Freie Universität Berlin

Two methods for modeling the architectural patterns

- Manar Zaboub
- Computer science institute
- Free University of Berlin

What are patterns?

Patterns

- They state a recurring problem
- Provide a solution for the problem
- Provide the rationale behind the solution

Patterns in computer science

Design patterns

- Proxy
- Command
- Iterator

Architectural patterns

- Service-based architecture
- Pipe and filter
- Microservices

History of the metaphor

1977

A pattern language

A book creates the language of patterns used for architecture

1994

Design patterns

A book describes software design patterns, inspired by “A pattern Language”

1995

JavaScript

Uses the keyword interpreter pattern labeled by design patterns

1997

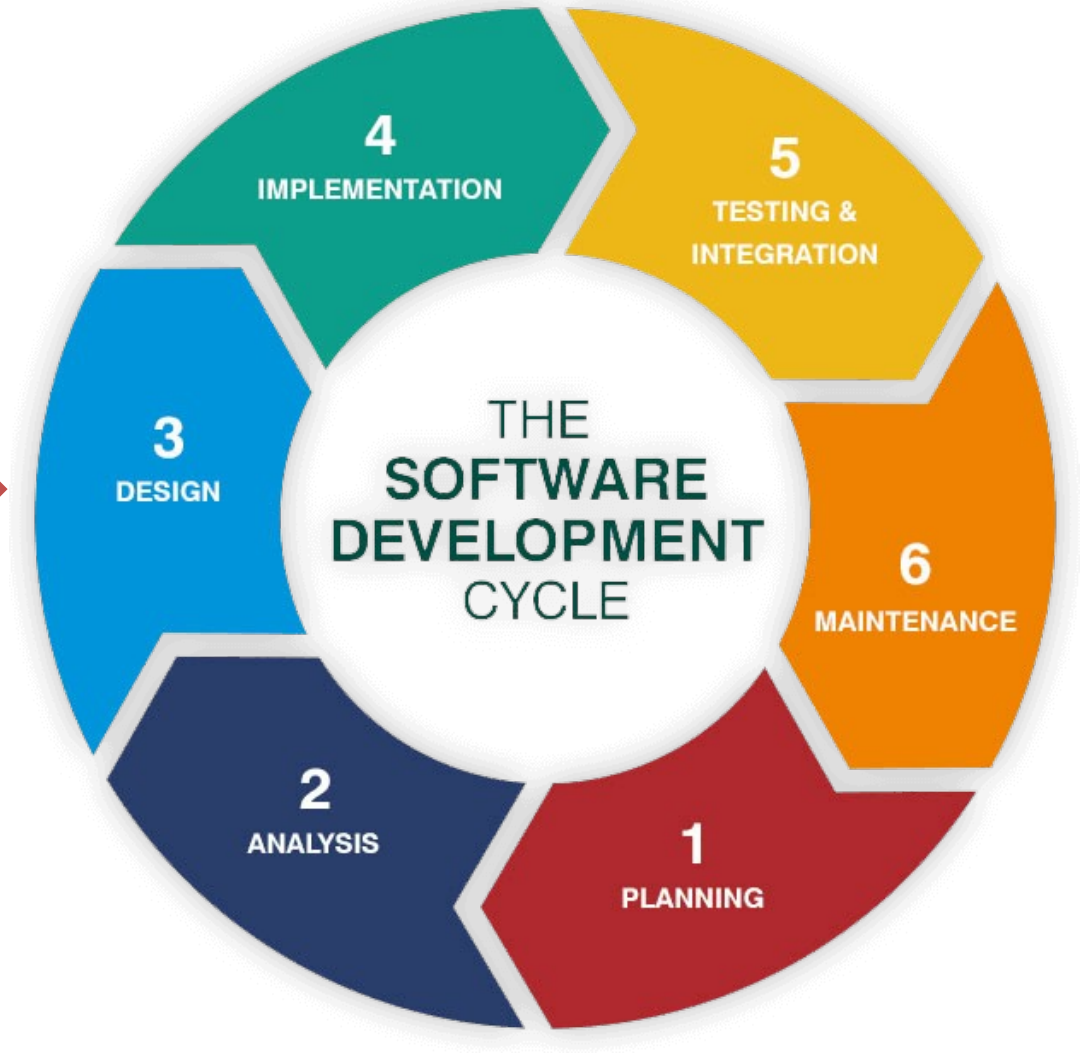
Web proxy servers

Intermediary program between clients and servers

Using architectural pattern

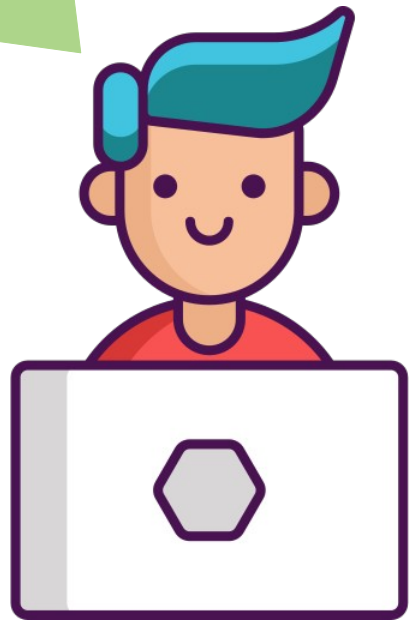
Software development life cycle

Use architectural patterns



Design a web app

I want to design a reliable web app, but I don’t know anything about web apps. I only new the pattern Model View Controller

