

Homework #2 is  
posted

# Model Driven Architecture

## MDA

- \* multi-year architecture paradigm
- \* We want to separate
  - \* platform independent "aspects" of the system

FROM

- \* platform specific "aspects" of the system

Idea

to create a platform  
independent model  
(PIM) that will be  
"stable"  
will not change for  
many years/versions.

PIM: Meta Model

## Platform

- \* operating system
- \* programming language
- \* data base
- + network .
- + hardware .
- + data structures
- \* algorithms

a . .

a ..

## Data

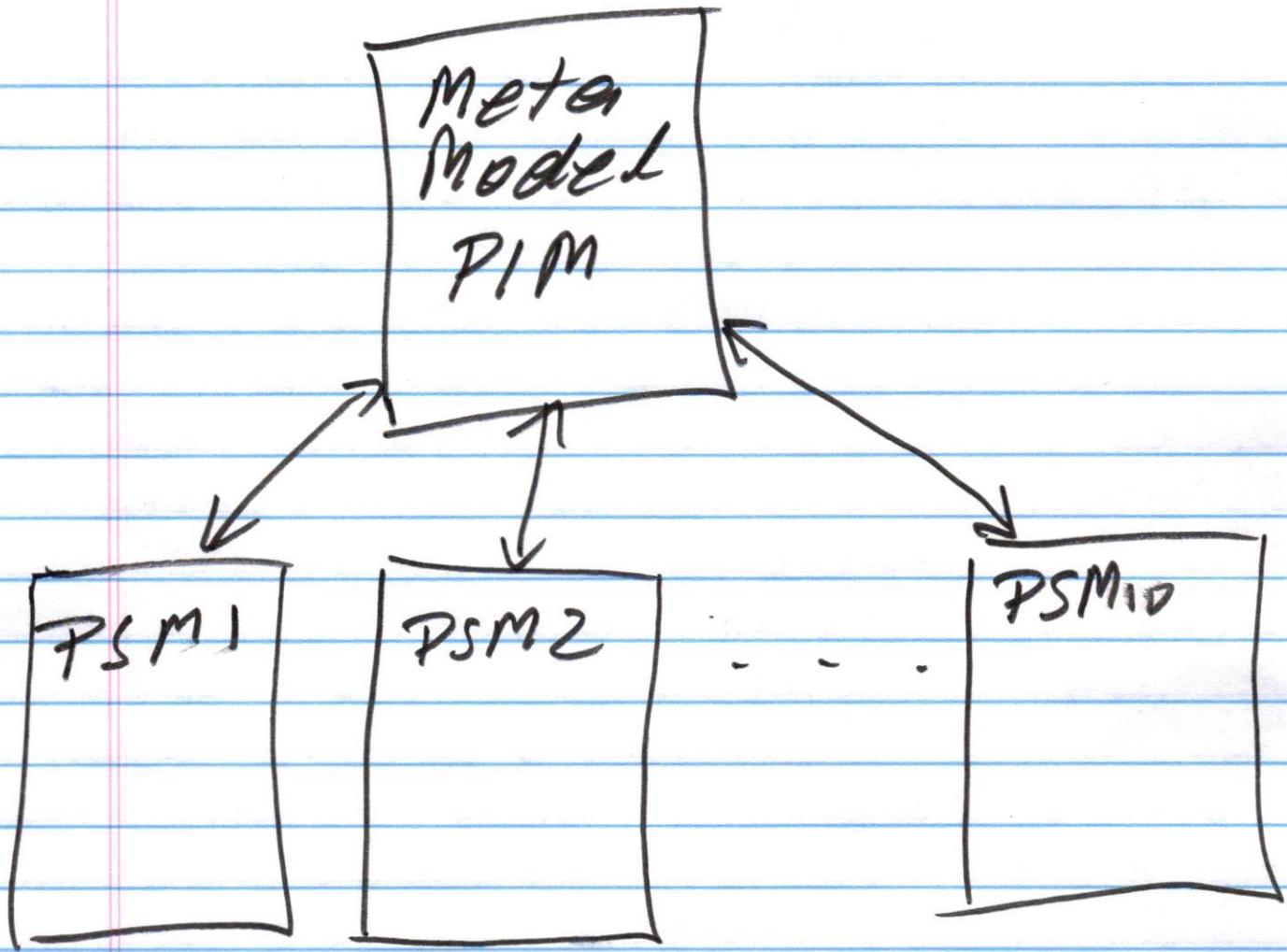
Data-oriented meta models



data base systems

Behavior

We want to concentrate  
on meta models that  
capture "meta" behavior  
of a family of systems



