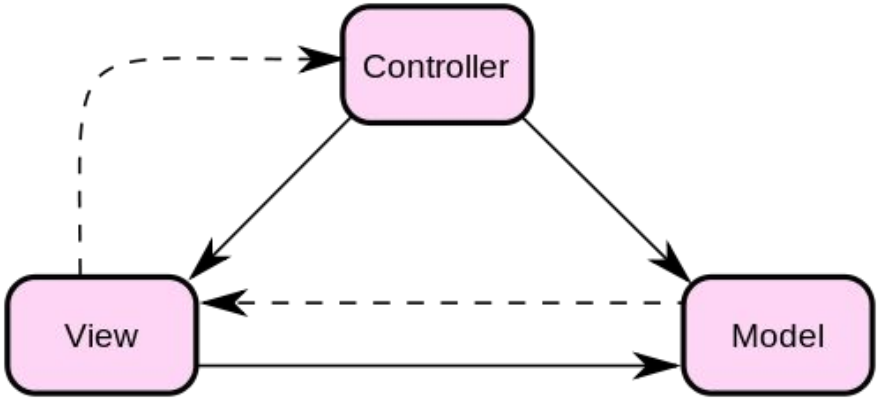
Layered Architecture

Model View Controller Architecture- MVC

Layered

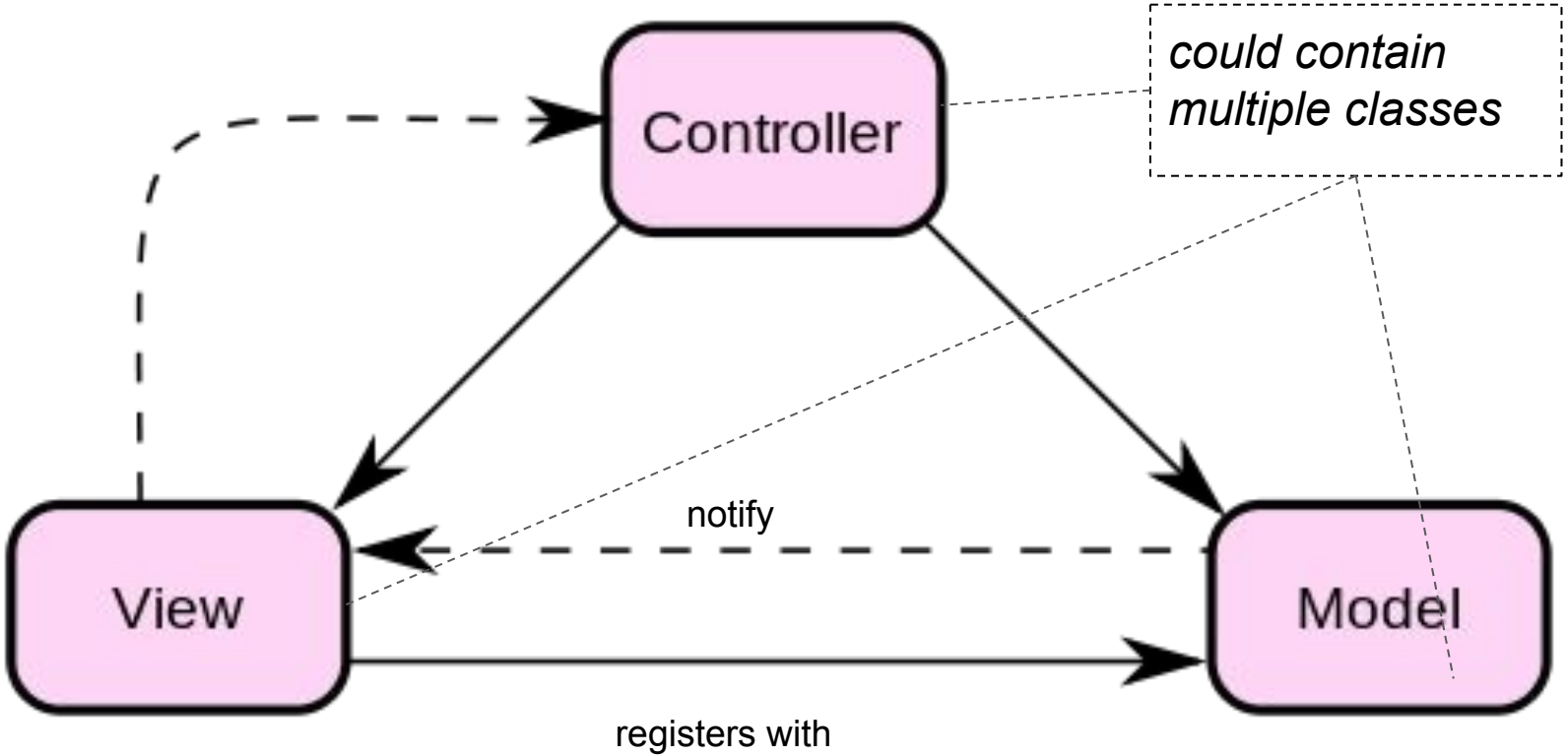

Application Specific
UI Layer
Application Layer
Domain Layer
Technical Services
Foundation Layer
Application Independent
GUI,Html, Reports, Speech Interface.
Workflow, Session state.
Business Logic, Model of System
Rule Engines, Frameworks.
Data structures, threads, Math libraries, I/O,Files, DB-IF, networking.