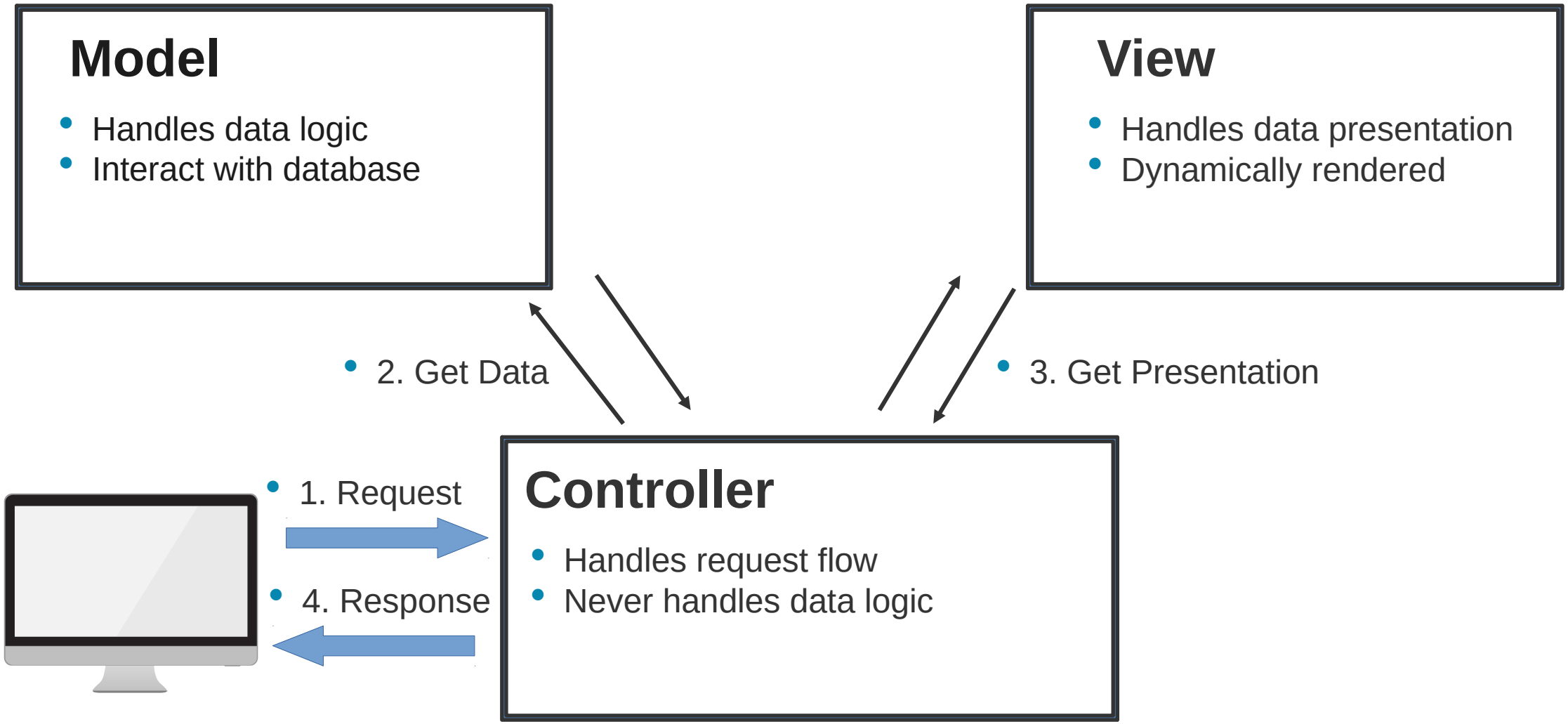


Novice developer

Model View Controller

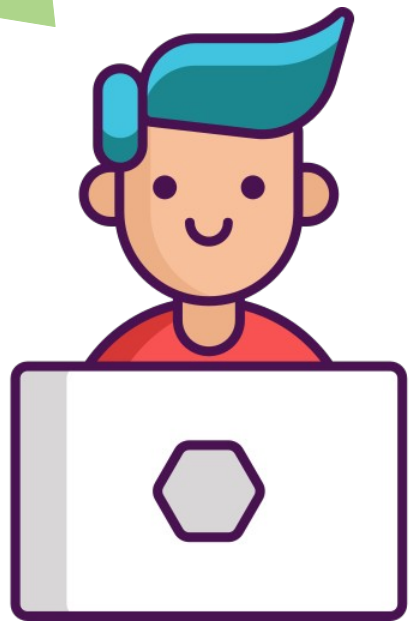
- One of the most famous architectural patterns for GUI’s
- Components have specific responsibilities
- **Model**: Represent the data, and save objects into database
- **Control**: Control the flow between the user of the app, and the app it self
- **View**: Render the result, determines presentation of the output

MVC structure



Documenting

I have to document my solution, because I am a good developer



Novice developer

Documenting advantages

- Documentation facilitates communication between stakeholders
- In progressed stage of development it gives the reason why a certain design was chosen in favor of the other
- It is a necessity for software maintenance
- Modeling of architectural patterns is a part of documentation