PSM: Platform specific Model

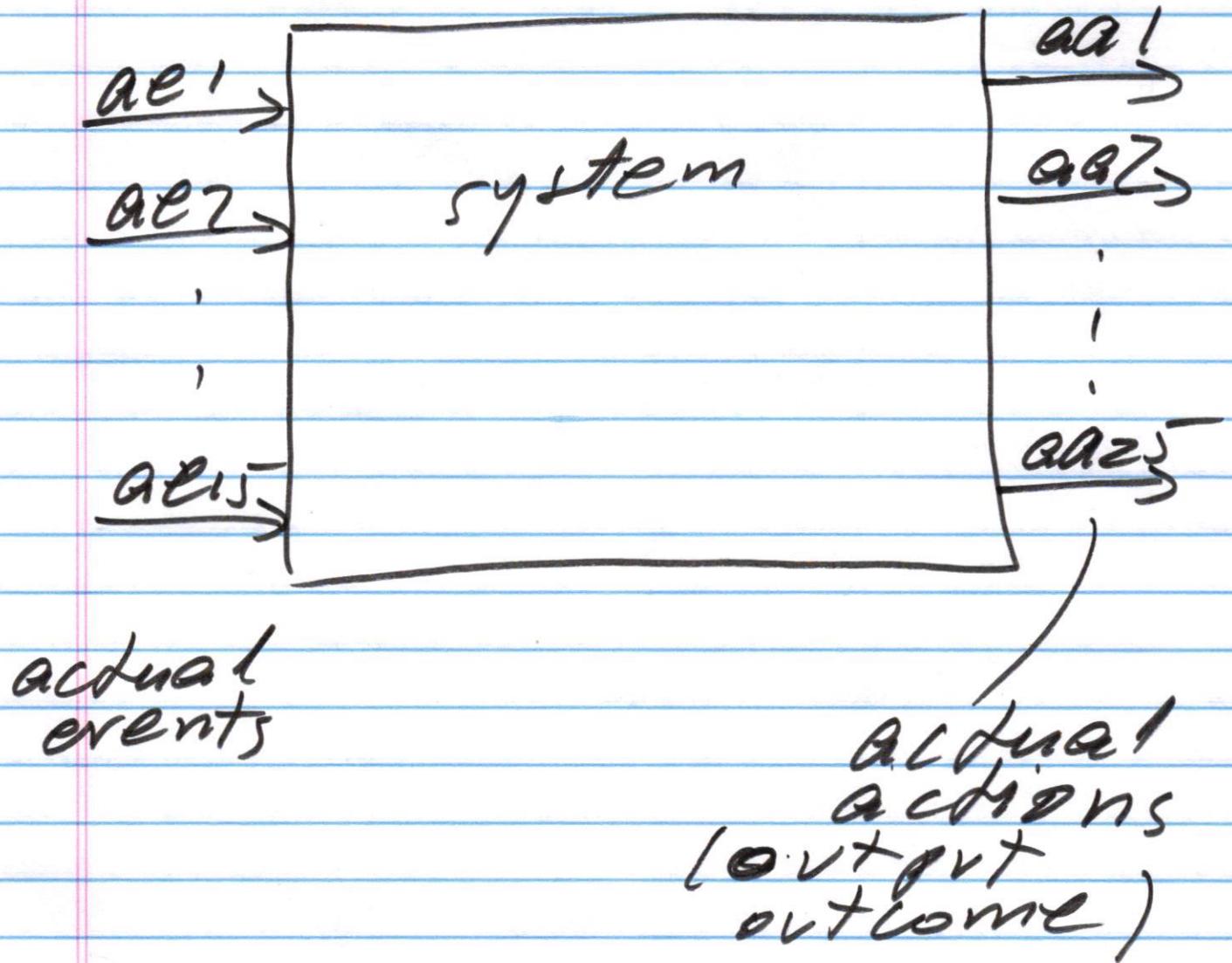
## Meta Model related to behavior

---

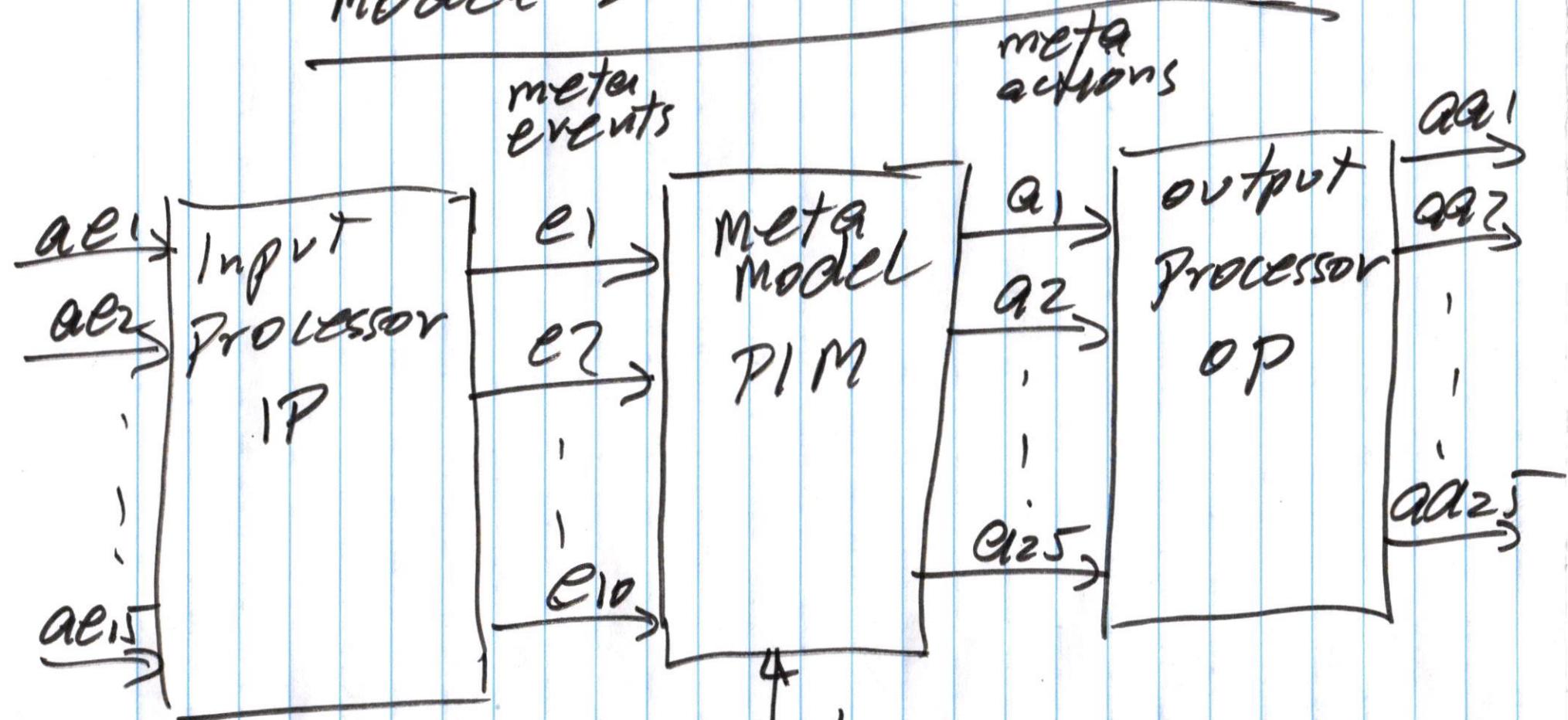
- \* should capture the behavior of a family of systems that is "platform" independent
- \* We want to separate the behavior for different OS, DB, network..
- \* should be executable
  - \* we should be able to test it
    - ↳ validate/verify the meta model

# Model-Driven Architecture

event based systems.



