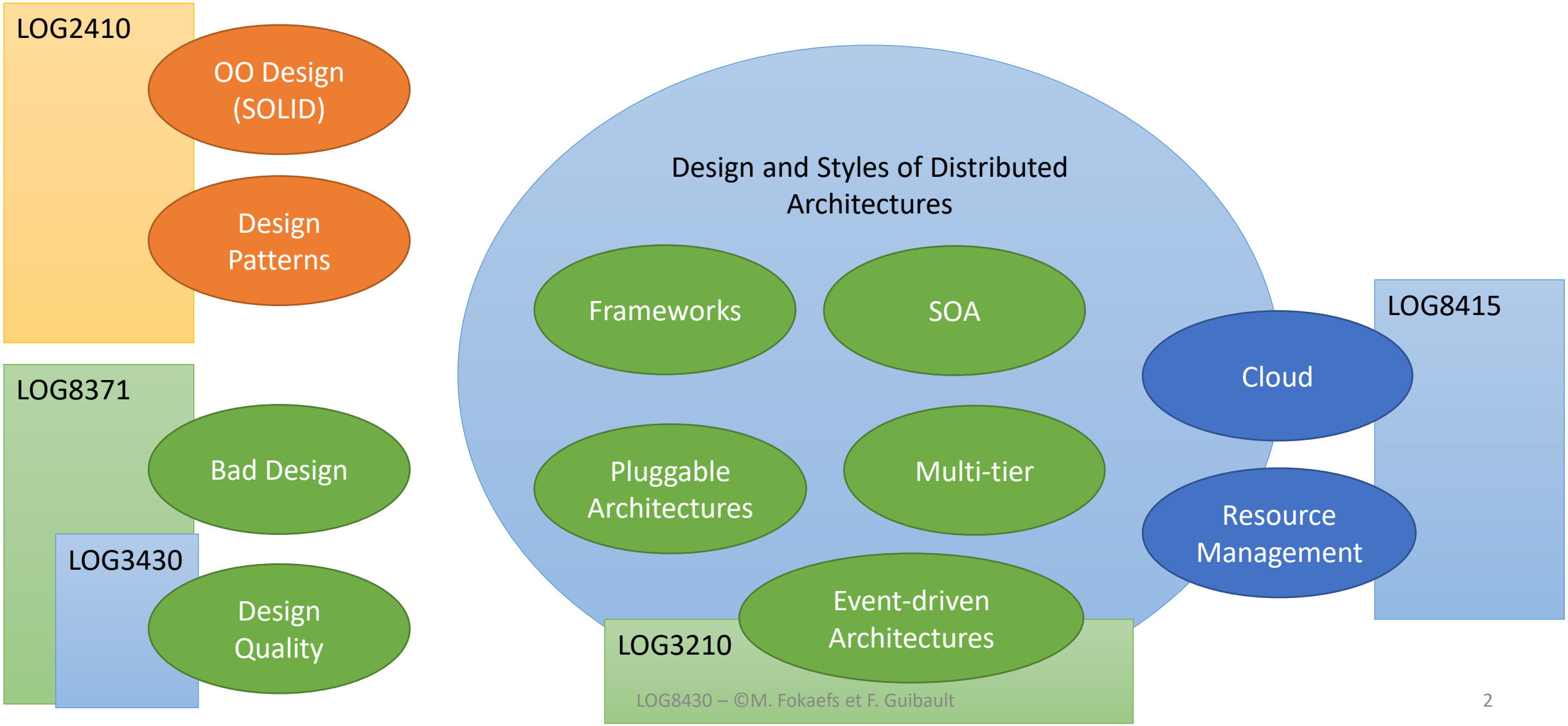
POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

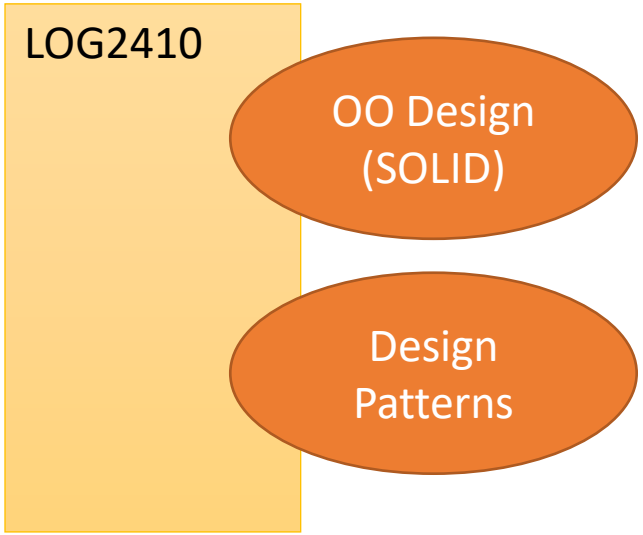
LOG8430E: Architectural Styles

Fall 2020

Course Map


LOG2410
OO Design (SOLID)
Design Patterns
LOG8371
Bad Design
LOG3430
Design Quality
Design and Styles of Distributed Architectures
Frameworks
SOA
Pluggable Architectures
Multi-tier
Event-driven Architectures
LOG3210
Cloud
Resource Management
LOG8415

So far…



Today

LOG2410

OO Design (SOLID)

Design Patterns

Design and Styles of Distributed Architectures

Evolution and styles of architecture

Monolithic

- Mainframe

Modular

- Pipe-Filter
- Layered
- MVC
- Blackboard
- Event-bus

Distributed

- Client-server
- Multitier
- Master-slave
- Peer-to-peer
- Broker
- SOA
- REST

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Monolithic Architectures

Monolithic Architectures

- A single program
  - A single point of change.
- There is no distinction between UI, logic, data etc.
  - The program elements are strongly connected.
- The program is developed and maintained solely by a small team of developers/experts.
  - Absolute and concentrated knowledge.
- There is a single owner of the entire system.
  - Absolute control.
- The logic and the data exist in a single place.
  - The ownership is well-protected.
- Each copy of the program is executed on a single computer.
  - Execution costs are reduced.

Monolithic: Advantages

- Maximum security! No opportunity for an external attack.
- The program is often efficient, because everything is loaded in memory and there are no delays related to network or to module loading.
- Thanks to the size of the team, the communication and the efficiency of the team are optimised.
- Development, maintenance, and evolution of the system are the responsibility of a single entity.