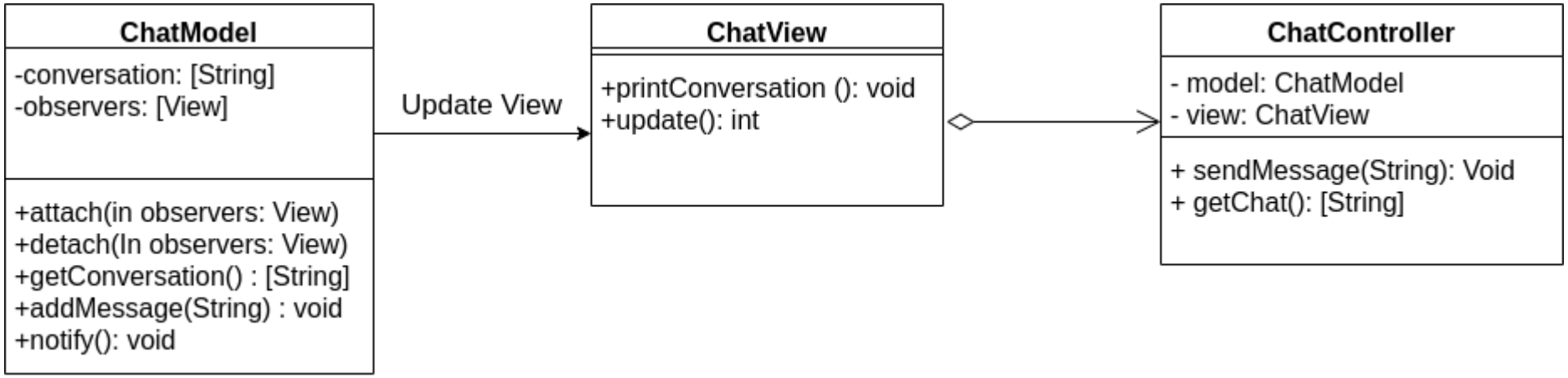
Modeling of architectural patterns

MVC using observer

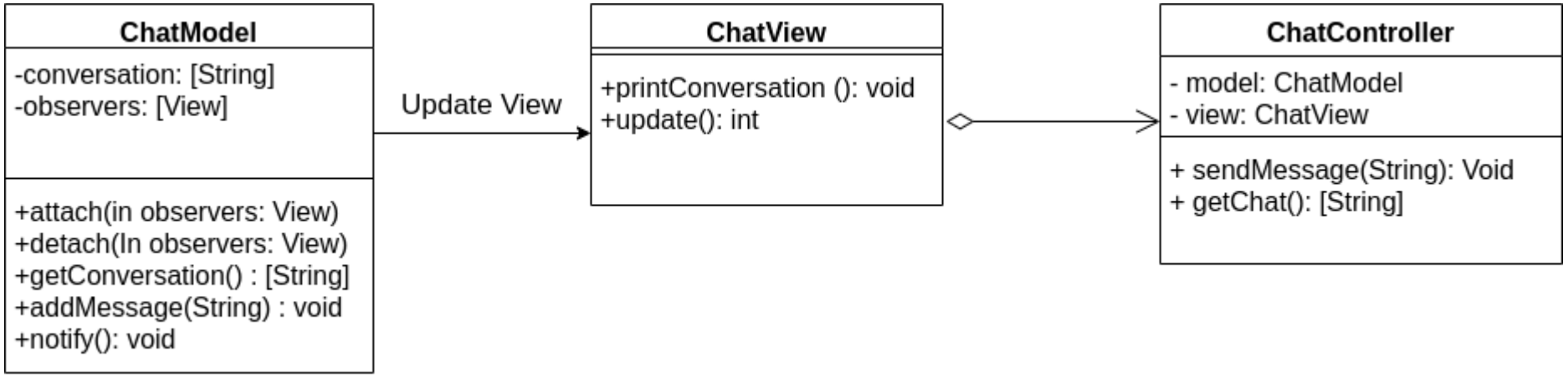
Send

Receive

UML modeling of observer



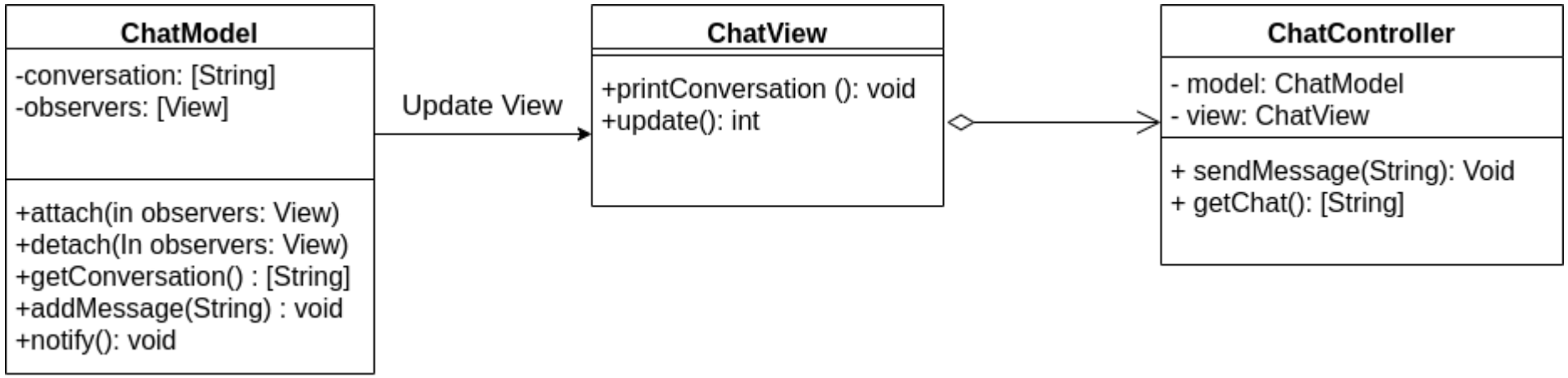
Implementation of update view




```
class Model {

    Void Attach(View view) {
      this.Observers.add(view);
    }

    void Notify(){
       This.Observers[0].update()
    }

    void addMessage(){
       this.notify()
    }
}
```

```
class View {

    void update() {
      SomeCodeHere();
    }

}
```

Arguments in modeling callbacks



- Use void for Callbacks: Callback may return a value
- Use sequence accompanying diagram:
  - No semantic annotations: It can be a normal back and forth invocation
  - Temporal decoupling: Many invocations happen between performing the call back and the event what caused it to be invoked

Proposed solutions

1 Using UML 2.0 meta-classes

- Zdun, Uwe & Avgeriou, Paris. (2005). Modeling architectural patterns using architectural primitives. ACM SIGPLAN Notices. 40. 133-146. 10.1145/1094811.1094822.

2 Using a new platform *alfa*

Mehta, Nikunj & Medvidovic, Nenad. (2003). Composing architectural styles from architectural primitives. ACM Sigsoft Software Engineering Notes. 28. 347-350. 10.1145/949952.940118.

1st Solution: UML

- Modeling Architectural Patterns Using Architectural Primitives
- Define templates for the primitive:

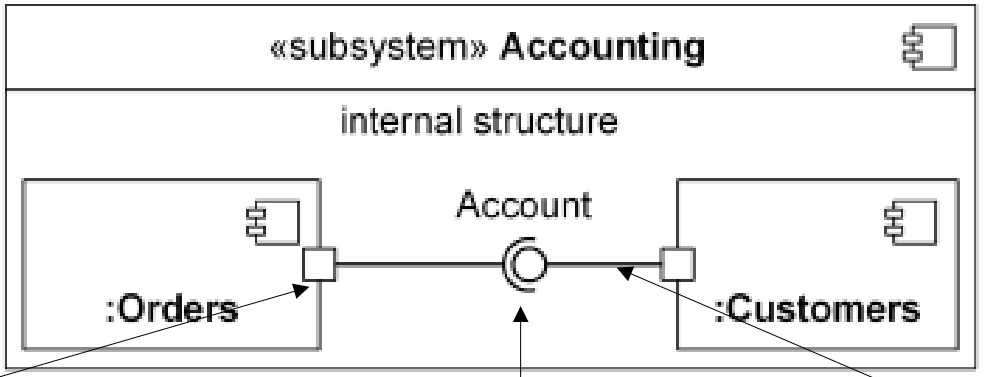
  1) Textual description: What is an observer/callback?

  2) Known uses in patterns: Where/When to use observer/callback?

  3) Modeling issues: Problems in modeling observer/callback

  4) Modeling solution: Extends UML meta-classes

UML 2.0 Meta classes



Port

specify a distinct interaction point between the com-ponent that owns the port and its environment

