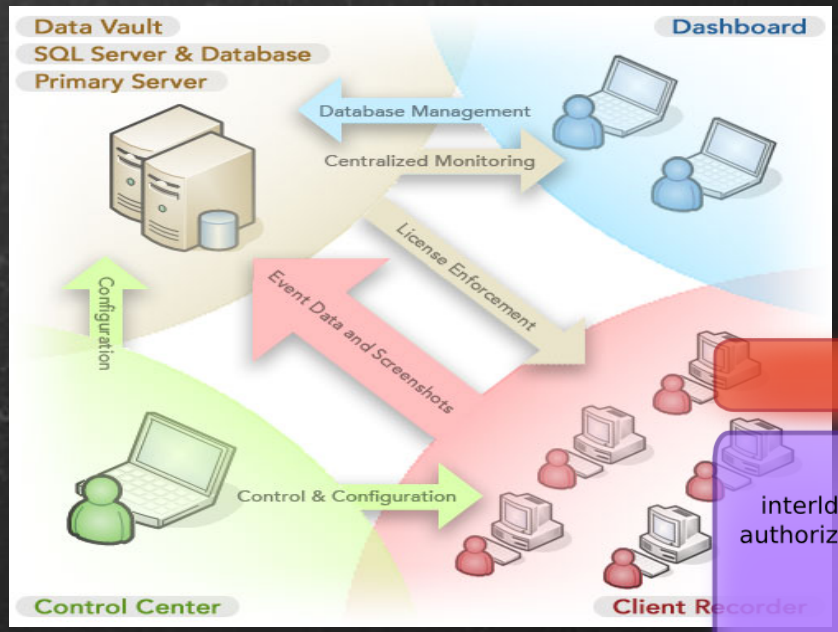
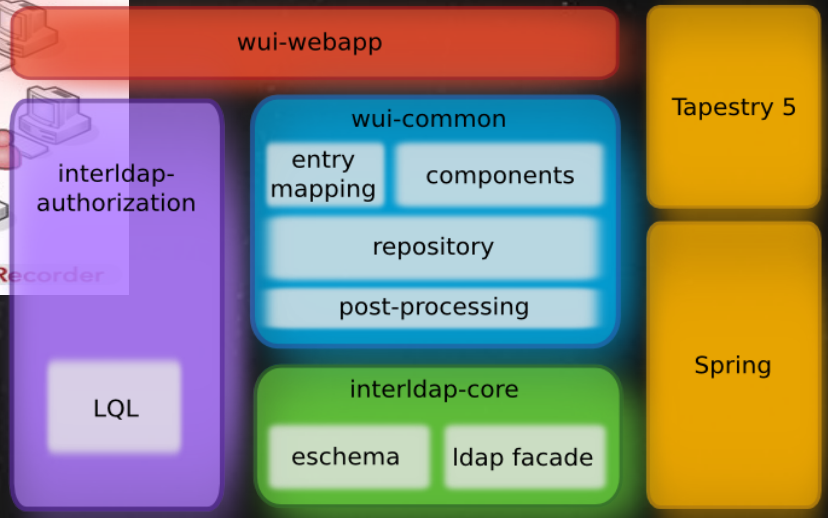
Software Architecture