Monolithic : Disadvantages

- No reusability. It is possible to reuse parts of the system, but only if they are copied entirely.
  - This creates clones.
- No scalability or extensibility.
- Difficult to maintain!
  - “The software crisis”
- Difficult to onboard new developers.
- Increased costs for the maintenance of the system.

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Modular Architectures

Modular Architectures

- The system consists of many modules.
- The functionalities and the responsibilities are more distinguishable.
- The modules are developed and compiled separately.
- The modules are smaller and easier to manage.
- The development is shared among more and smaller teams.
- These architectures inherently implement the SRP and DIP principles.
- The ownership of the system always belongs to a single entity.

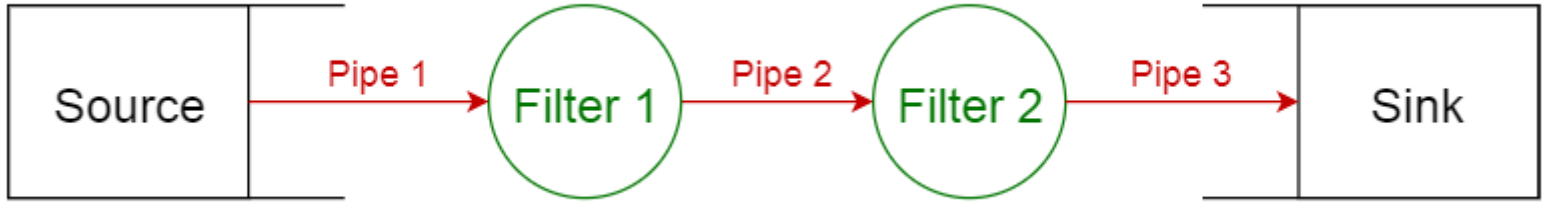
Modular: Advantages

- Development is parallelized, so it is more efficient.
- The maintenance is generally easier, since the local and internal changes do not affect the other modules.
- The modules are as autonomous as possible in terms of testing.
  - Unit testing.
- It is easier to add new staff, as the modules are smaller and easier to understand, and there are fewer dependencies.
  - Design by contract: The interfaces are very specific and known.
- The modules can be reused and loaded.

Modular : Disadvantages

- How to assemble the modules in a single system?
- The distribution of the work increases the need for communication between the teams.
- The knowledge of the entire system is lost.
  - We need a hierarchical organisation.
- There are fewer dependencies, but they are probably more important.
  - A fault can exist in a module, but it may remain unknown to the other modules. Its effect can become apparent in other places of the code, but its correction is possible only within the module.
- We need to invest more effort in the management of the system.
  - Testing, debugging, monitoring, maintenance.

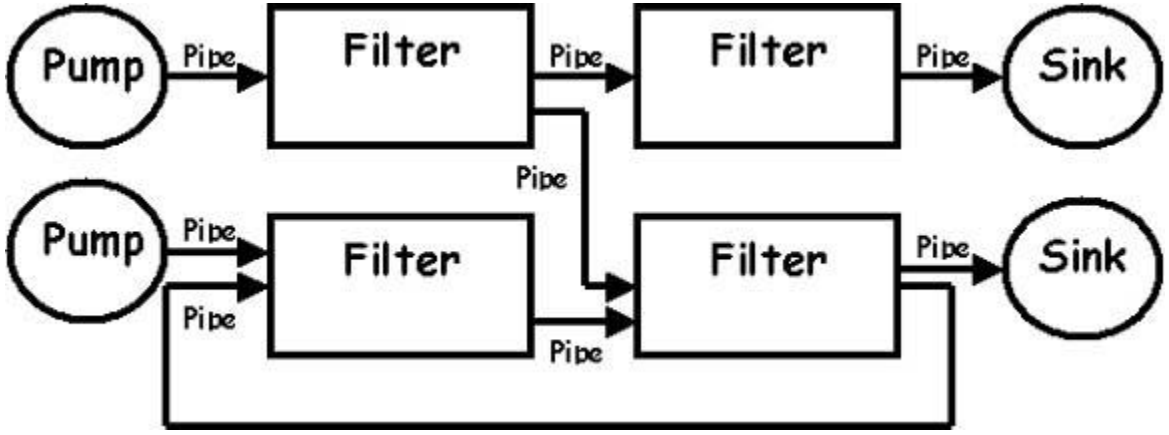
Pipe-filter

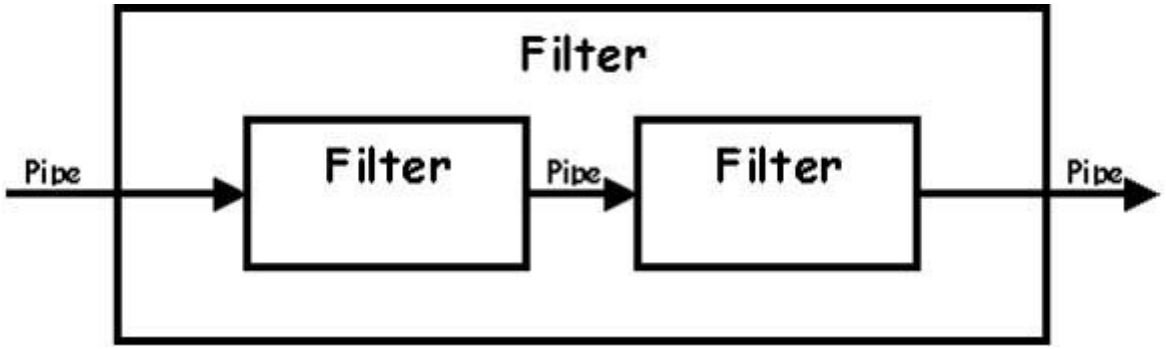


- Decomposition of a functionality in successive components applying incremental transformations on data.
- A producer sends data via pipes and through filters to a consumer.
- They can also function as aggregators of data or synchronisers.
- The filters are transformers applied progressively to data.
- Examples: UNIX programs, compilers (lexical analysis, syntactic analysis, code generation), deep learning

Pipe-filter : Advantages

- Efficient: The filters can be parallel.
- The filters are independent from producers and consumers and they can be reused.
- It is possible to add or remove pipes and filters at runtime.
- If there is not sufficient data in the pipes, the filters simply wait.
- It is possible to introduce recursion and composition.