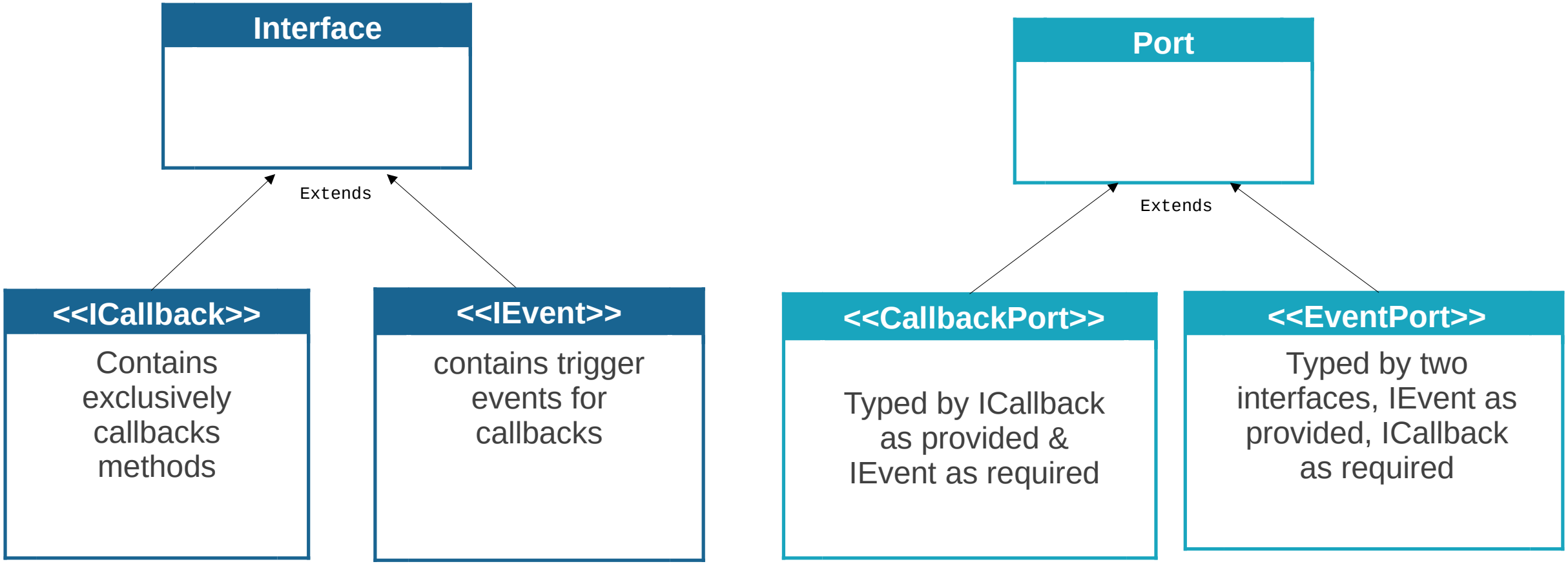
Interface

serve as contracts that components must comply with

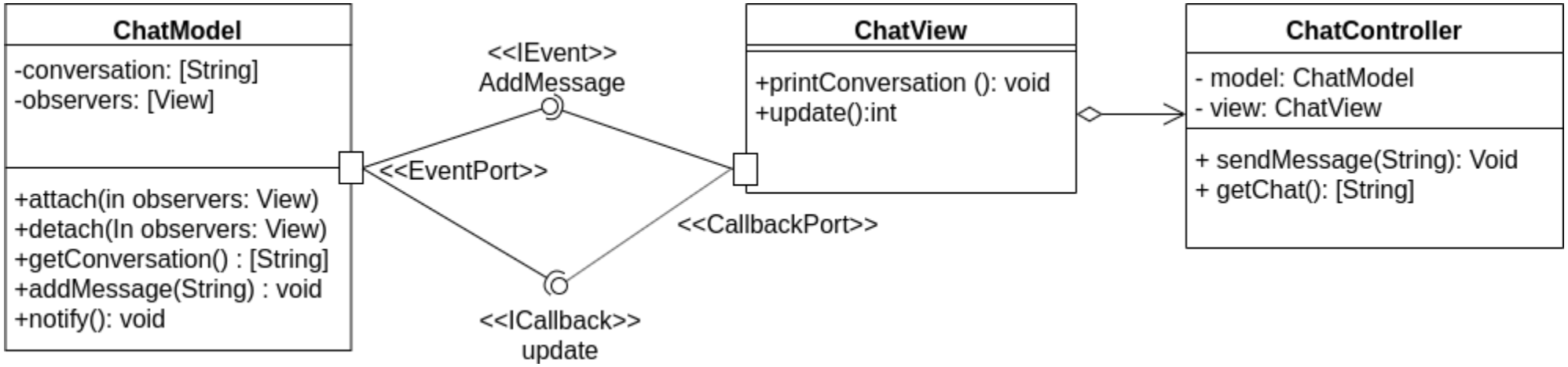
Connector

connect the required interface of one component to the provided interface of a second

Extending UML



Applying 1st solution



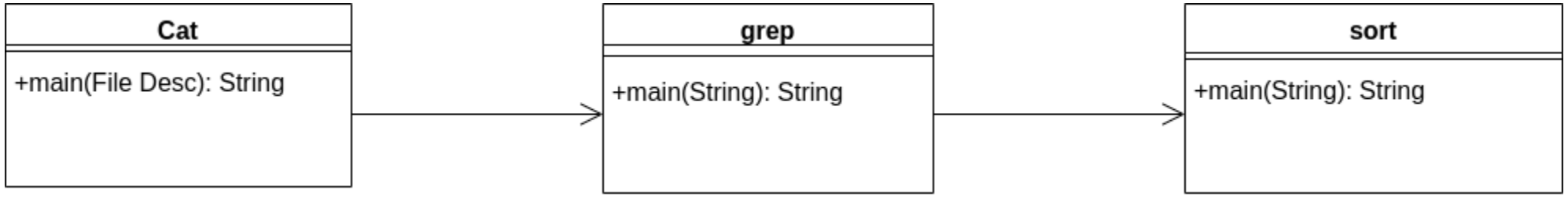
Questions?