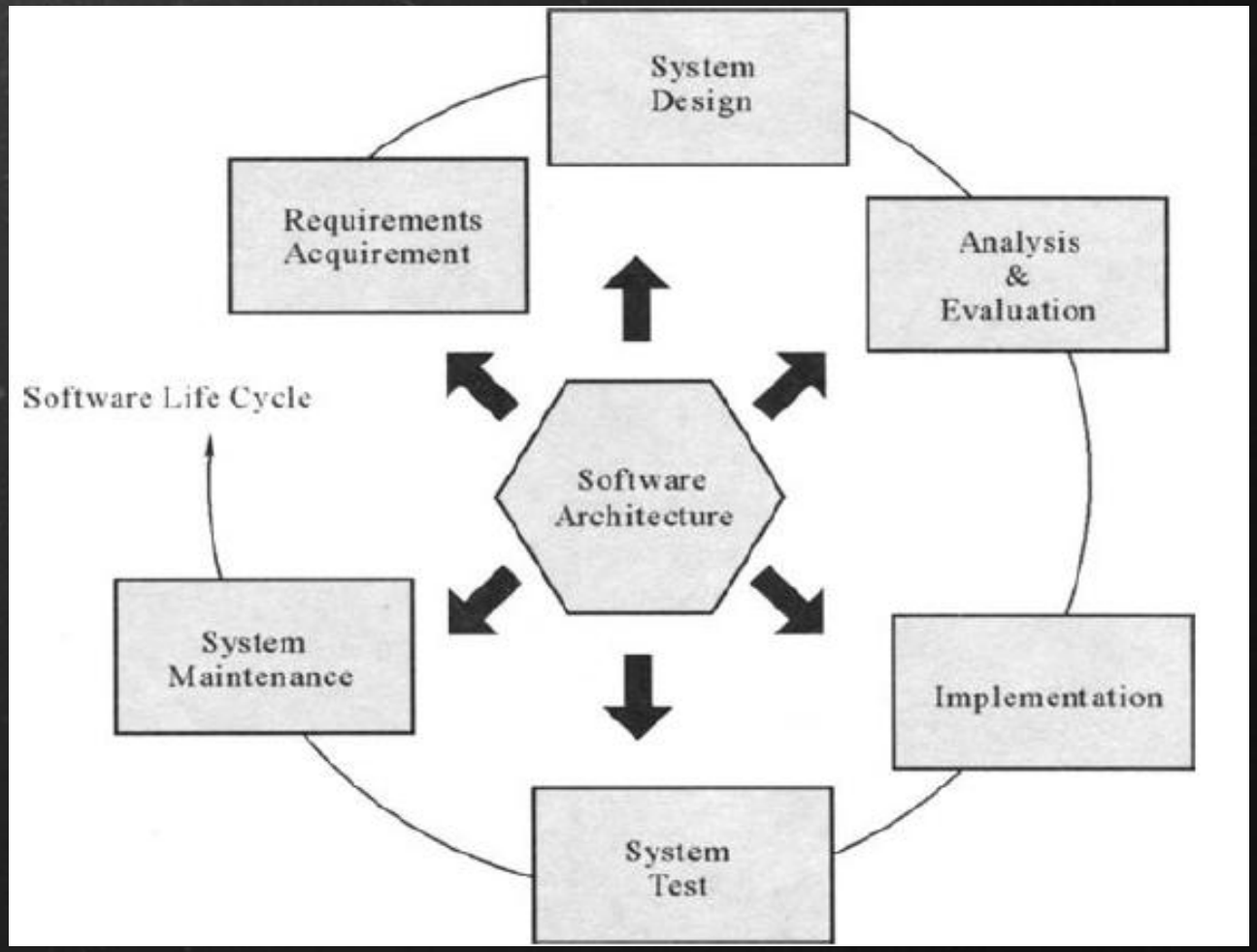
Nityashree Tumkur
Samyukta Mudugal

What is Software Architecture?

It is the structure of the system which consists of software components, the externally visible properties of those components and the relationship between them.

Features

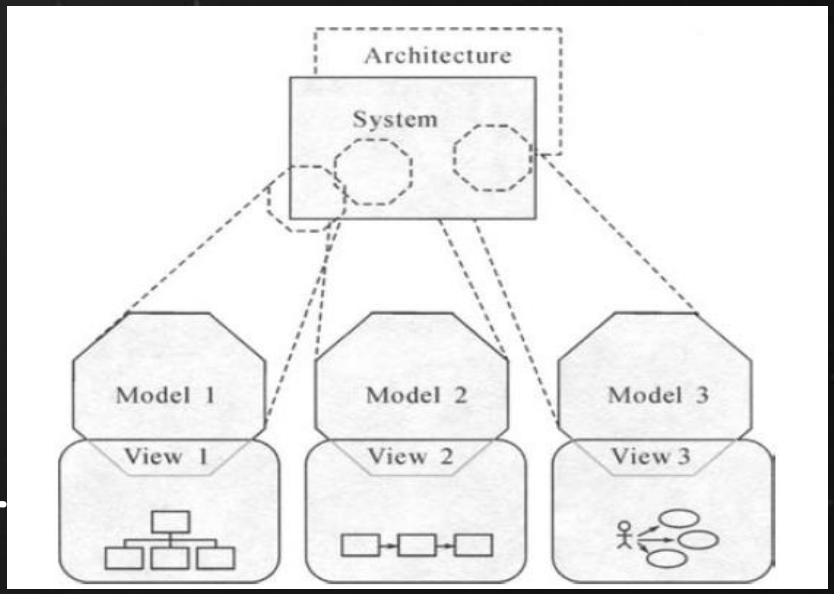
- Every System has its own architecture but they are not identical.
- Software architecture and its description are different.
- The different stakeholders are
  1. Users of the System
  2. Acquirers of the System
  3. Developers of the System
  4. Maintainers of the System



Architecture Centered Life Cycle

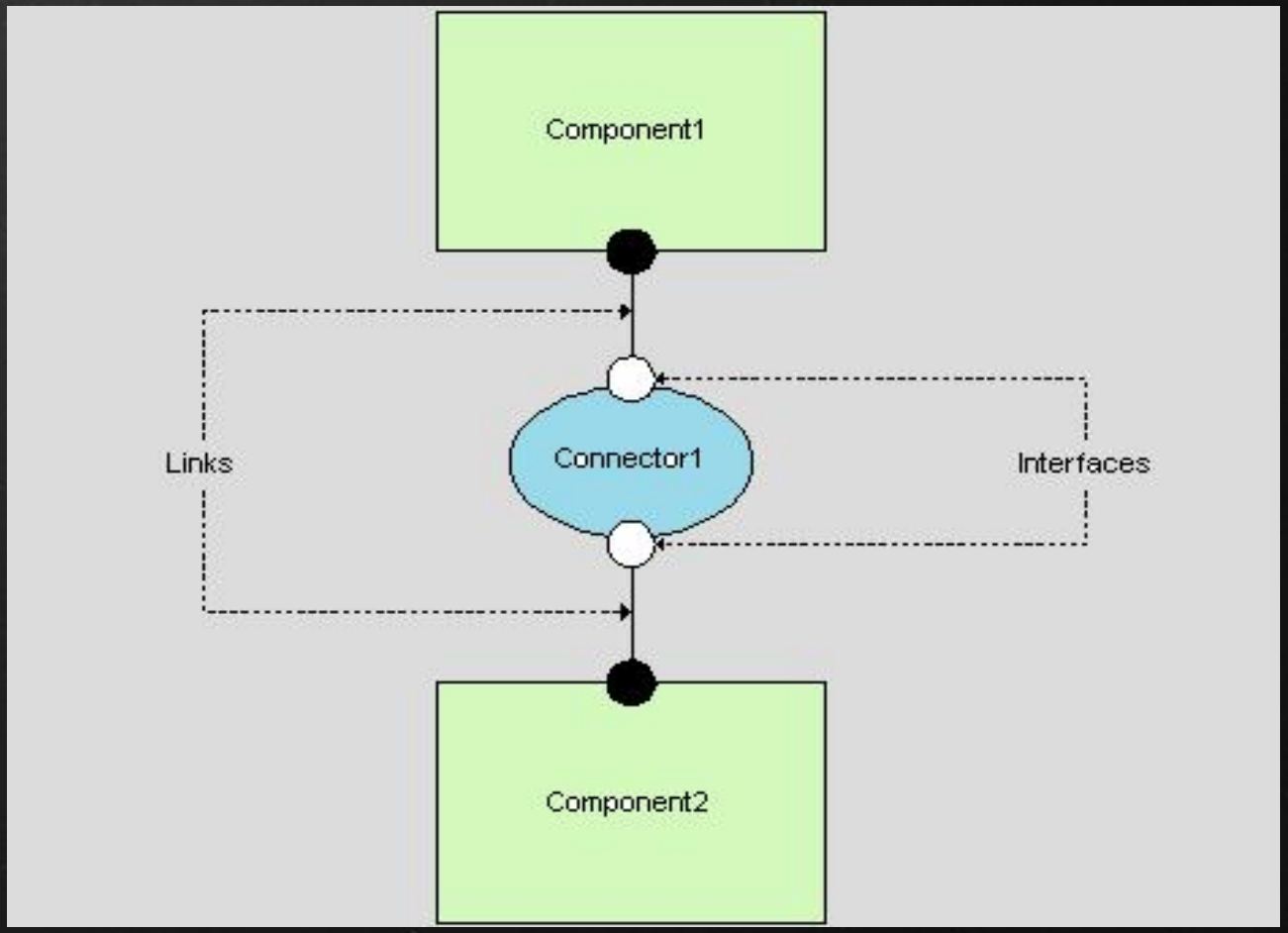
Views Used in Software Architecture

- Software architecture is organised in views which are analogous different types of blueprints made in building architecture.