# Model driven Architecture



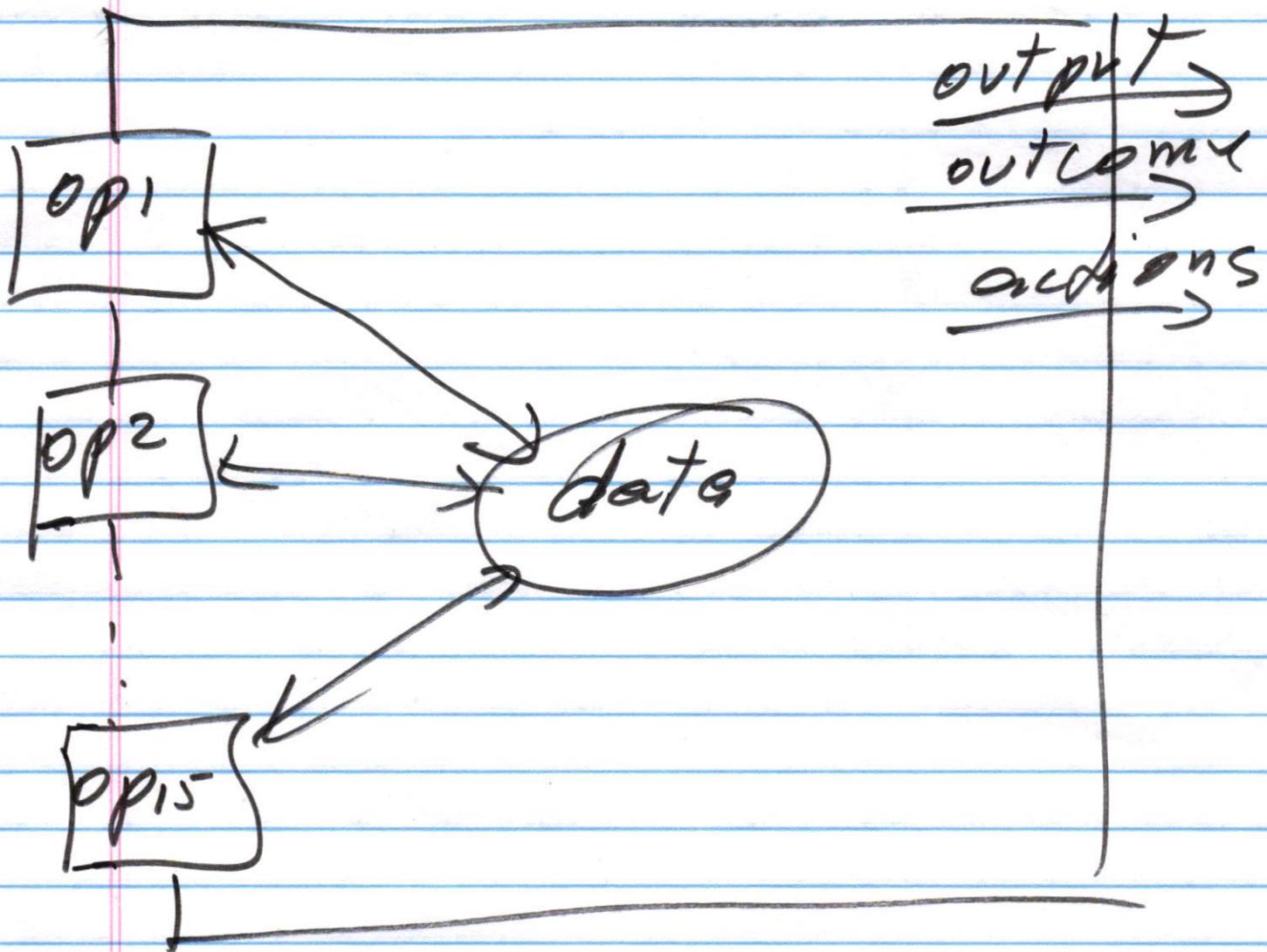
actual events

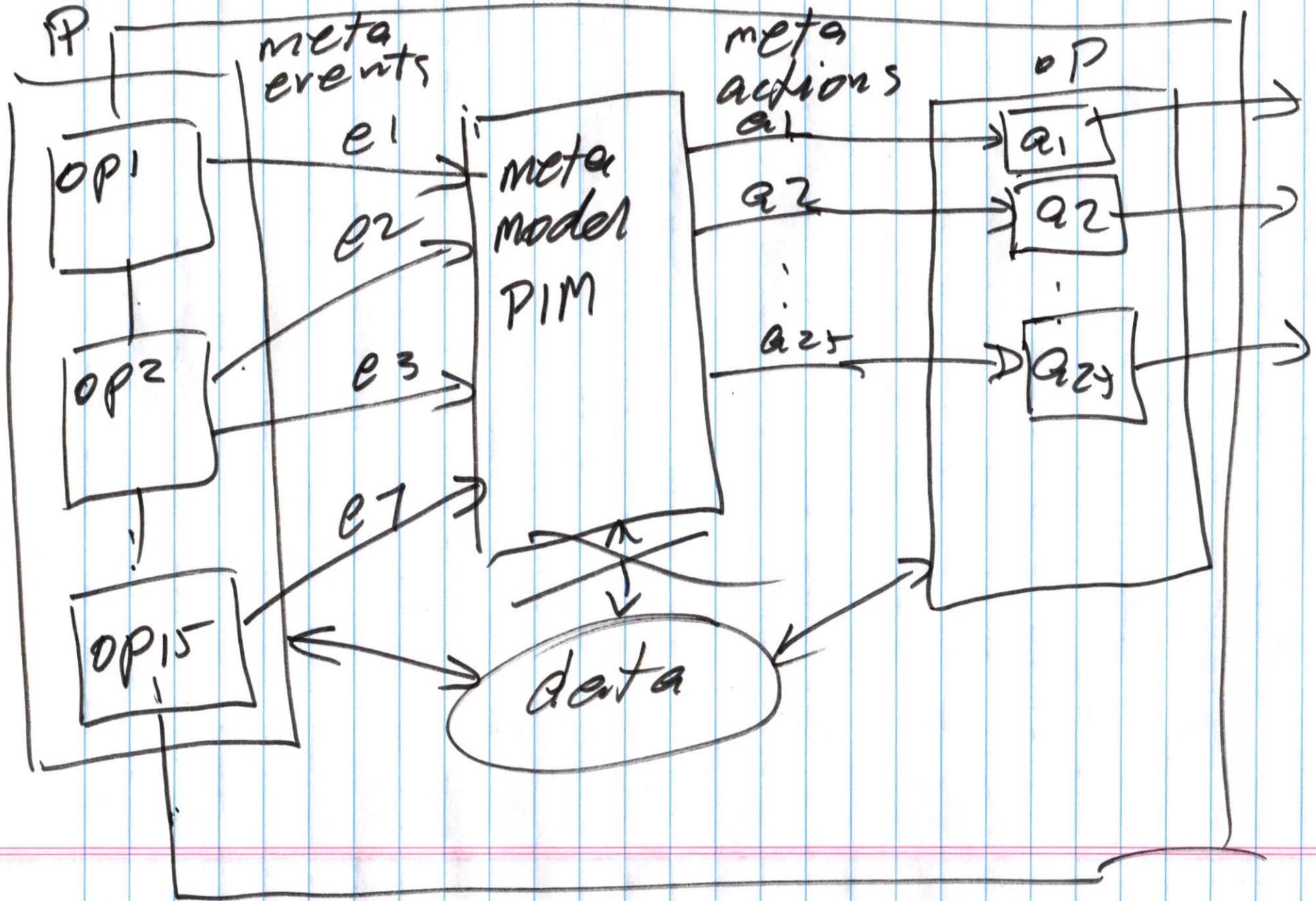
$$PSM = \begin{cases} IP \\ OP \end{cases}$$

captures  
the platform  
independent  
"behavior"  
of a family  
of systems.

a component  
a set of operations.

$OP_1, OP_2, \dots, OP_{15}$





meta  
model  
specification

code  
generator  
Java, C++

source  
code

meta  
events

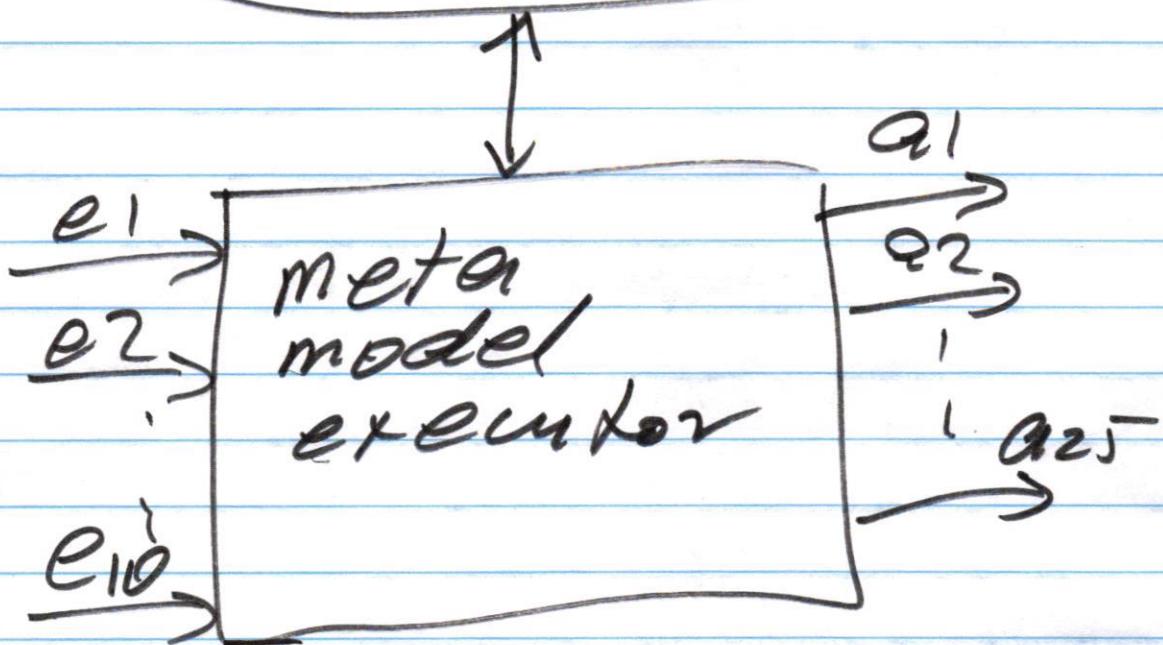
$e_1 \rightarrow$   
 $e_2 \rightarrow$   
 $\vdots$   
 $e_{15} \rightarrow$

executable  
binary et  
meta model

meta  
actions

$a_1 \rightarrow$   
 $a_2 \rightarrow$   
 $\vdots$   
 $a_{25} \rightarrow$

meta  
model  
specification



PIM

meta events

$e_1 \rightarrow$

$e_2 \rightarrow$

.

.

$e_{10} \rightarrow$

Meta  
model  
behavior

executable

meta  
actions

$a_1 \rightarrow$

$a_2 \rightarrow$

.

.

$a_{25} \rightarrow$

$e_i$ : meta event

$a_i$ : meta action

## Modeling languages

that can capture meta behavior.