Pipe and filter

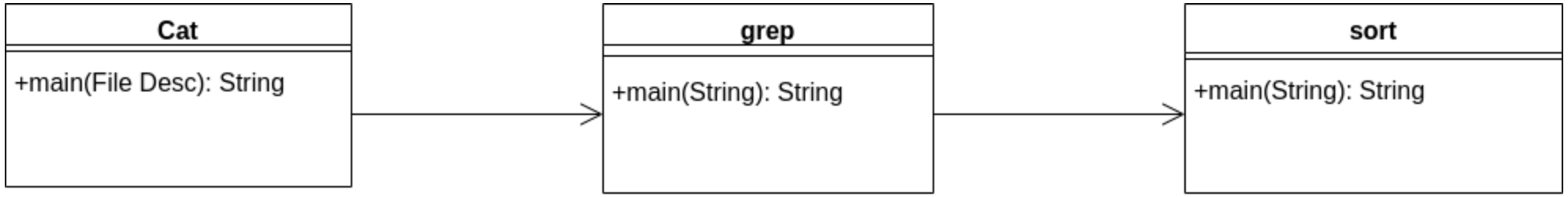
- Consists of chain of data processing filters
- Filters are connected through pipes
- In Linux “ | “ is a pipe. Any process is a filter
- Example to sort a file in linux:

Pipe and filter in UML

- How to express active filters, and passive filters
- Pipes doesn’t match the UML connector, since connectors don’t have a state (close, open)



Implementation of Pipe & filter




```
class grep {
  Void activate() {
    Message val = inPipe.read();
    val = transform(val);
    outPipe.write(val);
   }
}
```

```
class Sort {
  void controlLoop() {
     Message [] text = val;
     while(Message val = inPipe.read()){
      Text.append(val);
      }
      Message out = sort(Text);
      outPipe.write(out);
   }
}
```