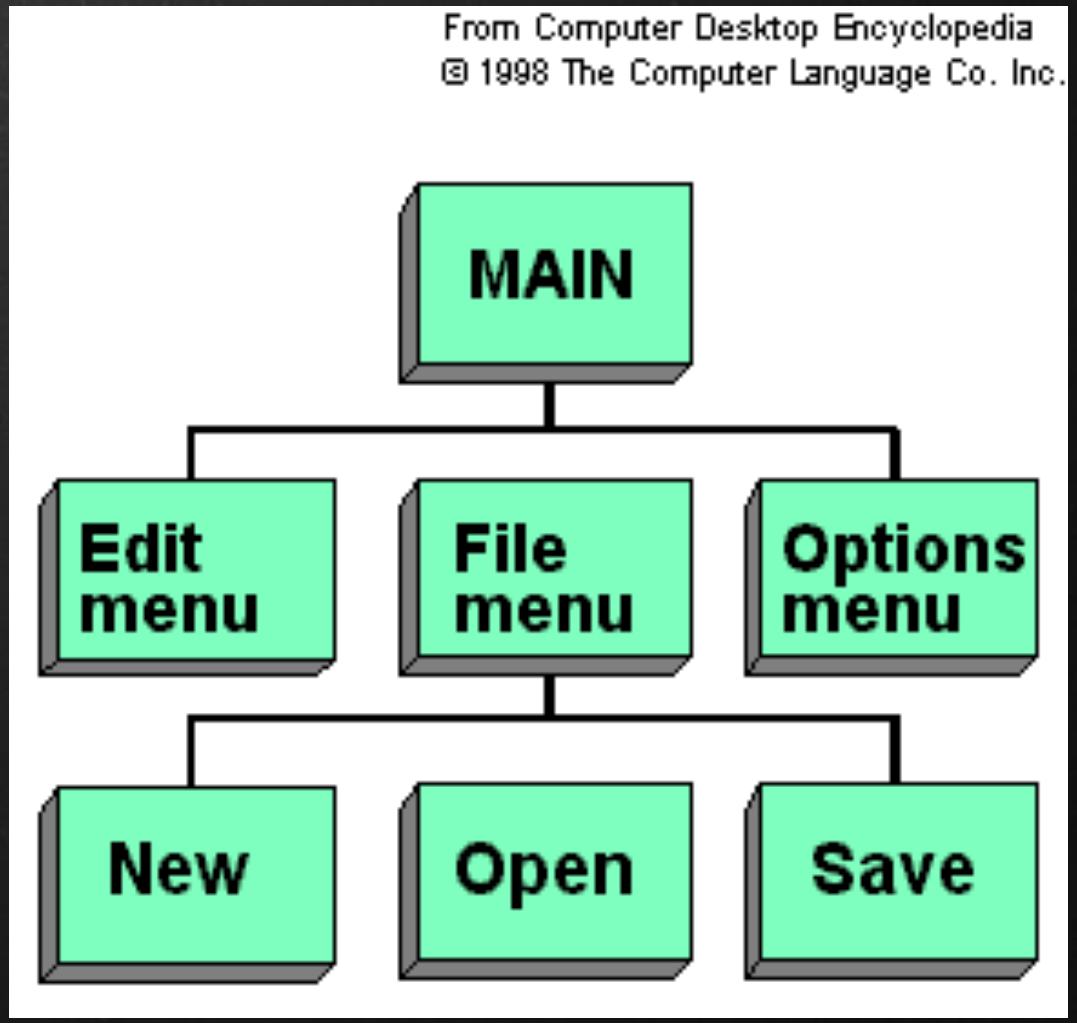


- Different views are:
  - Component and Connect
  - Decomposition view.
  - Allocation view