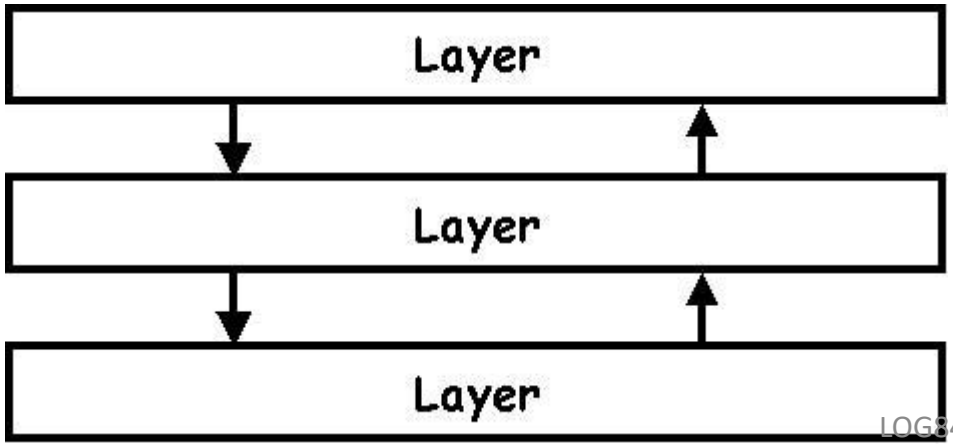


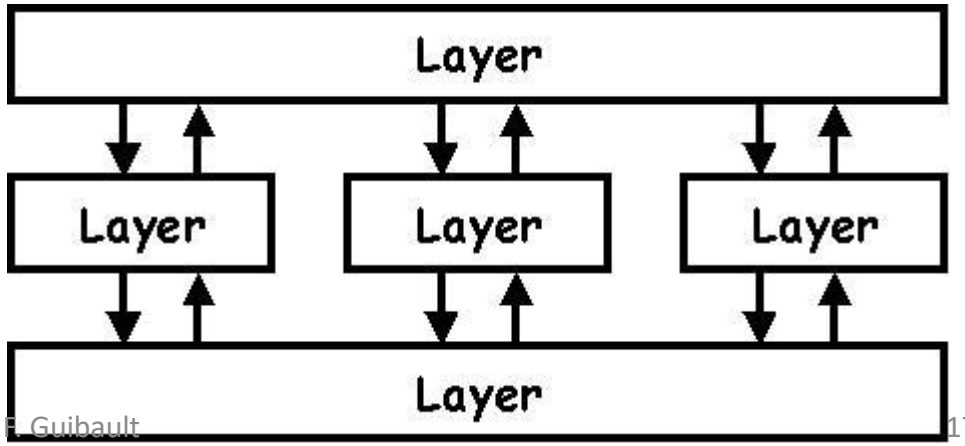
Pipe-filter : Disadvantages

- It is possible to have cases of capacity overload or deadlock, if a filter needs to wait for all data.
- The pipes often accept only simple types of data. As a consequence, the filters need to do more work to push and extract the data in the pipes.
- If we implement pipes for different types, it is no longer possible to connect any pipe to any filter.

Layered

- The division of the architecture in layers.
- The layer is an abstract notion and it can include groups of methods, of classes, of packages etc.
- Each layer captures a well-defined and distinct responsibility.
  - Presentation (UI), Application (Service), Logic (Domain), Data Access (Persistence)
- The elements of a layer can use only modules in the previous or subsequent layer.
  - The functionality is executed sequentially.
- It is possible to have multiple layers on the same level, but they cannot communicate with each other.





Layered : Advantages and Disadvantages

- Advantages
  - Maintenance is easy and cheap.
  - We can exploit the expertise of the developers.
  - The layers can be reused and shared among many other layers.
- Disadvantages
  - It is prohibited (or really complicated) to use a layer that is not immediately adjacent.
  - The architecture is sensitive with respect to the design of interfaces.

Model-View-Controller

- We can say that it is a version of the layered architecture (but not always).
  - Model layer: responsible for data.
  - View layer: responsible for the representation.
  - Controller layer: responsible for the logic.
- A single model can be represented by multiple views.
  - The model is responsible for knowing all the views and to inform them of any change (Observer pattern).
- The model cannot be part of the system. A link to a database and the functionality to access the database are sufficient.
- The controller reacts to events in the view and it knows how to change the model.

MVC

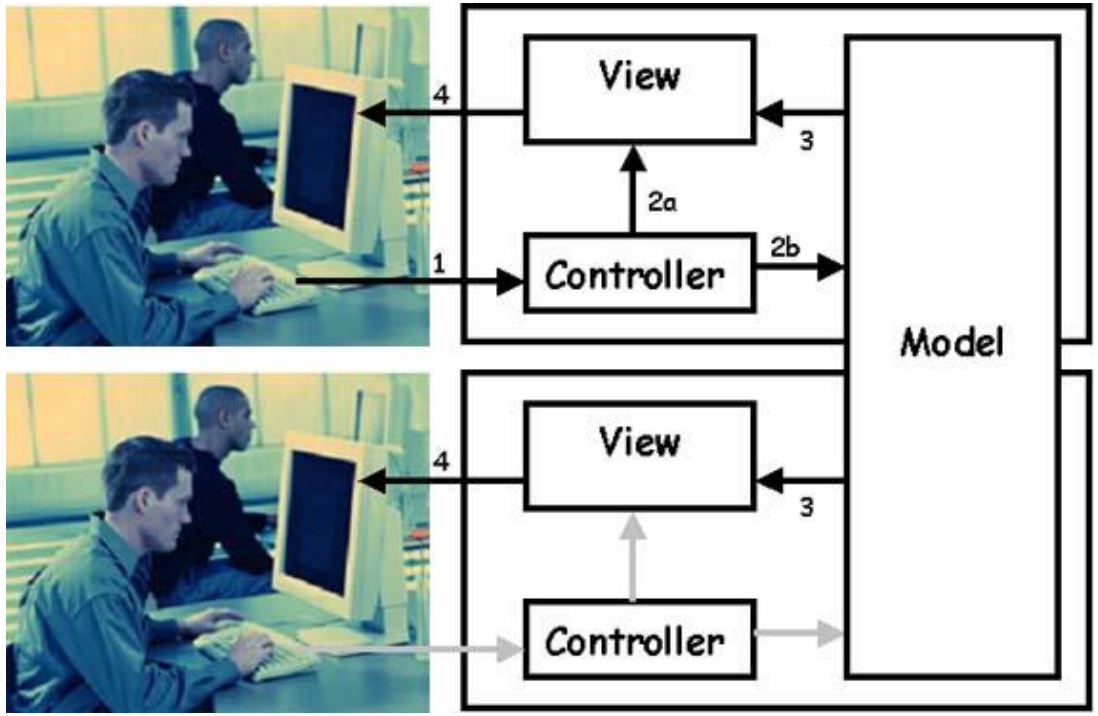
1. The controller accepts an action by the user.