\* FSM: Finite state Machine

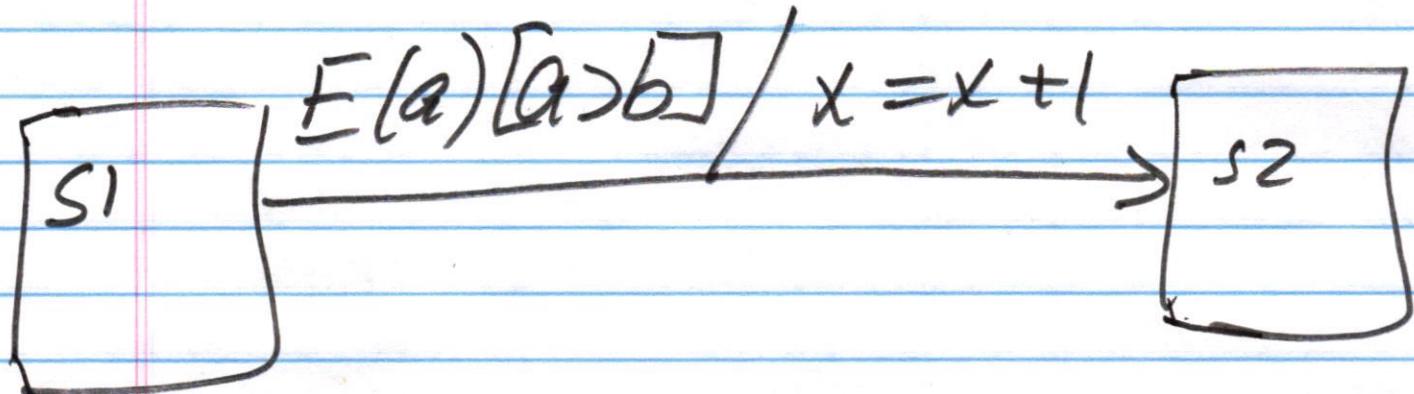
\* EFSM: Extended Finite state Machine

\* VFSM: Virtual Finite state Machine

A - -  
K - - -

# EFSM

- \* states
- ↳ transitions
- \* conditions
- \* variables / data
- \* actual events
- \* actual actions



EFSM is platform dependent

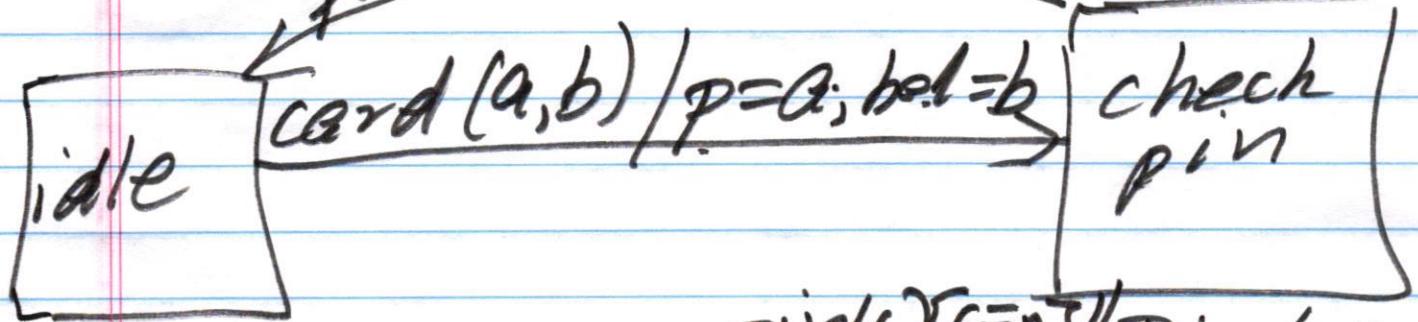
# ATM

card(pin, balance)  
pin(pin-id).

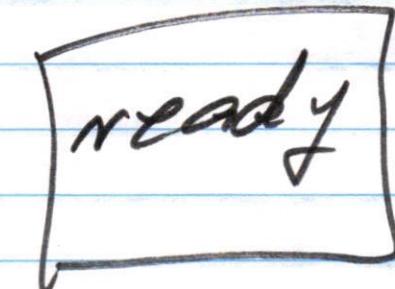
## version I

card(int pin, int balance)  
pin(int pin-id)

~~pin(c)[c=p] / Eject(card)~~



pin, balance  
 $a, b, c, p, bal$   
are integer.



## version II

card(float balance, string pin)  
pin (string pin-id)

pin, balance are different  
types

## version III

card(string pin, int balance,  
string user-id)  
pin (string pin-id)

different # of parameters

# VFSM: Virtual Finite state Machine

- \* states
- \* transitions
  - \* virtual events
  - \* virtual actions
  - \* conditions.

---

virtual event: name  
Boolean variable

virtual action: name

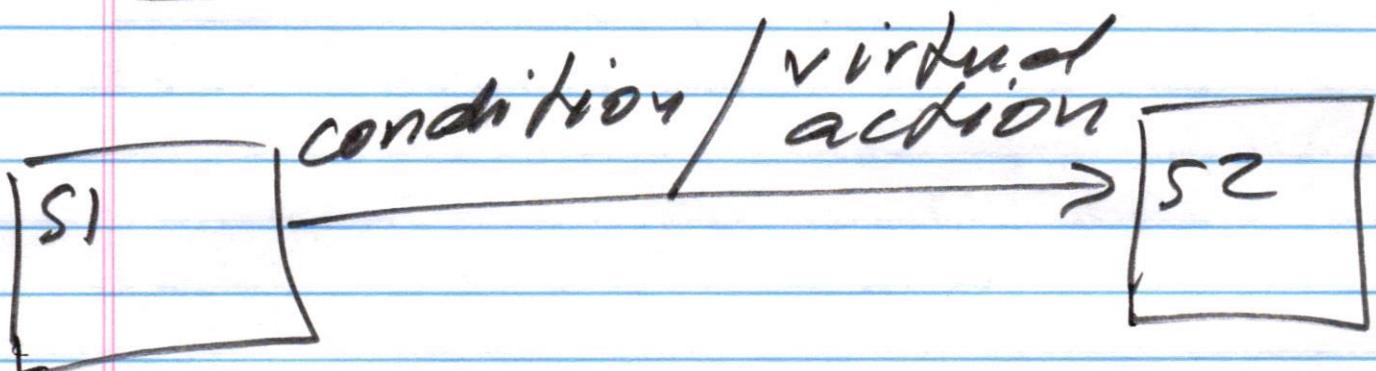
---

ATM } card (int pin, int bal)  
EFIM } card (float bal, string pin)  
card (bal, pin, p2, p3, p4)