Component - Connector View

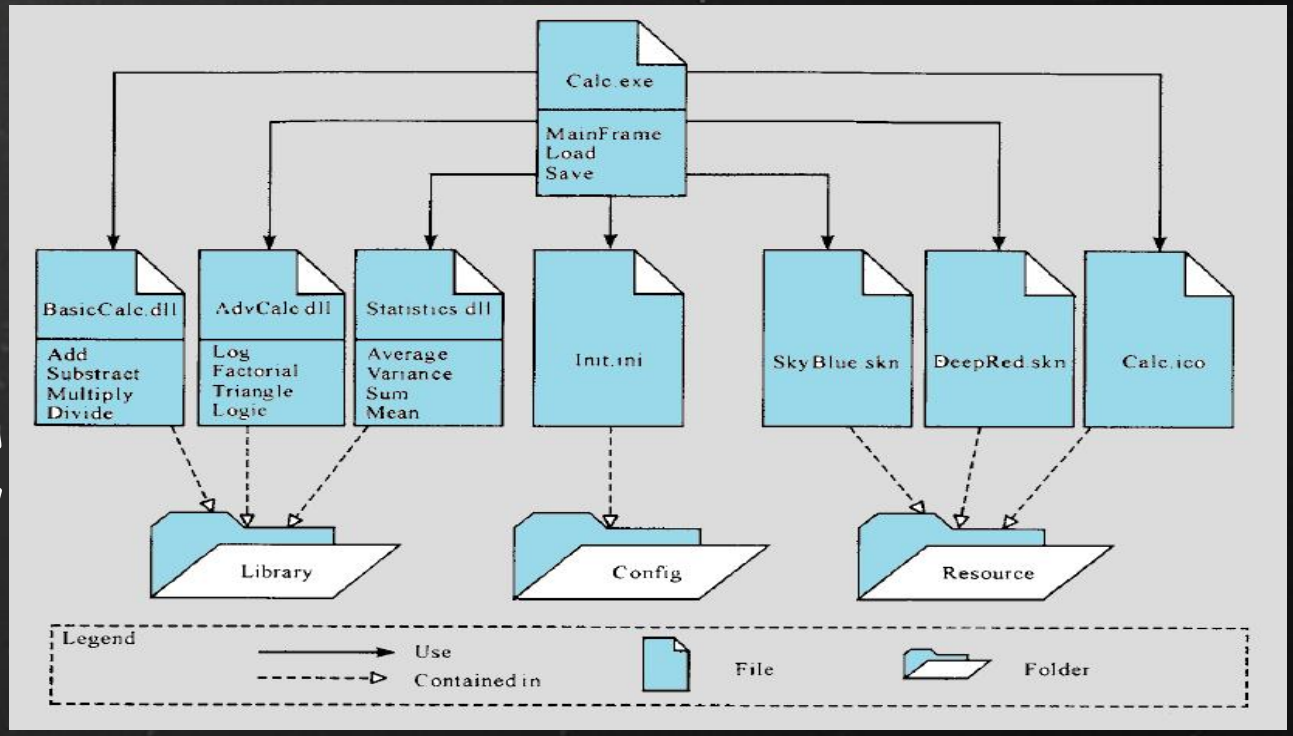


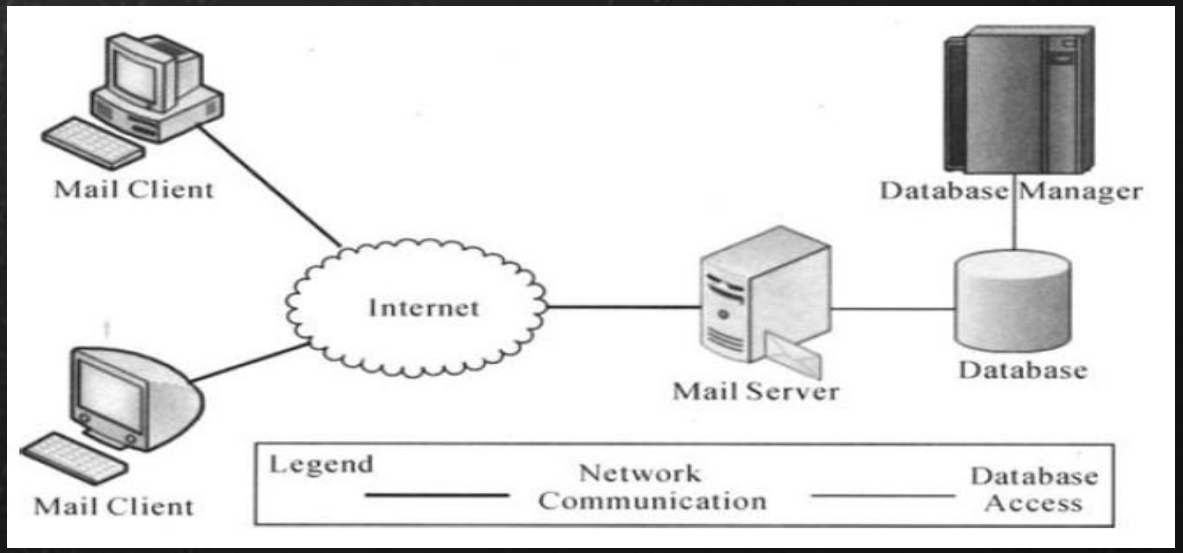
Decomposition View



Allocation View

Implementation view





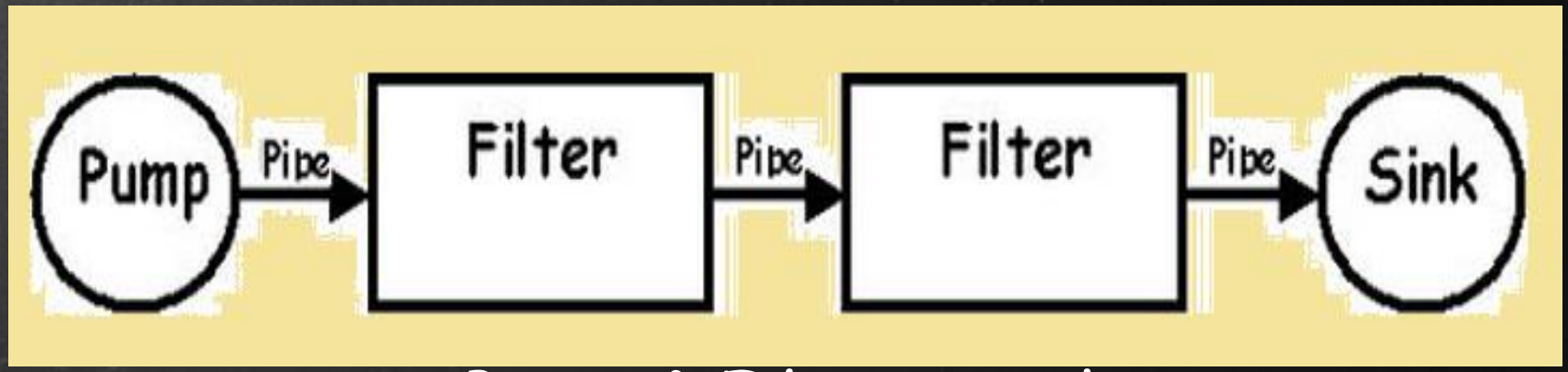
Deployment View

Architectural Styles

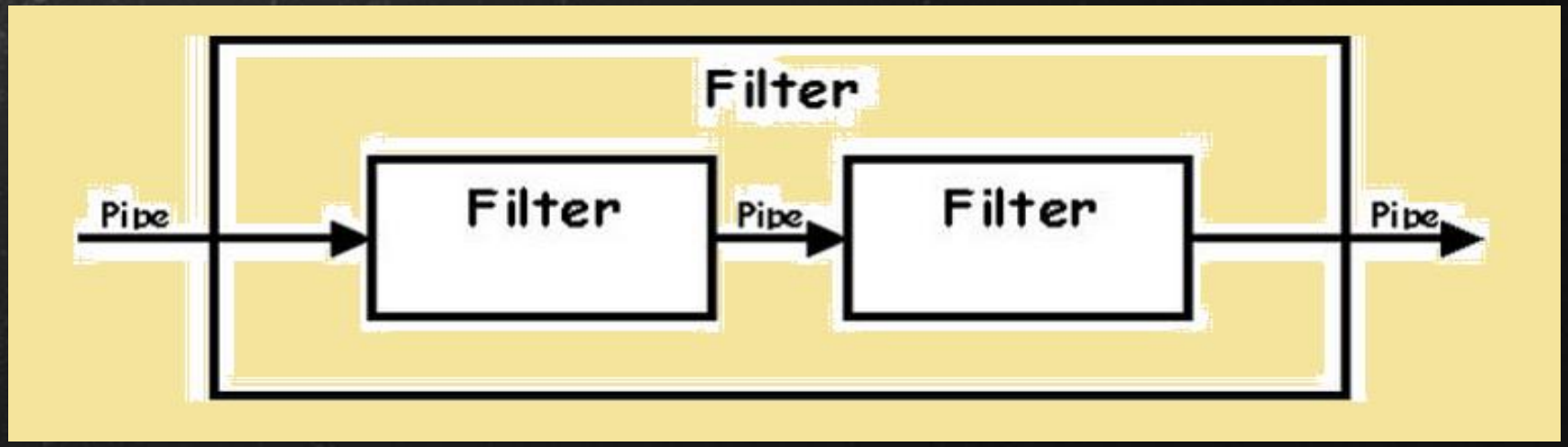
- Pipes & Filters