2a. According to this action the controller updates the characteristics of the view…

2b. … and/or the model.

3. If the model has changed, it notifies all registered views.

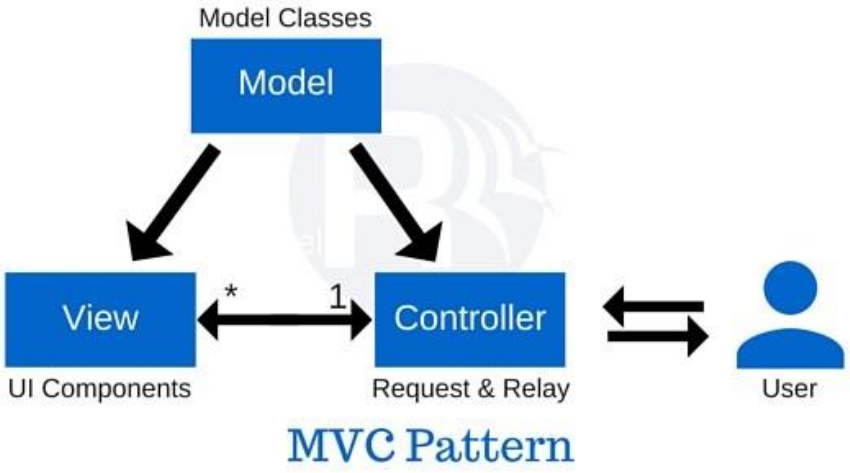
4. The views are updated.

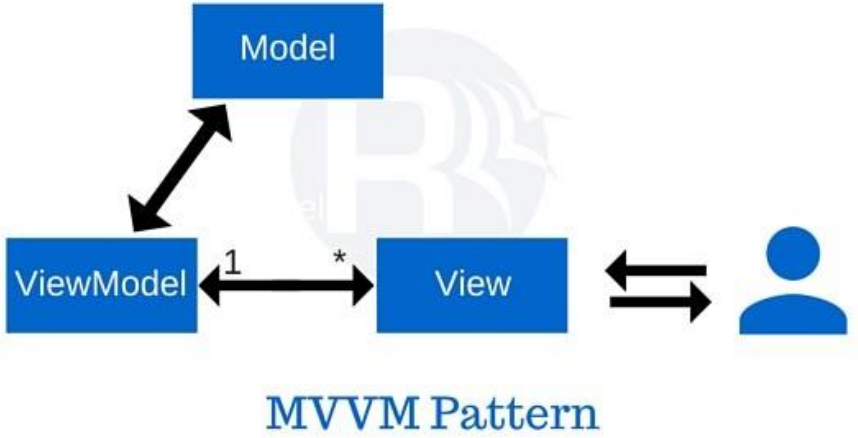


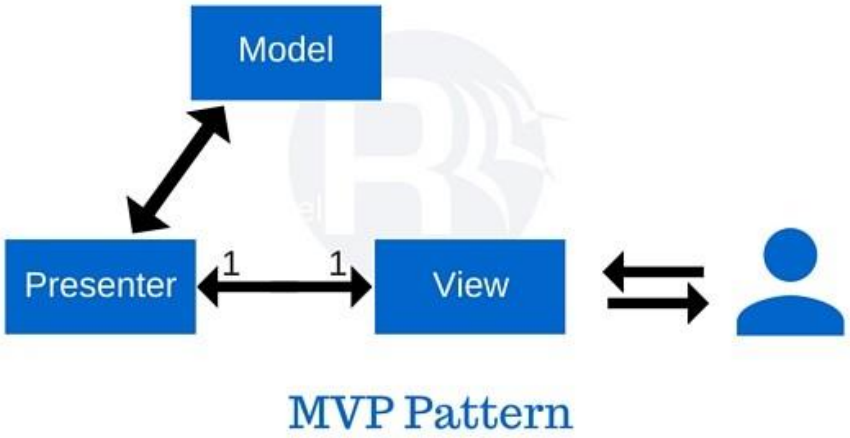
MVC - alternatives

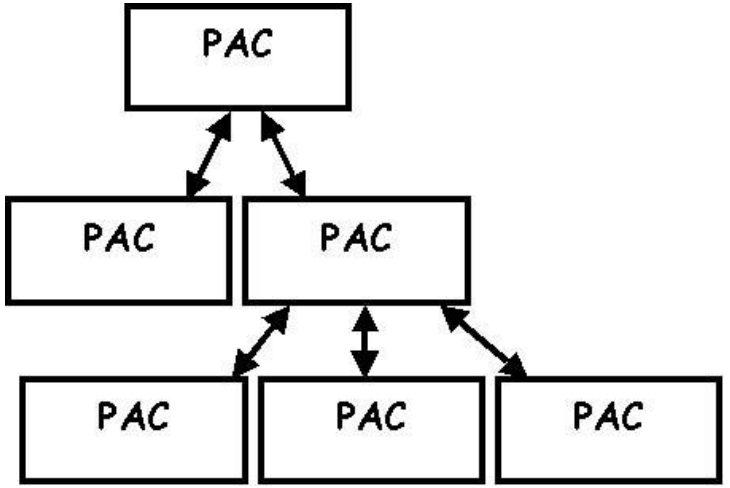
- There are many alternative implementations for MVC.
  - MVP – Model-View-Presenter (IBM)
    - The user interacts with the view, not with the presenter, which is the equivalent of the controller. One presenter per view.
  - MVVM – Model-View-ViewModel (Microsoft)
    - ViewModel can control multiple views.
  - MVA – Model-View-Adapter
    - The model and the view are not connected. Multiple adapters for the same model-view pair.
  - PAC – Presentation-Abstraction-Control
    - In case the model is modular. The components are organized in a hierarchy.