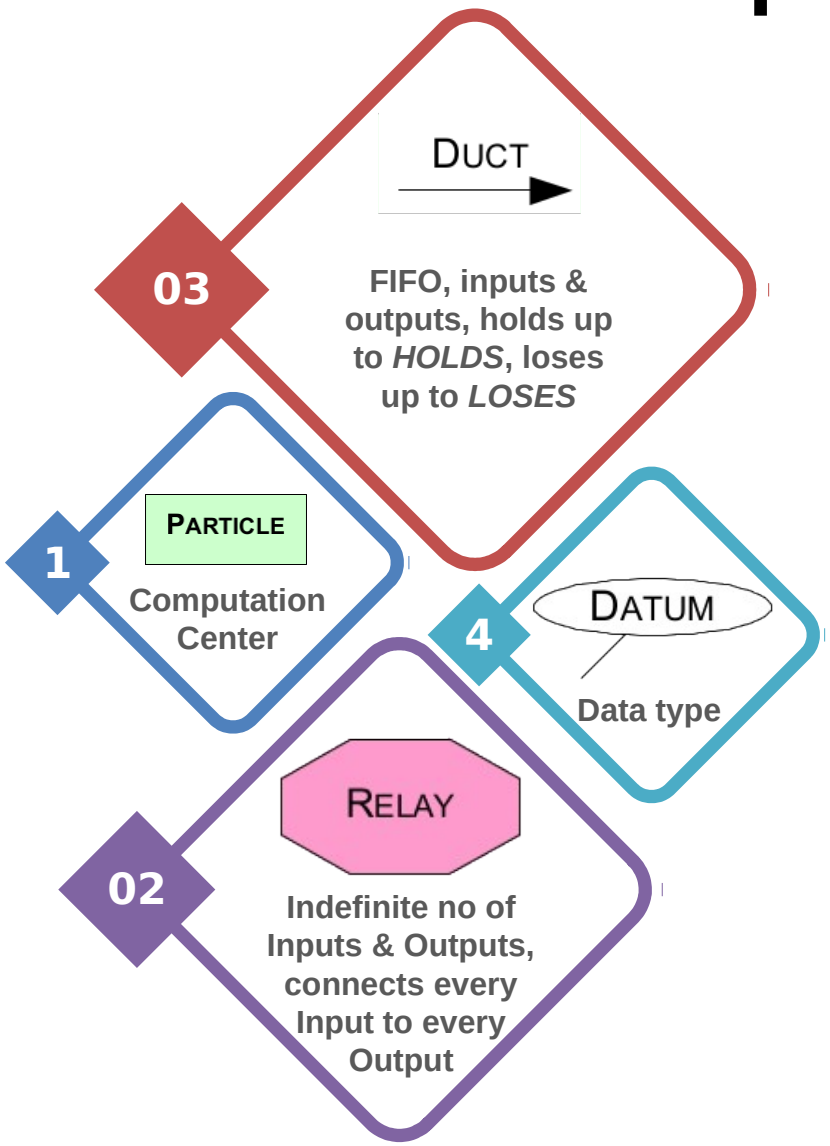
2nd Solution: *Alfa*

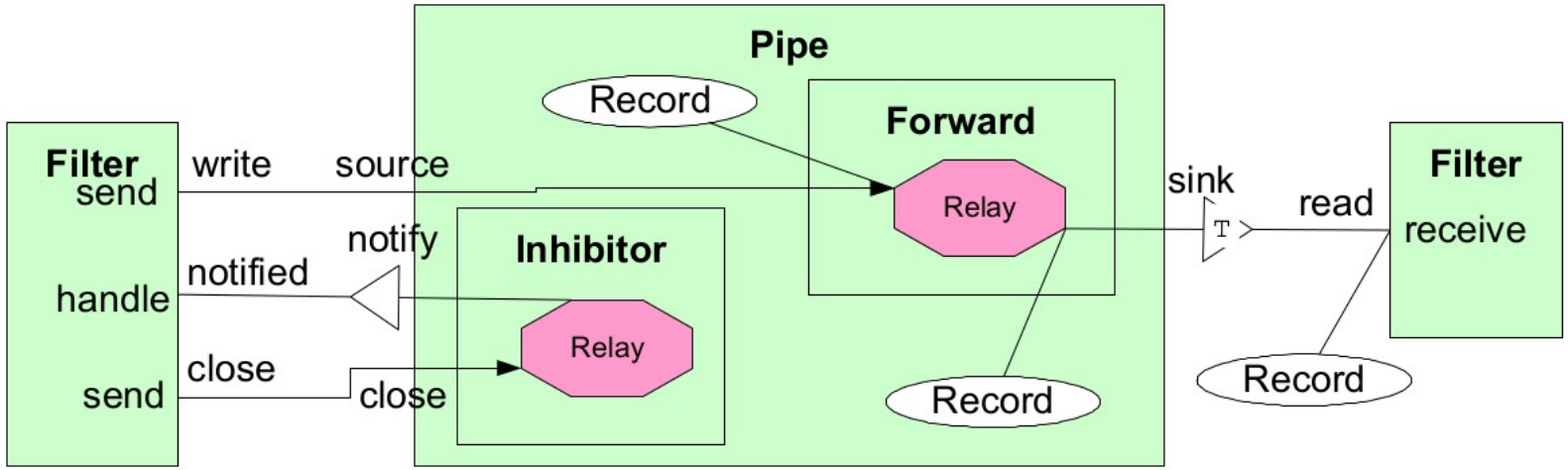
- Components are black boxes relating inputs and outputs
- Connectors have a visible structure made of *Alfa*'s primitive
- *Alfa*'s primitives:
  - 8 nouns "capturing the form of architectural styles"
  - 9 verbs "capturing the elements' function"

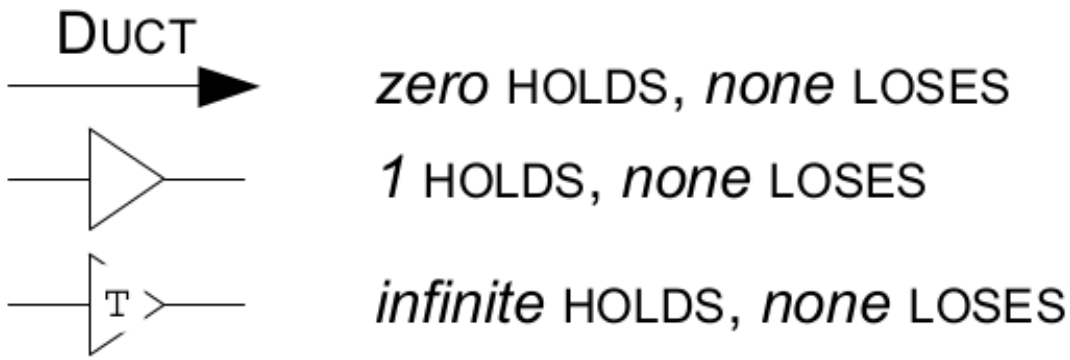
Alfa's primitives

- **Data** - DATUM