- Client- Server
- Event Driven
- Hierarchical Layer
- Data Sharing
- Object Oriented

Pipes & Filters

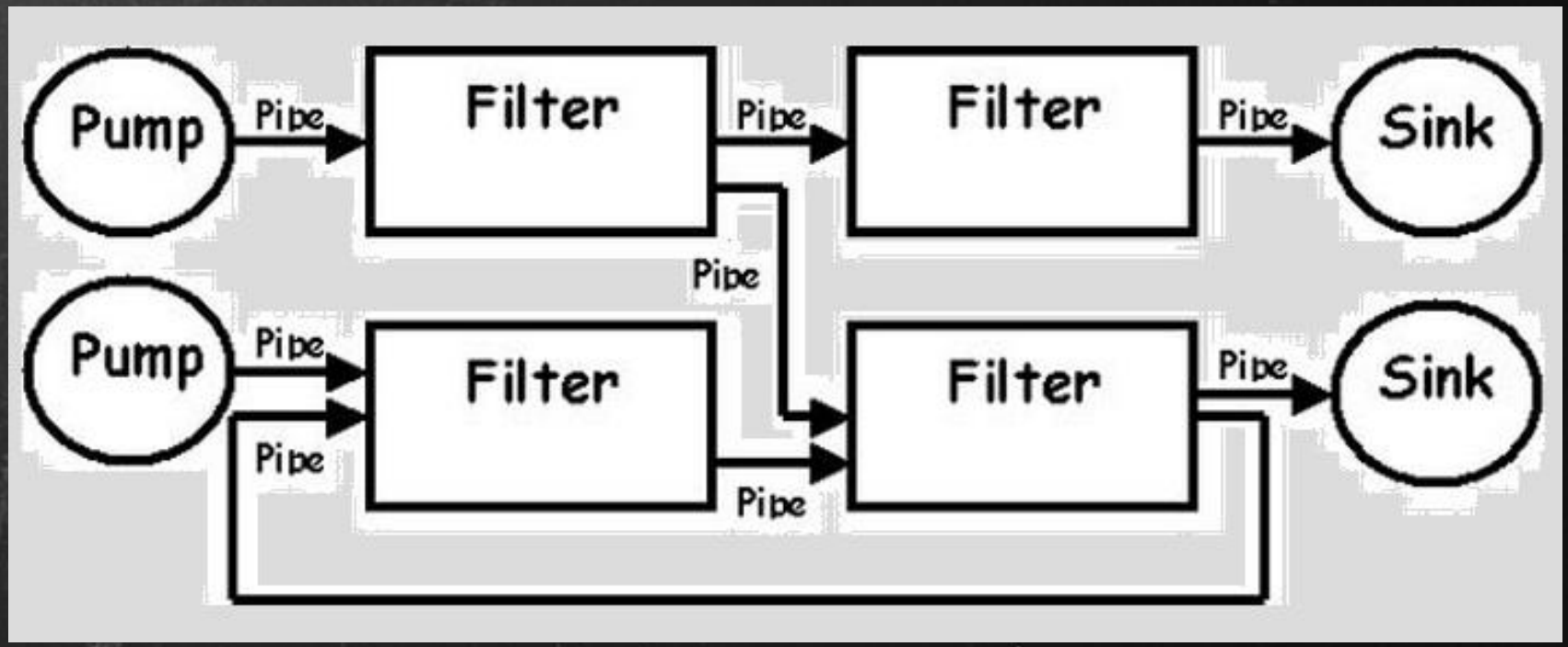
- Very Simple yet powerful and robust architecture.
- Examples:
  1. Unix Programs
  2. Compilers
- Components
  1. Pipe
  2. Filter
  3. Pump
  4. Sink