MVC - alternatives







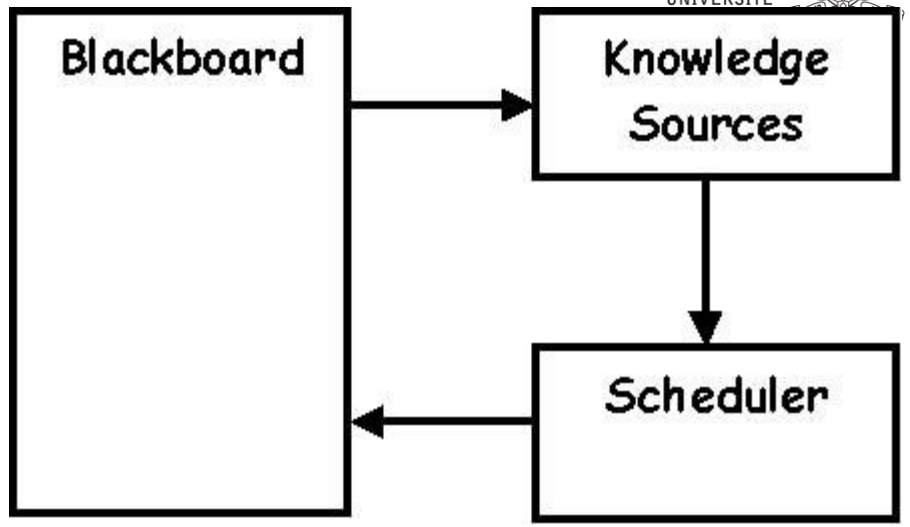


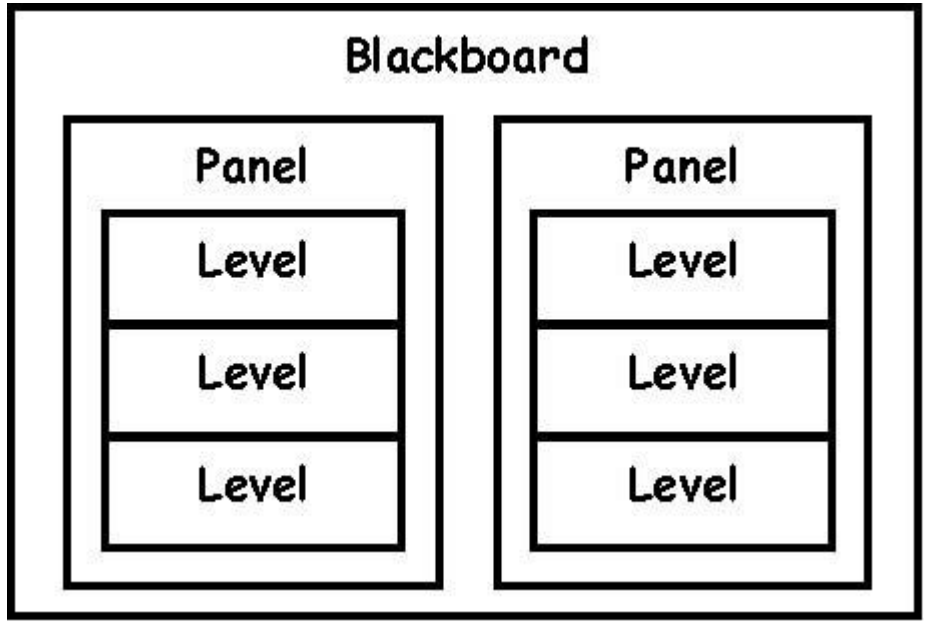
MVC – Advantages and Disadvantages

- Advantages
  - Separation of responsibilities
  - The model can be shared among various views and controllers.
  - There are already many frameworks ready to extend to implement the MVC.
- Disadvantages
  - The architecture can increase the complexity of the system.
    - There is the responsibility for multiple updates of components, not truly necessary.
  - The multiple alternatives and definitions can cause confusion.

Blackboard

- Imagine you have a large problem on a blackboard and a group of students. Each student can solve a part of the problem. We also have a professor, who decides which student will write on the blackboard. When the student has finished, the process is repeated.
- The blackboard represents a large problem, which can be divided in smaller parts. It provides a structure for the shared data.
  - The structure can be complex with multiple representations, decomposed data and organized on different levels.
- The students are the knowledge sources. They have the expertise to solve a single part.
  - Every knowledge source adds to the solution of the problem.
- The professor is the planner, who decides which expertise is necessary according to the work that is already completed.
  - The professor reacts to changes on the blackboard to make decisions.
- Example: object and speech recognition
  - Radar Defence System (Microsoft): https://social.technet.microsoft.com/wiki/contents/articles/13461.blackboard-design-pattern-a-practical-c-example-radar-defence-system.aspx