- **Structure** – PARTICLE, OUTPUT, INPUT, TWOWAY
- **Interaction** – DUCT, RELAY, BIRELAY, HOLDS, LOSES
- **Behavior** – CREATE, SEND, RECEIVE, HANDLE, REPLY
- **Topology** – CONNECT, DISCONNECT

Pipe and filter in *alfa*

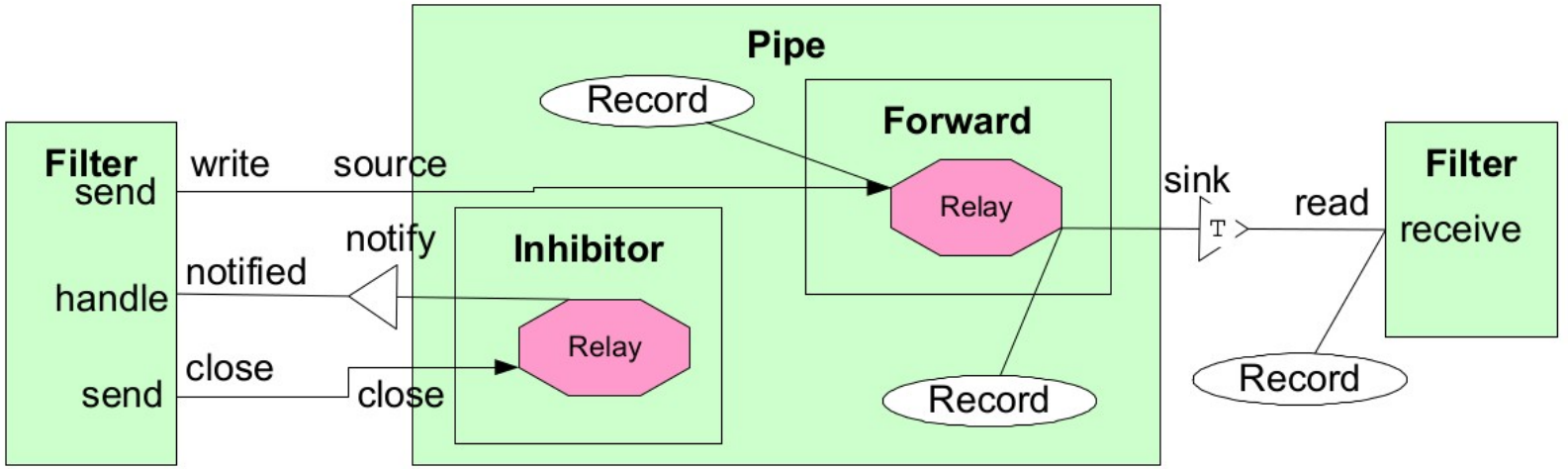






Cons of *alfa*

- Particles (or Classes) are only black boxes
- We don’t know what methods or attributes are used
- *Alfa* can’t be used for maintenance of software