Pipes & Filters Style



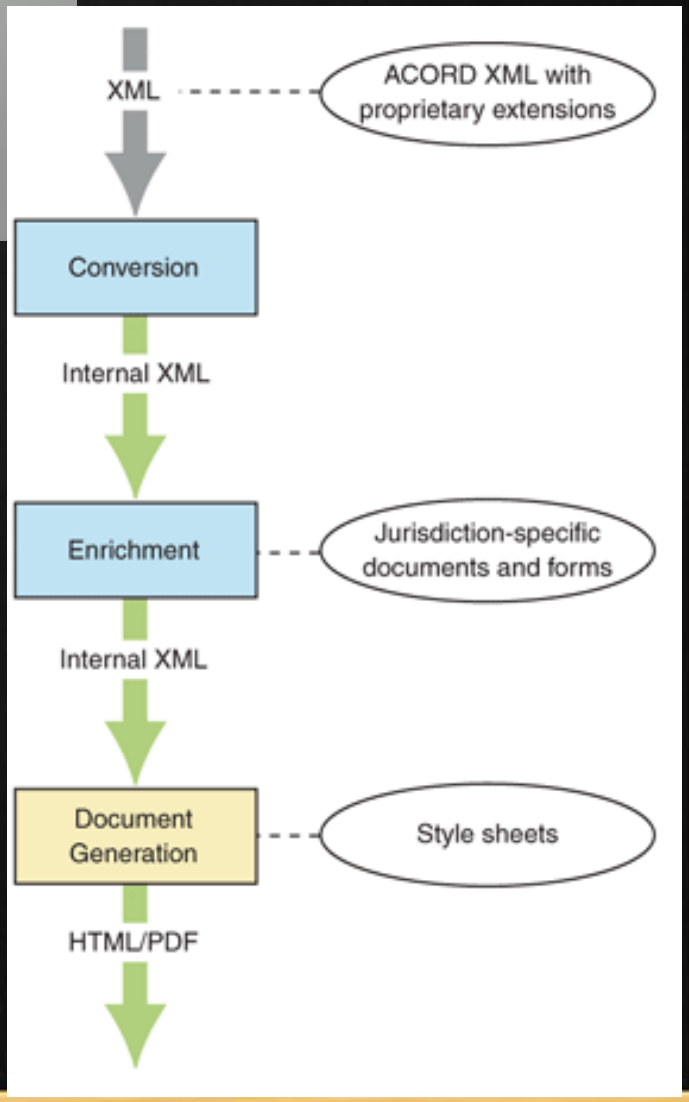
Recursive Technique

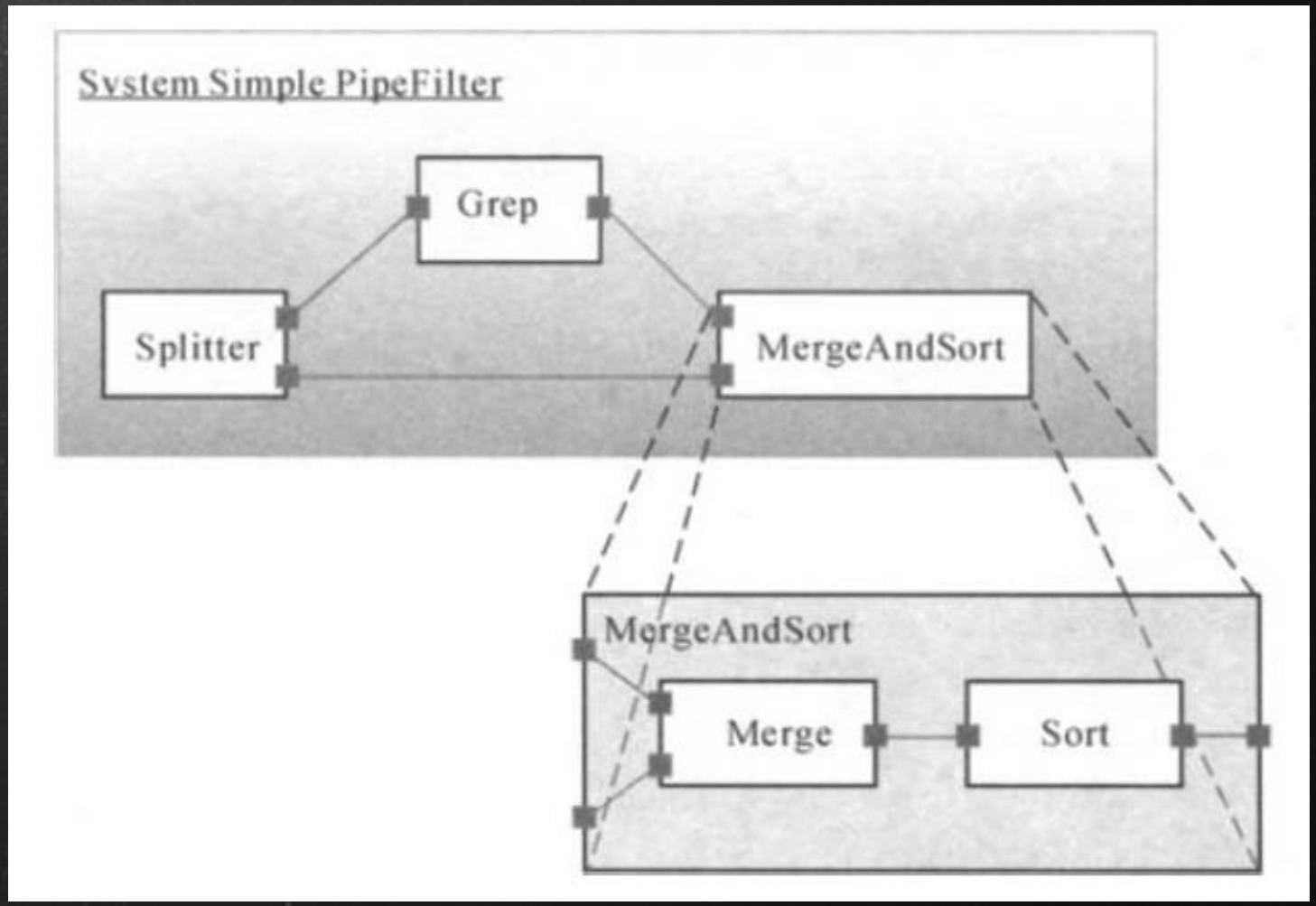


Relationship between different filter processes.