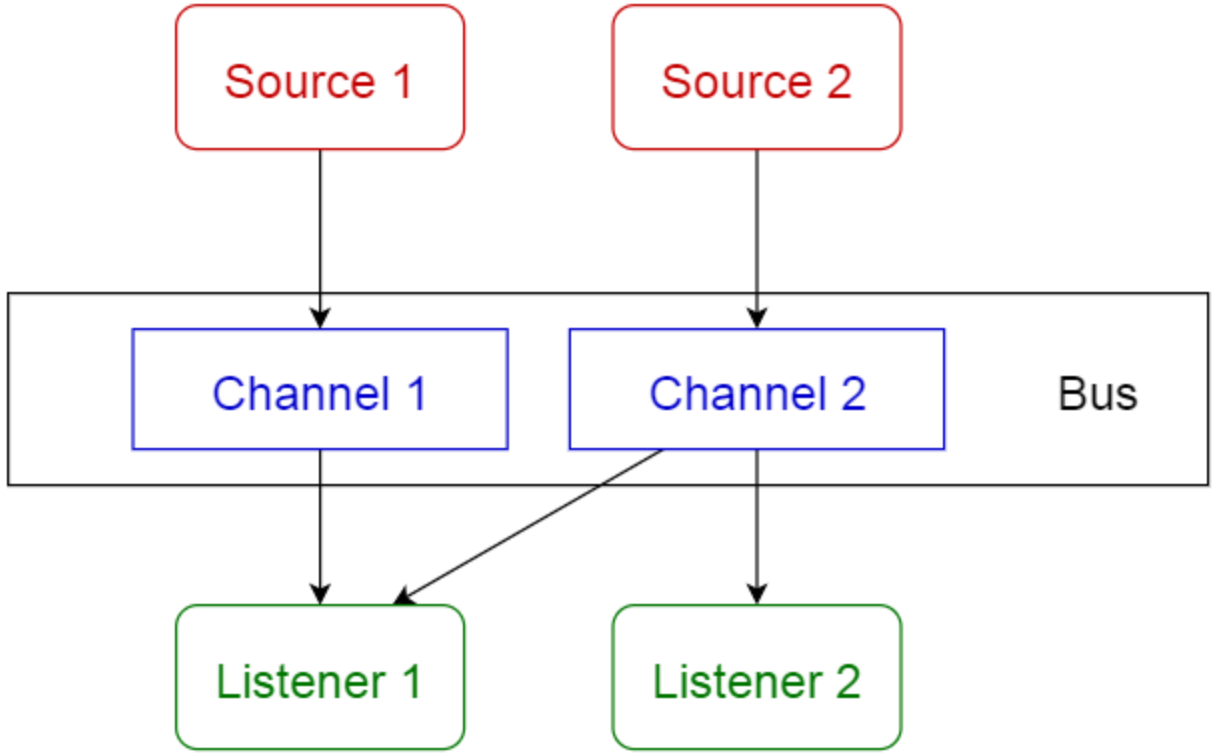


Blackboard – Advantages and Disadvantages

- Advantages
  - They can function with large, but decomposable problems.
  - They can accommodate multiple “expertises”.
  - It is easy to add new knowledge sources and extend the data structures.
- Disadvantages
  - The planner can be heavy and complex.
  - How will we decide the order of execution?
  - The synchronisation and the access control of data are necessary.
  - It is difficult to modify the data structures.

Event-bus

- If we have multiple modules that want to communicate with each other, the event-bus can manage the communication.
- The modules can submit direct messages or wait for messages of specific type (Publish-Subscribe)
- The messages have a header, which can specify the receiver and the type of the message, and a body.
- Examples: notification systems, monitoring, graphical interfaces.
- A bit like the Observer pattern.



Event-bus – Advantages and Disadvantages

- Advantages
  - We can add modules at runtime.
  - The modules are not responsible for delivering the messages
- Disadvantages
  - Scalability problems: There is a risk of overloading, if there are many messages for a single bus.
  - Synchronisation is an equally critical consideration.

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Distributed Architectures

Distributed Architectures

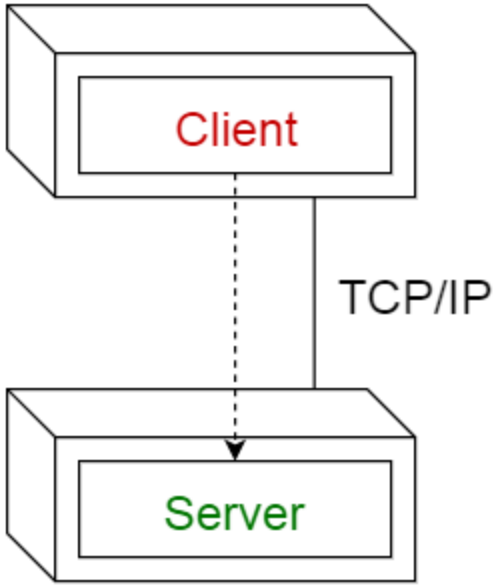
- They are also modular architectures, but in this case, the modules are distributed among more providers and are located in different places.
- The importance of the network is critical.
- The development is shared among the various organizations, which has an impact on the economic aspects of the software.
- The software becomes more flexible and dynamic.
- The technologies advance rapidly to respond to challenges posed by these architectures.

Distributed – Advantages and Disadvantages

- Advantages
  - Opacity: Many details are hidden, which facilitates the development.
  - Sharing of resources (hardware and software).
  - Openness: Increased flexibility to use various providers of hardware or software.
  - Concurrent processing to improve performance.
  - Scalability: Increased capacity by adding new resources.
  - Fault tolerance: Capacity of continuing to function after the detection and the resolution of a fault.
- Disadvantages
  - Increased complexity
  - Reduced security
  - Increased management effort.
  - There may be unpredictable responses and effects from the parts of the system.