MVC

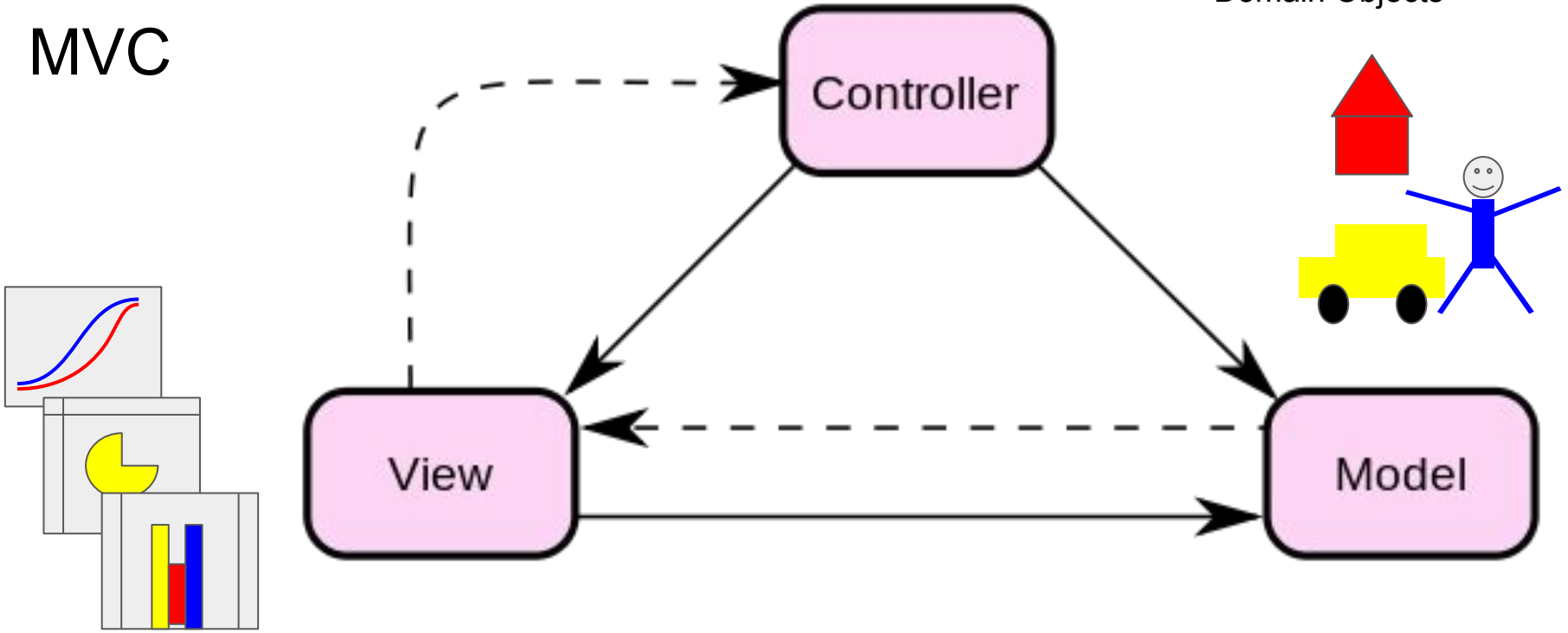


Internal presentation of information is separated from how it is presented

MVC

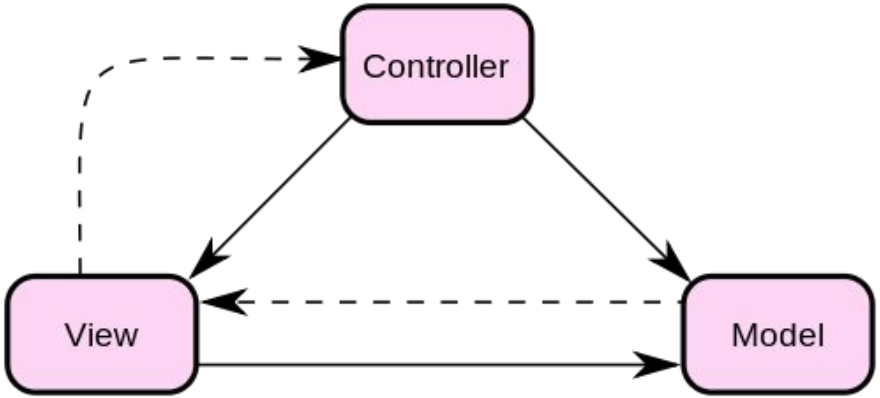

Controller
could contain
multiple classes
notify
View
Model
registers with

MVC



Multiple views possible

MVC



Simultaneous Development : for example, views can be developped in parallel, independant of an existing model implementation (*a dummy model can be used if needed*).