Digital Communication System

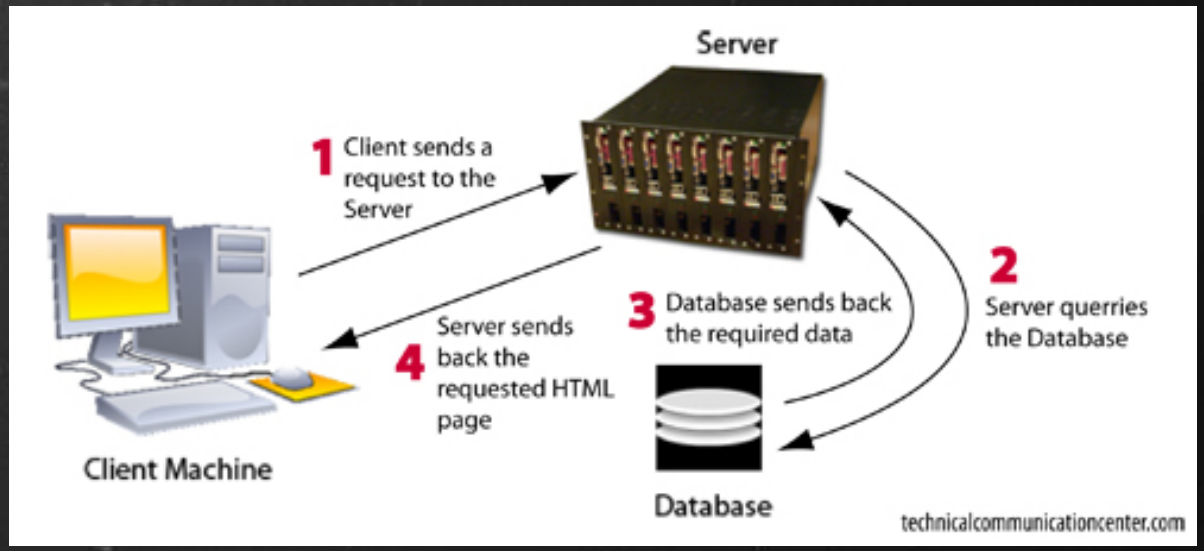
Web Application





Another Example

Client Server Style



Distributed Application Architecture that partitions the tasks into service providers and service requesters

Advantages

- Roles and responsibilities of computing systems to be distributed among independent computers known to each other only through the network.
- All the data is stored in the server which have better security controls.
- Caters to multiple different clients with different capabilities.
- Data updates are easier and faster as Data is centralized.

Disadvantages

- As the number of client requests increases the server becomes overloaded
- Client - Server Architecture lacks the robustness of Peer to Peer Architecture.

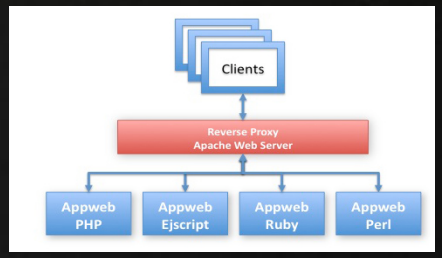
Lets look at this architecture implementation in ACME...

```
System simple_cs ={
    Component client ={
          Port send-request;
          Properties  { Aesop-style : style-id = client-server;
                        UniCon-style : style-id = cs;
                        source-code:external = "CODE-LIB/client.c"}}
    Component server ={
          Port receive-request;
          Properties  { idempotence : boolean = ture;
                        max-concurrent-clients : integer = 1;
                        source-code:external = "CODE-LIB/server.c"}}
    Connector rpc ={
          Roles {caller,callee}
          Properties  { synchronous : boolean  =true
                        max-roles : integer = 2;
                        protocol : Wright = " "}}
    Attachments {
          client.send-request to rpc.caller;
          server.receive-request to rpc.callee}

}
```