---

VFSM: card

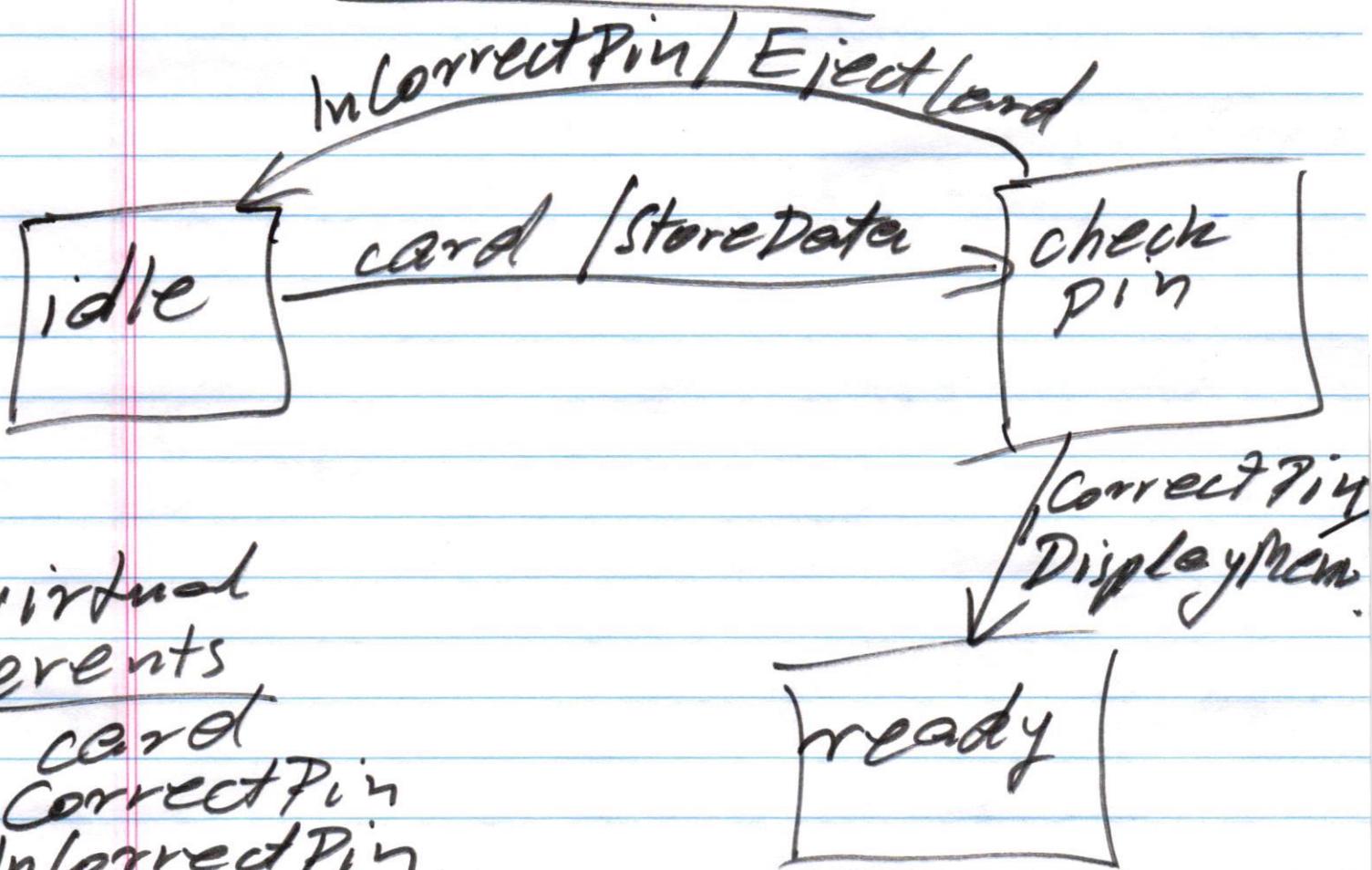
## VFSM



Condition: Boolean Expression

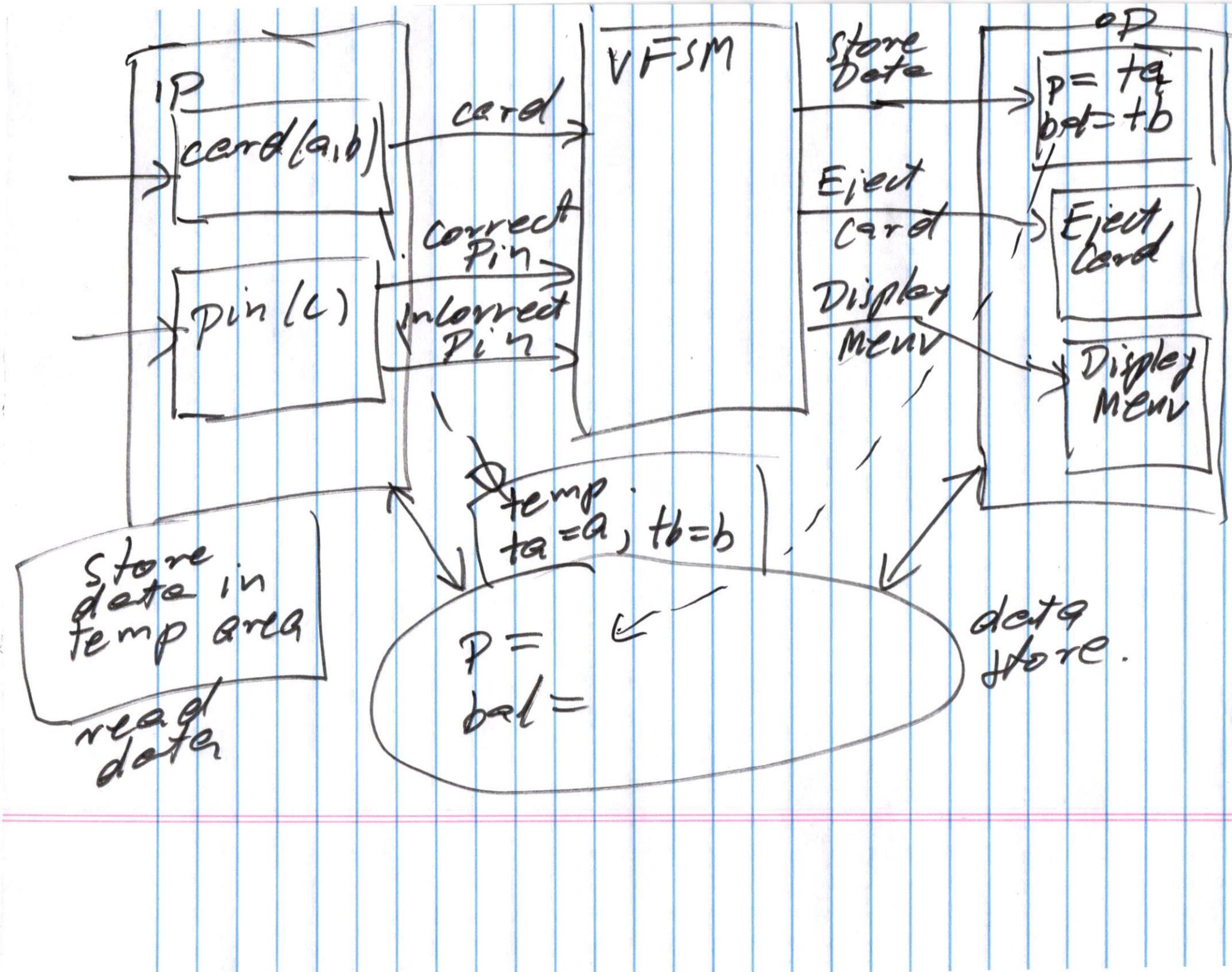
1. model is in  $S_1$
2. virtual event  $e$  is invoked  
     $e$ : Boolean variable  
    "True" value is assigned to  $e$
3. Condition is evaluated  
     $(e \text{ or } e^{\dagger})$   
    evaluates to true.
4. Transition from  $S_1$  to  $S_2$   
    virtual action is invoked