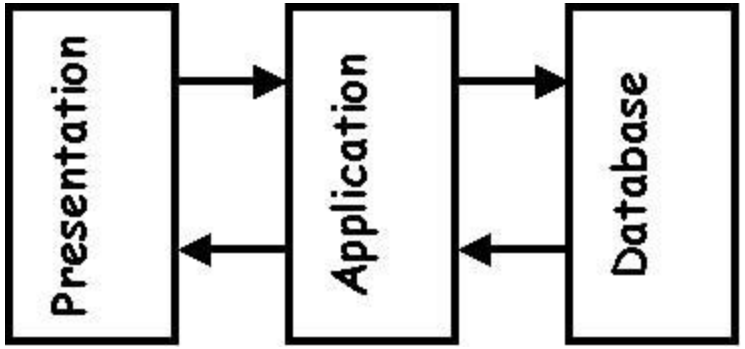
Client – server

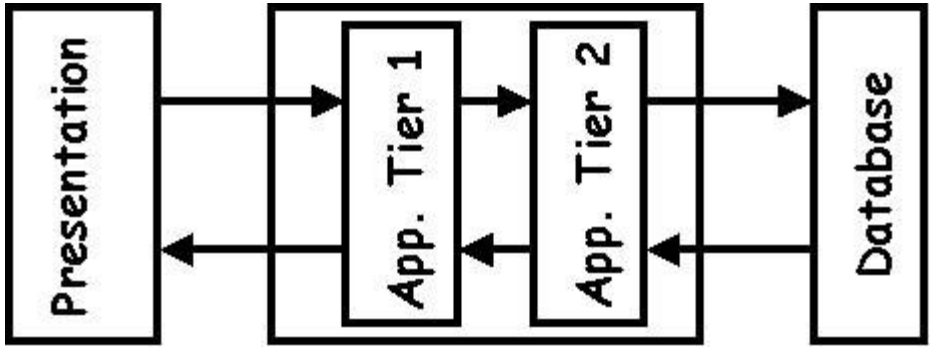
- Two entities:
  - A server, which provides the functionality (a service) and a client, which is a consumer of the service.
  - The client makes requests and the server responds.
- The server can serve multiple clients.
- The communication occurs over a network.
- Two possible types of clients:
  - Thin-client: Just a graphical interface. No logic. Terminal.
  - Thick-client: A graphical interface and some part of the logic. Browser.
  - Examples: Practically all web applications.
- The fundamental architecture of the Internet.
- We cannot really talk about advantages and disadvantages, because the architecture is really generic.
  - More specific styles have more concrete advantages and disadvantages.



Multitier

- It’s an extension of the client-server architecture.
- In place of two levels (client, server), we can have more.
  - Presentation (interface), Application (more levels), Database.
- Like the layered architecture, but:
  - The levels can communicate independently with each other.
  - The levels can be distributed over many computers or even many owners.
- The modules are developed completely independently and they are very specialised.
- Examples: E-commerce applications, data analytics, machine learning.



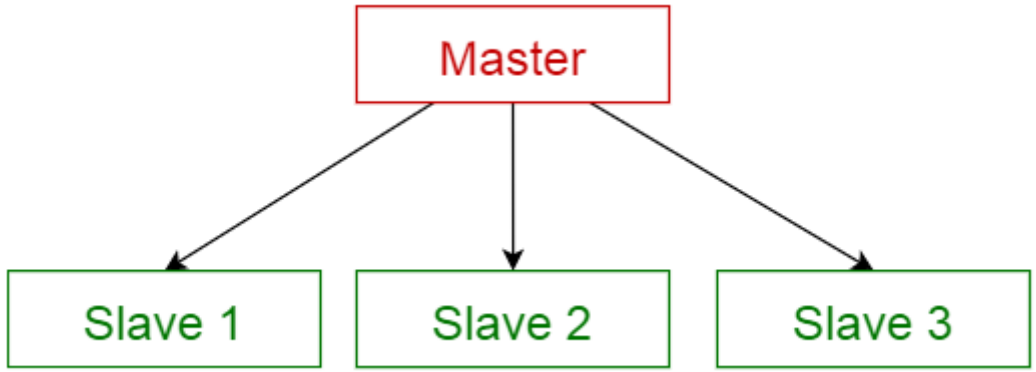


Multitier – Advantages and Disadvantages

- Almost the same as the layered architecture.
- Advantages
  - The relevant technologies exist for a long time and they are robust.
  - The modules are reusable. They are not just autonomous, but completely independent.
  - Dynamic switching of modules is possible and easy.
- Disadvantages
  - The modules can be specific to particular technologies and this reduces flexibility.
  - The functionality and the availability depends on the network.
  - Security becomes complex and reduced.

Master-slave

- Another architecture of two levels, but the roles are different.
- We have a large task, which can be divided and shared.
- The “master” is responsible for dividing the task and distributing the parts to the slaves.
- The “slaves” execute the task on parts of the data. In fact, the slaves have identical programs.
- In the end, the master assembles the results and saves them in a database.
- Examples: MapReduce, distributed databases (NoSQL)

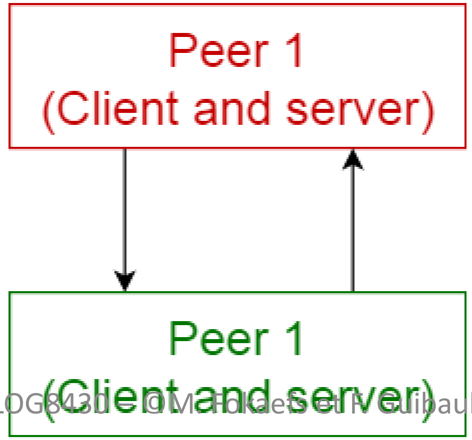


Master-slave – Advantages and Disadvantages

- Advantages
  - The parallelisation increases the efficiency.
  - The presence of the master increases the reliability of the system.
  - The result is guaranteed.
- Disadvantages
  - The architecture is sensitive on the reliability of the master.
    - There are solutions that employ multiple masters (backup).
  - The slaves are isolated. There is a risk of redundancy.
  - The architecture is unusable without enough computation capacity.

Peer-to-peer

- The data is not centralised, but everyone can have all the data.
- The data is shared through direct connections among the participants of the network.
- A central structure is possible, but only for the addresses of the participants or to conserve information on who owns what.
- Examples: file sharing applications

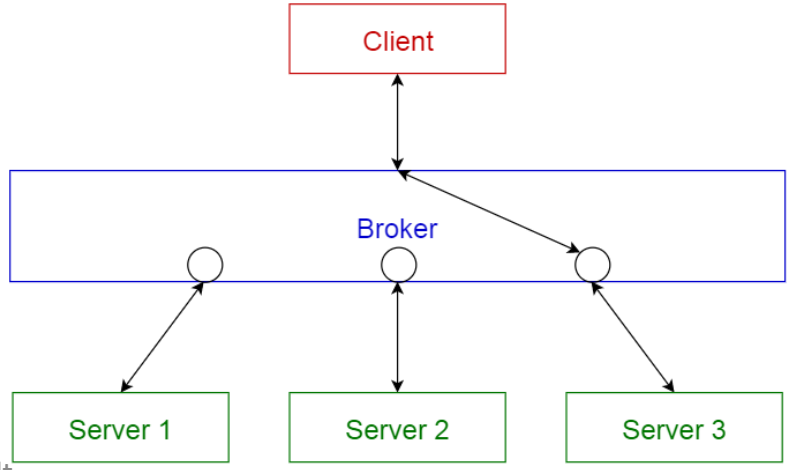


Peer-to-peer – Advantages and Disadvantages

- Advantages
  - The system is robust against failures of the participants.
  - The architecture offers increased scalability.
- Disadvantages
  - The quality, the performance and the security are not guaranteed.