Comparison: UML or no-UML

- We have two solutions, one with UML, and the other one is without UML
- *Alfa* is a one purpose approach, can be used only for architectural patterns
- UML can be used to model design, and architectural patterns
- Many perceive the documentation of analysis and design models in UML to be a wasteful activity [1]
- Experimental studies shows that UML facilitates the understanding and change of code

Experimental evaluation

- Arisholm, Erik & Briand, Lionel & Hove, Siw & Labiche, Yvan. (2006). The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation. Software Engineering, IEEE Transactions on. 32. 365- 381. 10.1109/TSE.2006.59.

Experiment

- Research Questions:

  1) Does UML documentation help reduce the cost to change the code?

  2) Does UML documentation help achieve better code change reliability?

- 2 Experiments, one in Oslo, second in Ottawa
- Give 2 groups the same set of tasks to be solved, one group should use UML documents, one group shouldn’t use UML documents
- Provide UML documents: Use case, sequence, and class diagrams
- Measurements: Time with and without diagram modification, correctness of change, and quality of the changed design

Experimental tasks

- 2 systems, ATM and Vending Machine

Systems' and Tasks' Descriptions

System	Tasks	Description	Oslo	Ottawa
ATM	Task 1	Print out an account transaction statement	x	x
	Task 2	Transfer money between two accounts	not given	x
Vending	Task 3	Implement a coin-return button	x	not given, already implemented
	Task 4	Make bouillon as a new type of drink	x	not given, already implemented
	Task 5	Check whether all ingredients are available for the selected drink	x	x
	Task 6	Make your own, customized drink based on the available ingredients	"Time sink" task	x

Oslo’s experiment results

- Some of the UML experiment participants didn’t use the UML diagram
- On average more time required (T`) when using UML
- Correctness (C) significantly improved when using UML

		Tasks											
		Task 1			Task 3			Task 4			Task 5		
Dep. Var.	Group	Min	Med	Max	Min	Med	Max	Min	Med	Max	Min	Med	Max
T (excl. model modification)	No UML	20	75	240	5	20	60	10	22	55	16	54	150
	UML	31	53	95	8	15	23	12	19	35	45	65	95
T’ (incl. model modification)	UML	43	70	105	15	27	41	24	36	85	62	101	132
C (binary)	No UML	46%			91%			91%			46%		
	UML	56%			89%			100%			89%		

Ottawa's correctness results

- C` is the ratio of successful tests to the failed tests, because in Ottawa experiments the correctness measurement was automated
- There is an inconsistency between two of the results

		Tasks											
		Task 1			Task 2			Task 5			Task 6		
Dep. Var.	Group	Min	Med	Max	Min	Med	Max	Min	Med	Max	Min	Med	Max
T (excl. model modification)	No UML	22	66.5	159	20	75	154	32	89.5	180	74	166.5	194
	UML	24	82	240	38	87	162	35	99	196	51	141	184
T' (incl. model modification)	UML	40	128	261	74	149.5	180	58	141	223	75	168	184
C' (ratio)	No UML	2/8	8/8	8/8	0/8	5/8	8/8	0/8	5/5	5/5	0/12	0/12	12/12
	UML	2/8	8/8	8/8	2/8	5/8	7/8	1/8	5/5	5/5	0/12	5/12	12/12

Result’s inconsistency

- Qualitative results show that the learning effect played a big role in Oslo’s results
- Upon reaching task 5, participants were aware of the potential benefits of using UML more clearly, hence used it more actively

Conclusion

- We can extend UML 2.0 to model architectural patterns
- We can use a third party platform to model architectural patterns
- Using UML is important for development and maintenance of software

Sources

- **Photos:**
  - https://161cli18elctkuzva3yluzd6-wpengine.netdna-ssl.com/wp-content/uploads/2018/12/Relevant-Software-product-development-life-cycle.png
  - https://image.flaticon.com/icons/png/128/1488/1488581.png
- **Slides**:
  - https://www.free-powerpoint-templates-design.com/lighthouse-landscape-powerpoint-templates/

Sources

- **Tables and diagrams:**
  - Slide# 29-30: Mehta, Nikunj & Medvidovic, Nenad. (2003). Composing architectural styles from architectural primitives. ACM Sigsoft Software Engineering Notes. 28. 347-350. 10.1145/949952.940118.
  - Slide# 34-36: Arisholm, Erik & Briand, Lionel & Hove, Siw & Labiche, Yvan. (2006). The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation. Software Engineering, IEEE Transactions on. 32. 365- 381. 10.1109/TSE.2006.59.