Code Reuse : for example, using the same (kind of) view in a different application

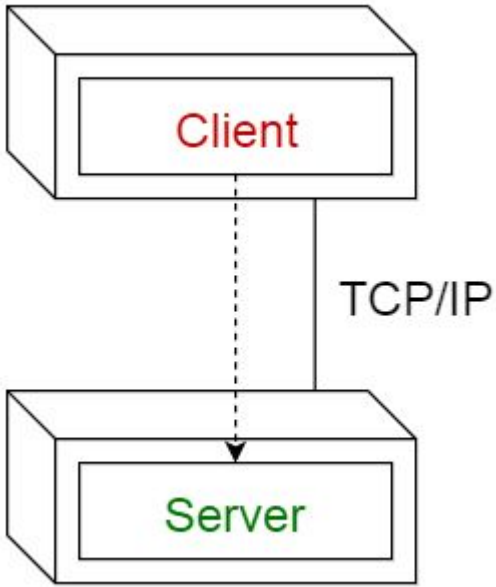
Example: http://blog.nuclex-games.com/2010/09/mvc-in-games/

MVC

Which behavioral pattern can help to implement MVC?

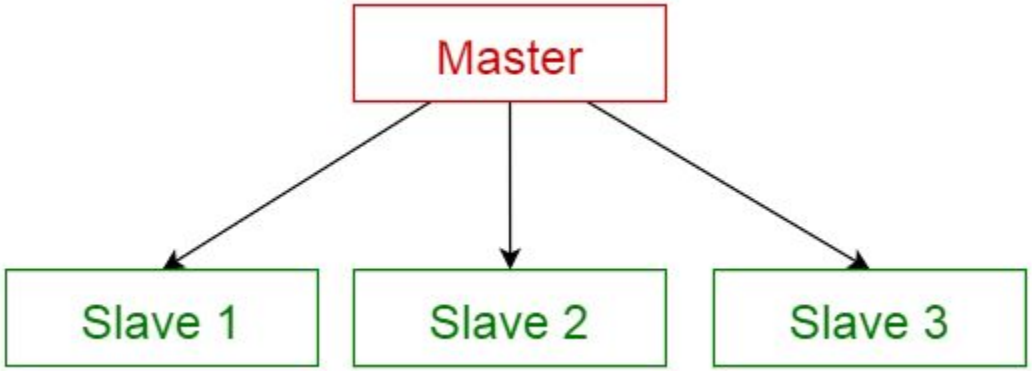
Client Server

For example:
Online email
Online document editor



Master-Slave

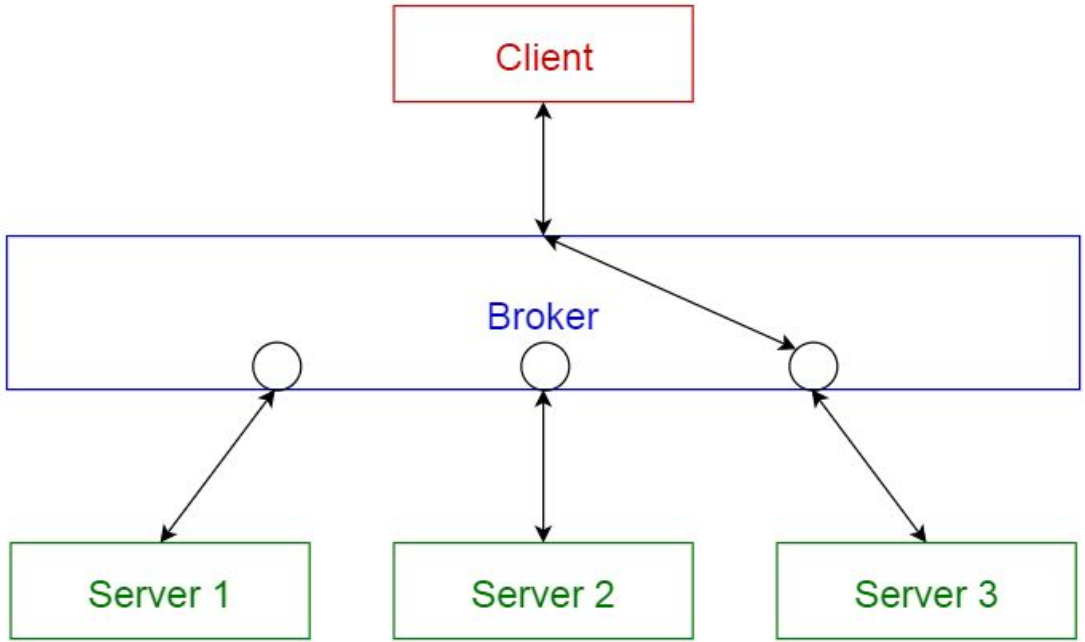
For example:
Database Replication
Master-Slave drives



Broker

Subscribe to a ‘channel’ of interest

- Internet Of Things



More Patterns

See:

- Larman
- Martin (GUL: DesignPrinciplesAndPatterns.pdf) ← mandatory reading!

and the internet