Broker

- We know everything we want to do, but we do not know who can do it and how we can contact them.
- The “broker” is a discovery service, which knows all the providers and their services, whose description is public.
- The clients submit their requirements to the broker and the broker is responsible to find the service that satisfies the requirements and to establish a direct connection between the client and the service.
- It is necessary for the servers and the clients to submit the functionalities of their services and the requirements respectively in a specific format.
- This is the predecessor of the service-oriented architecture.



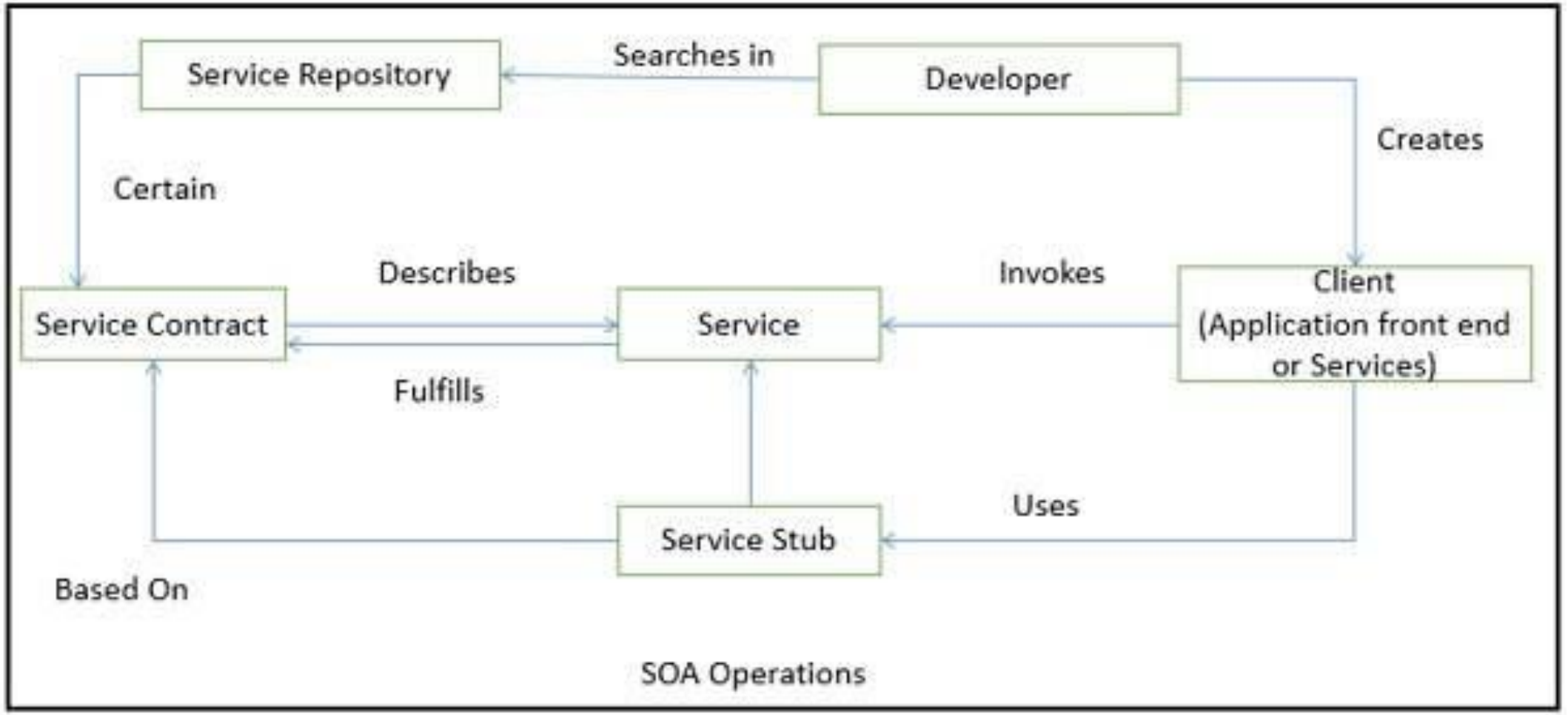
Broker – Advantages and Disadvantages

- Advantages
  - It is possible to dynamically add, modify or remove servers.
  - The distribution is opaque to the developer.
  - The server and the client are decoupled.
- Disadvantages
  - The transactions and the exceptions need to be managed.

Service-oriented architectures

- Everything is a service!
- The services are autonomous and atomic software.
- They make possible the exposure of expertise (functionality) and data.
- SOAP (Simple Access Object Protocol) was developed to replace the old broker technologies.
  - In fact, it replaced the simple XML format for messages.
- The protocol is accompanied by multiple standards.
  - For security, composition, semantic web.
- The service is published as a XML interface.
  - The interface specifies the available functionality and the data types.
  - The service level agreement (SLA) specifies the non-functional properties.
- The client uses the interface to create a “stub” in the language of the local application to invoke the service.

Service-oriented architectures



SOA – Advantages and Disadvantages

- Advantages
  - The technology promotes the composition, reuse and interoperability.
  - The clients do not depend on a specific technology.
    - The service can be implemented in Python and the client in Java.
  - There is an abundance of options and it is possible to switch services dynamically (in theory).
  - There is also an abundance of automated tools to facilitate development and maintenance of services and clients.
- Disadvantages
  - SOAP is too heavy.
  - Many technologies never actually worked.
    - The automatic service discovery, the universal service directory (UDDI) etc.

Representational state transfer

- Everything is a resource!
- In theory, REST services expose just the data and the data structures (uniquely identified) and allow simple operation over these structures (GET, PUT, POST, DELETE).
- The technology uses many data formats (text, XML, JSON).
- An explicit interface is not necessary.
- A communication over HTTP is required.

SOAP vs REST

SOAP

- Standardized
- Language, platform, communication protocol independent.
- Increased support for tools.
- Included management of faults and exceptions.

REST

- Easier
- Simpler
- More performant

Next time

LOG2410

- OO Design (SOLID)
- Design Patterns