# VFSM



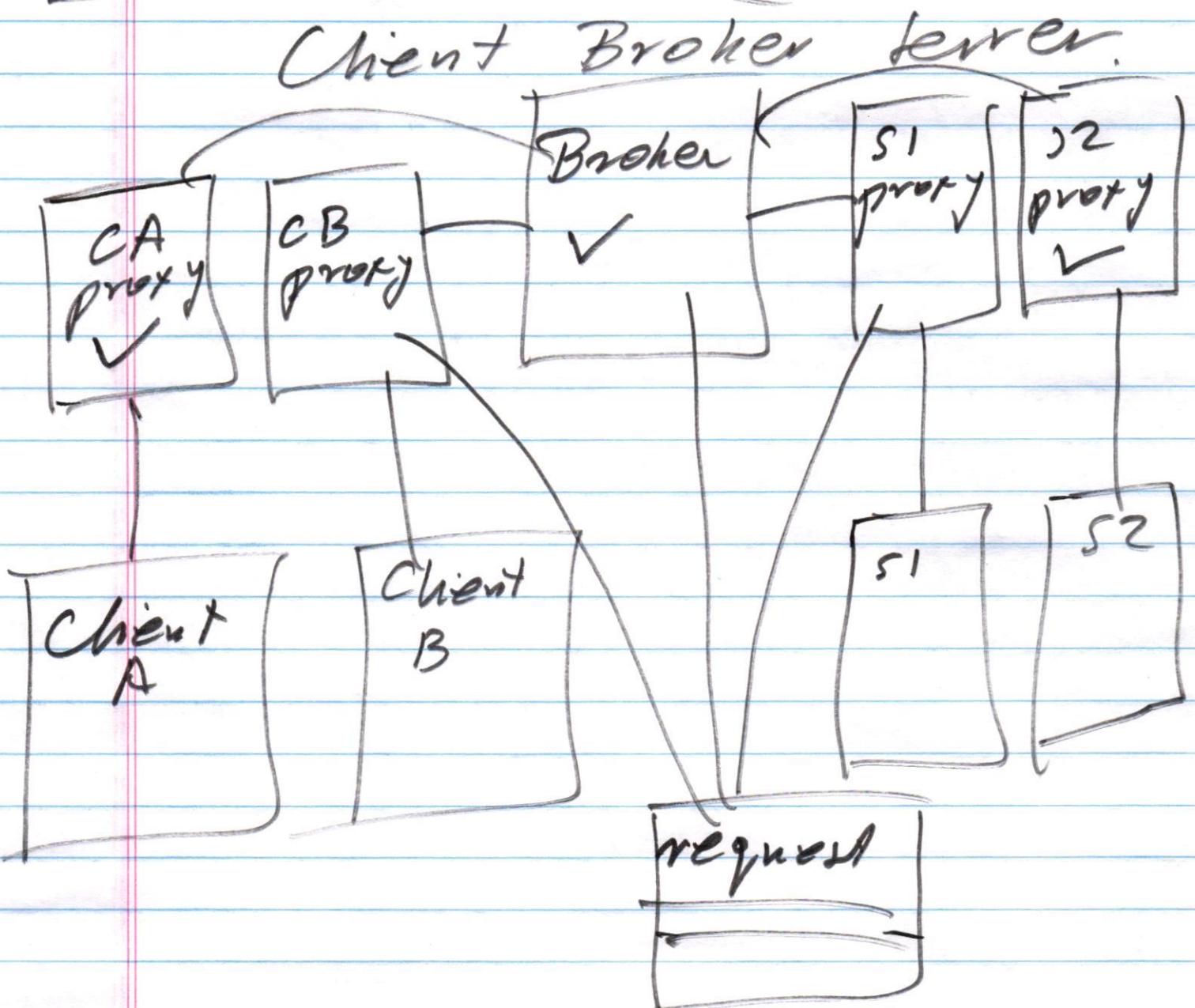
virtual actions

- StoreData
- EjectCard
- DisplayMenu



# Homework # 2

## Problem #1

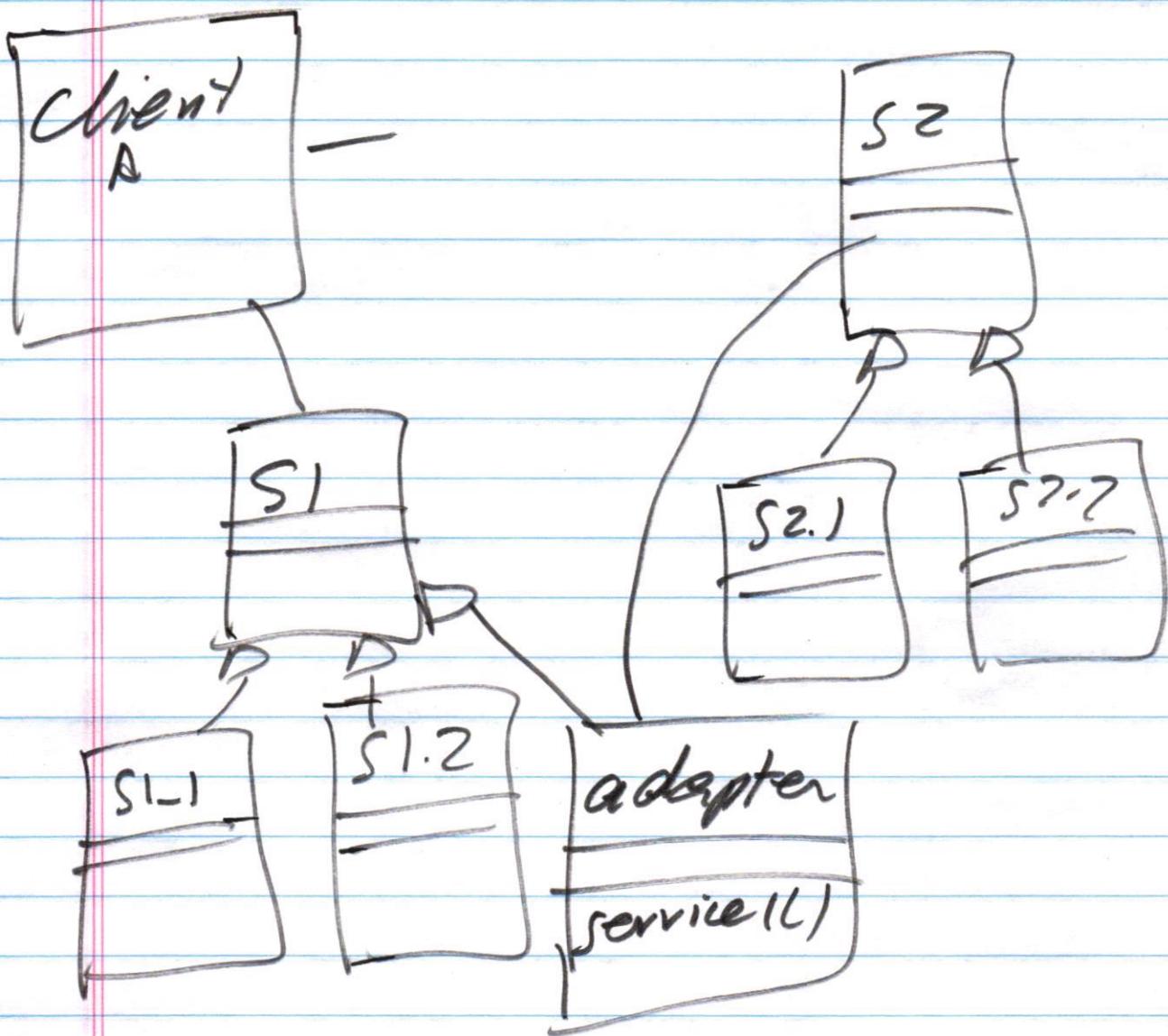


## Problem #2

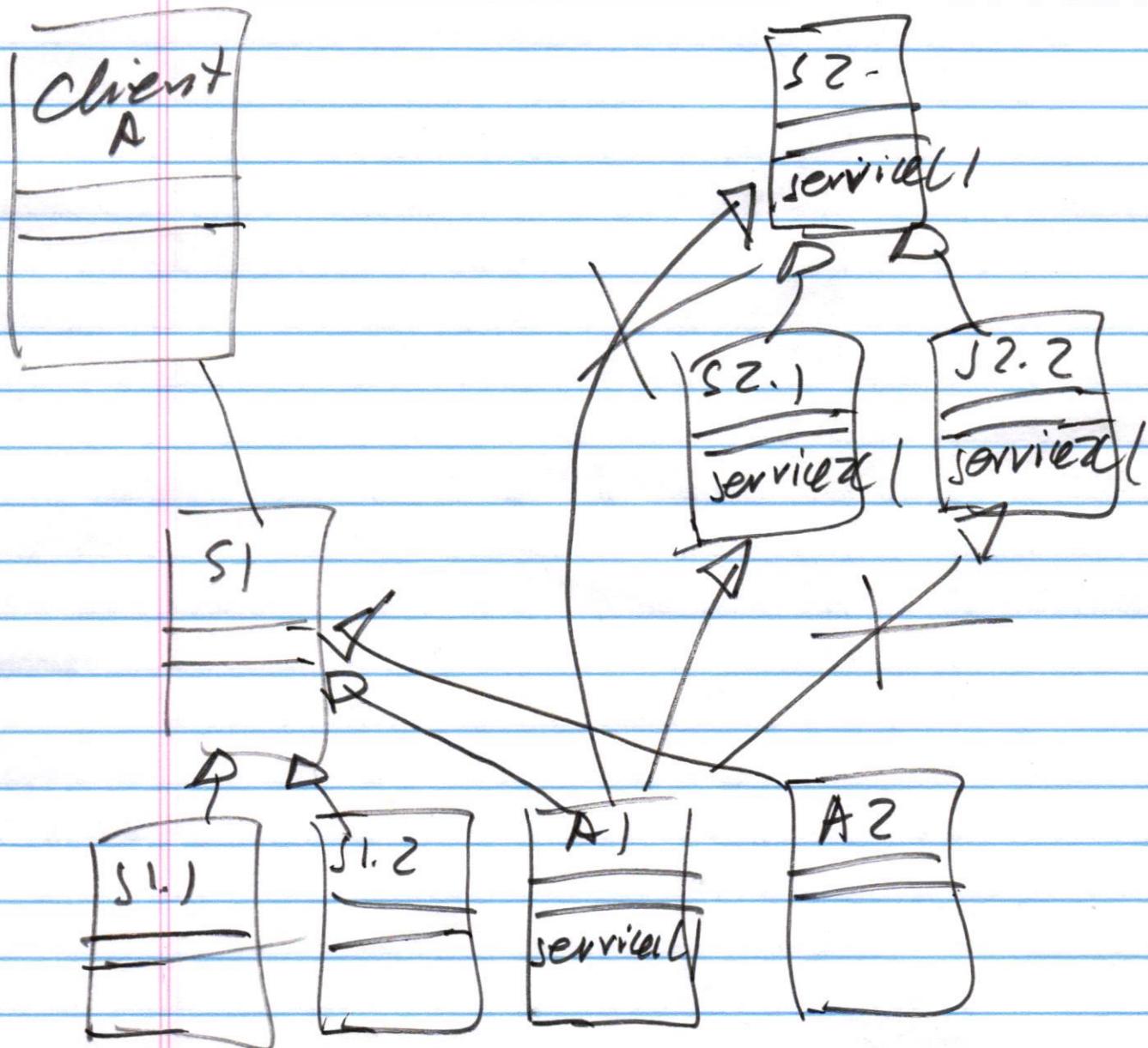
### Adapter pattern

1. ~~association~~ based  
association version
- \* inheritance based  
solution.

## association-based

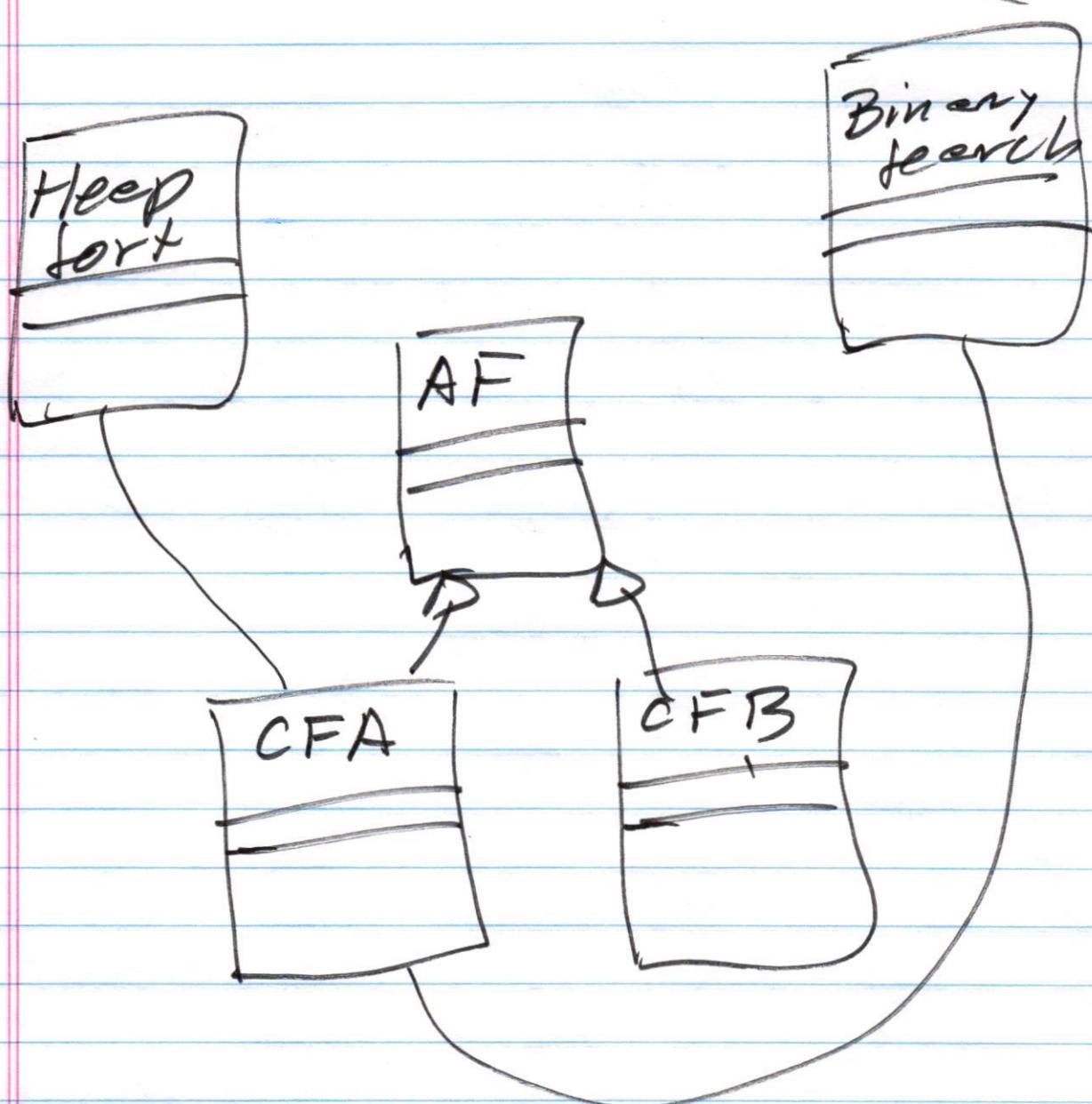


# inheritance - based



## Problem # 3

abstract factory -



## HOMEWORK ASSIGNMENT #2

CS 586; Fall 2025

Due Date: **October 16, 2025**

Late homework: 50% off

After **October 21**, the homework assignment will not be accepted.

This is an **individual** assignment. **Identical or similar** solutions will be penalized.

**Submission:** All homework assignments must be submitted on the Blackboard. The submission **must** be as one PDF file (otherwise, a 10% penalty will be applied).

### **PROBLEM #1** (40 points)

There exist two servers **S1** and **S2**. Both servers support the following services:

Services supported by **server-S1**:

```
void Service1(string, int, int)
void Service2(string, int, int)
int Service3(string)
float Service4(string)
```

Services supported by **server-S2**:

```
void Service1(string, int)
void Service2(string, int)
int Service3(string)
float Service4(string)
```

There exist two client processes, and they request the following services:

**Client-A**

```
void Service1(string, int, int)
void Service2(string, int)
int Service3(string)
float Service4(string)
```

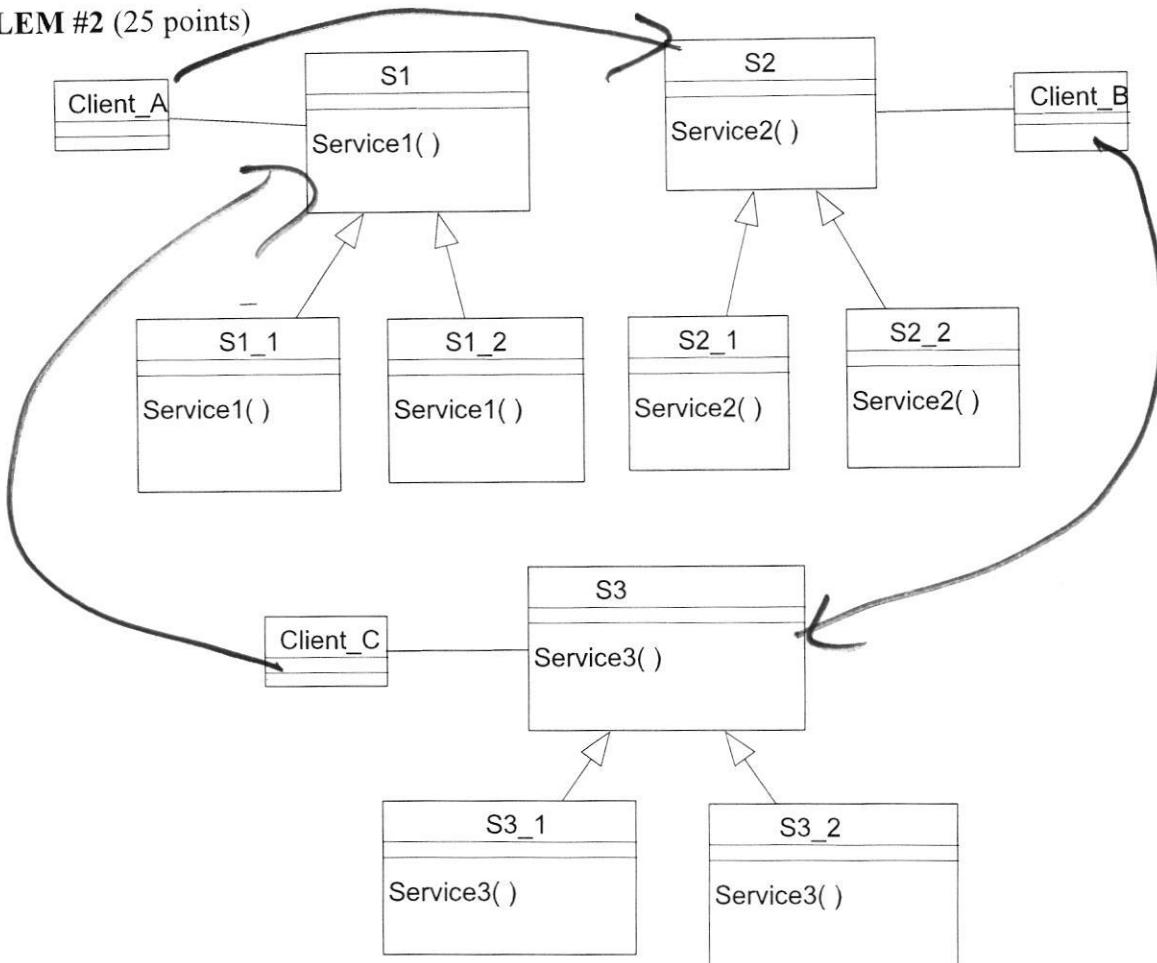
**Client-B**

```
void Service1(string, int)
void Service2(string, int, int)
int Service3(string)
float Service4(string)
```

The client processes do not know the location (pointer) to servers that may provide these services. Devise a software architecture using a **Client-Broker-Server** architecture for this problem. In this design, the client processes are not aware of the location of the servers providing these services.

- Provide a class diagram for the proposed architecture. In your design, all components should be **decoupled** as much as possible.
- Provide the pseudocode for all operations of the following components/classes:
  - Broker
  - Client Proxy of Client-A
  - Server Proxy of Server-2.
- Provide a sequence diagram to show how Client-A gets “void Service2(string, int) service.”

**PROBLEM #2 (25 points)**



A design of a system is shown above. In this system, *Client\_A* uses objects of classes *S1\_1* and *S1\_2*, *Client\_B* uses objects of classes *S2\_1* and *S2\_2*, and *Client\_C* uses objects of classes *S3\_1* and *S3\_2*.

*Client\_A* would like to use objects, operations *Service2()*, of classes *S2\_1* and *S2\_2* by invoking operation *Service1()*. *Client\_C* would like to use objects, operation *Service1()*, of classes *S1\_1* and *S1\_2* by invoking operation *Service3()*. In addition, *Client\_B* would like to use objects, operation *Service3()*, of classes *S3\_1* and *S3\_2* by invoking operation *Service2()*.

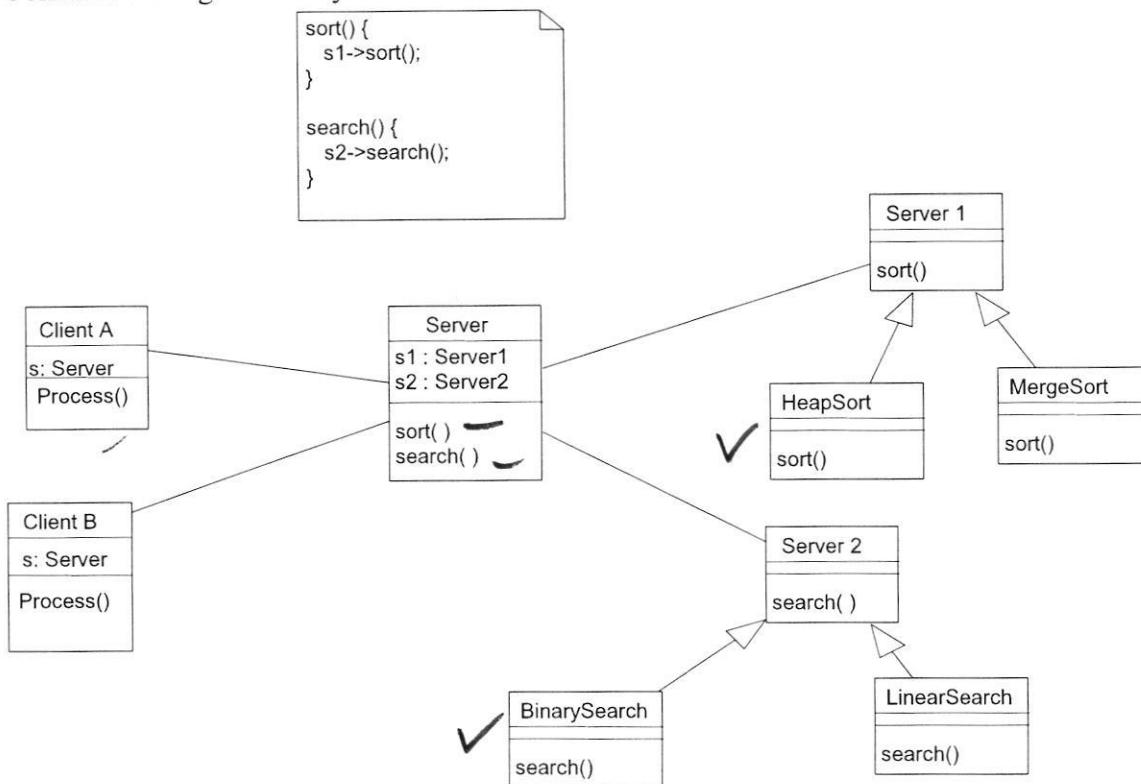
Provide a design with **minimal** modifications to the existing system using the **Adapter design pattern** in which (1) *Client\_A* can use objects of classes *S2\_1* and *S2\_2* by invoking operations *Service1()*, (2) *Client\_B* can use objects of classes *S3\_1*, and *S3\_2* by invoking operations *Service2()*, and (3) *Client\_C* can use objects of classes *S1\_1*, and *S1\_2* by invoking operations *Service3()*. Notice that none of the classes shown in the above class diagram should be modified. Provide two solutions that are based on:

1. An association-based version of the Adapter pattern
2. An inheritance-based version of the Adapter pattern

Provide a class diagram for each solution. You do not have to provide any description for classes/operations of the above class diagram (only new classes/operations should be described using **pseudo-code**).

### PROBLEM #3 (35 points)

Consider a design of the system shown below:



*Client A* and *Client B* get services, `sort()` and `search()`, directly from the *Server*. However, the *Server* gets the appropriate services from *HeapSort* or *MergeSort* servers for `sort()` service. In addition, the *Server* gets the appropriate services from *BinarySearch* or *LinearSearch* servers for `search()` service.

In this design, clients do not have control where the services are coming from. However, *ClientA*, by invoking `sort()` and `search()` on the *Server*, would like to get `sort()` service from *HeapSort* server and `search()` service from *BinarySearch* server. On the other hand, *ClientB*, by invoking `sort()` and `search()` on the *Server*, would like to get `sort()` from *MergeSort* server and `search()` from *LinearSearch* server. Notice that the current design does not support this option.

Use the **abstract factory** design pattern to solve this problem. In your solution, the *Client* classes should be completely **decoupled** from the issue of invoking services by the *Server* for appropriate versions of `sort()` and `search()`. Notice that none of the classes (and operations) shown in the above class diagram should be modified. However, new operations/classes can be introduced.

- Provide the class diagram and briefly describe the responsibility of each class and the functionality of each operation using **pseudo-code**. In your design, all components should be **decoupled** as much as possible. You do not have to provide any description for classes/operations of the above class diagram (only new classes/operations should be described).
- Provide a sequence diagram to show how *ClientB* gets `sort()` service from *MergeSort* server and `search()` service from *LinearSearch* server by invoking `sort()` and `search()` on the *Server*.