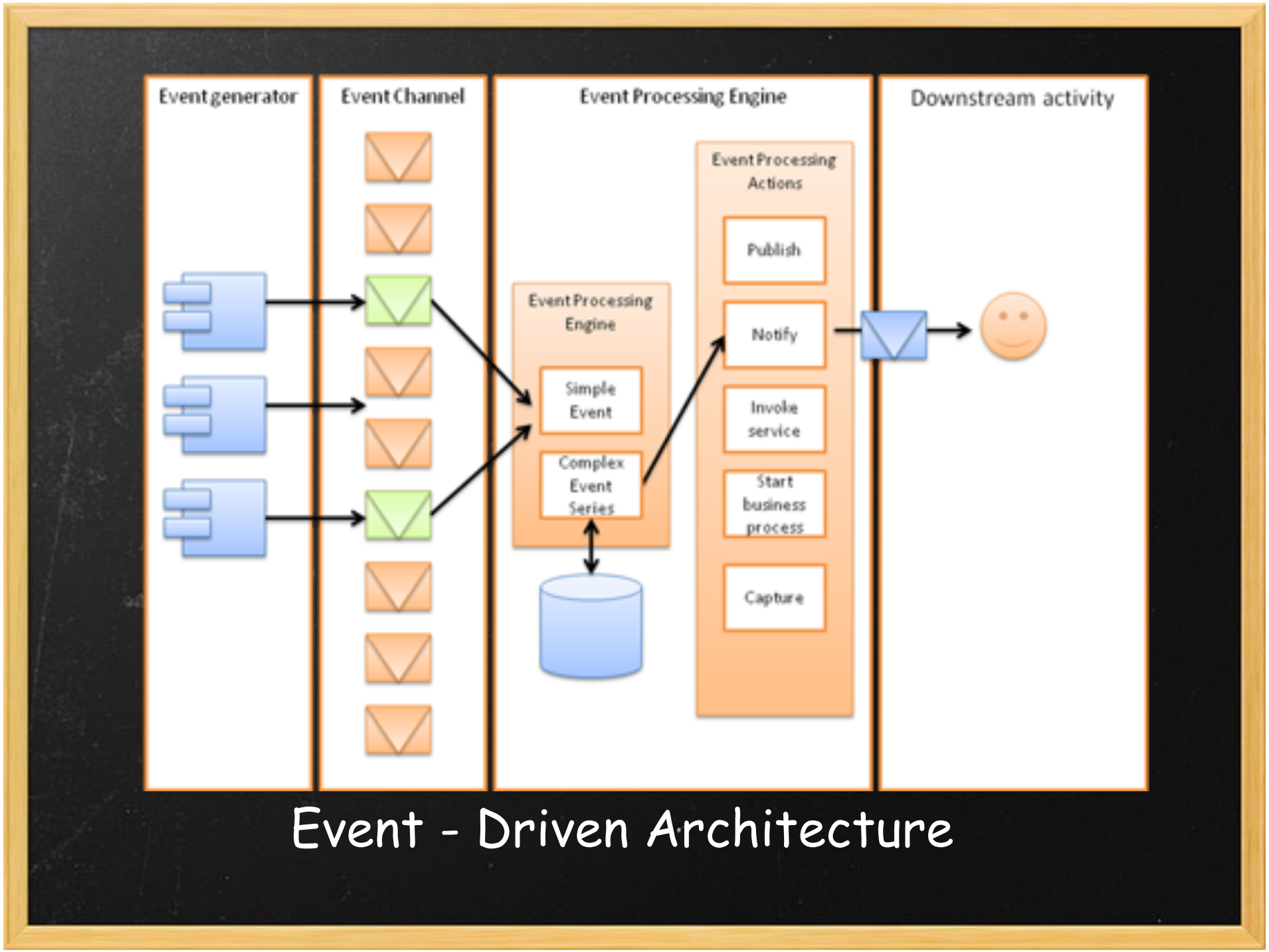
Client - Server in ACME

Event-Driven Architecture



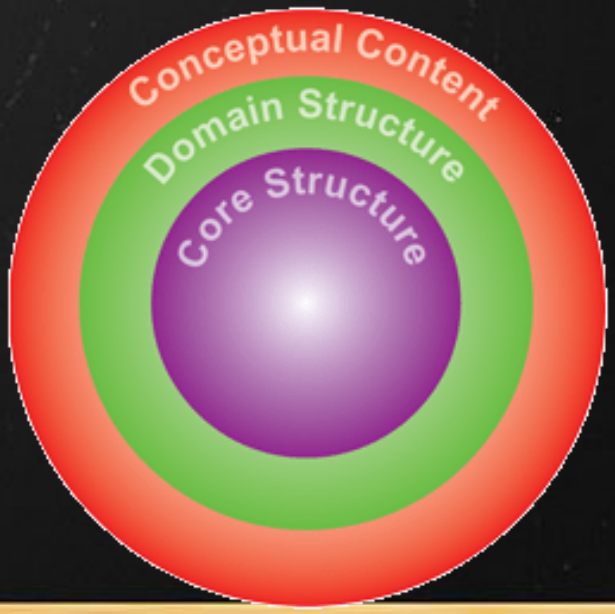
- Architecture pattern that promotes production, detection, consumption of and reaction to events.
- It consists of event emitters and event consumers.
- Sinks have the responsibility of applying a reaction as soon as the event is presented.

Systems have certain goal under the control of some message mechanism and the subsystem collaborates with each other to achieve system's ultimate goal.


Event generator
Event Channel
Event Processing Engine
Downstream activity
Event Processing Actions
Publish
Notify
Invoke service
Start business process
Capture
Event Processing Engine
Simple Event
Complex Event Series
Event - Driven Architecture

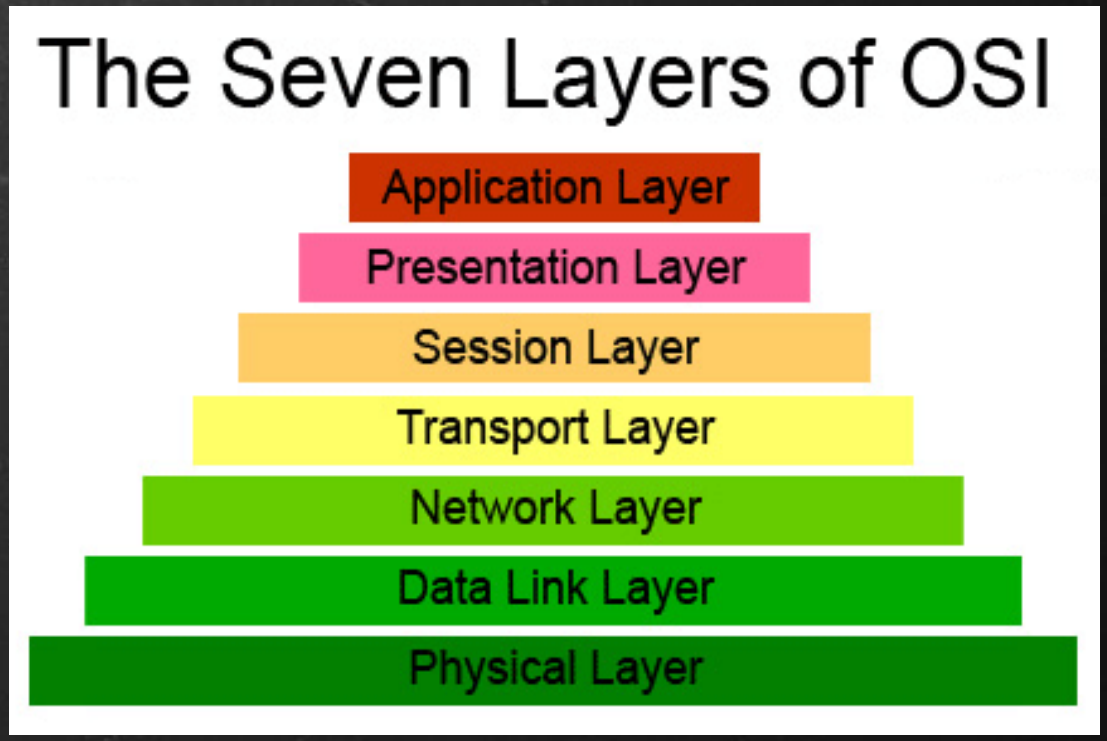
Hierarchical Layer

- It is a layered architecture.
- Each layer has 2 roles:
  1. Provide services for the upper layers.
  2. Call lower layers functions.
- Conceptual layer system model:



Advantages of Layering

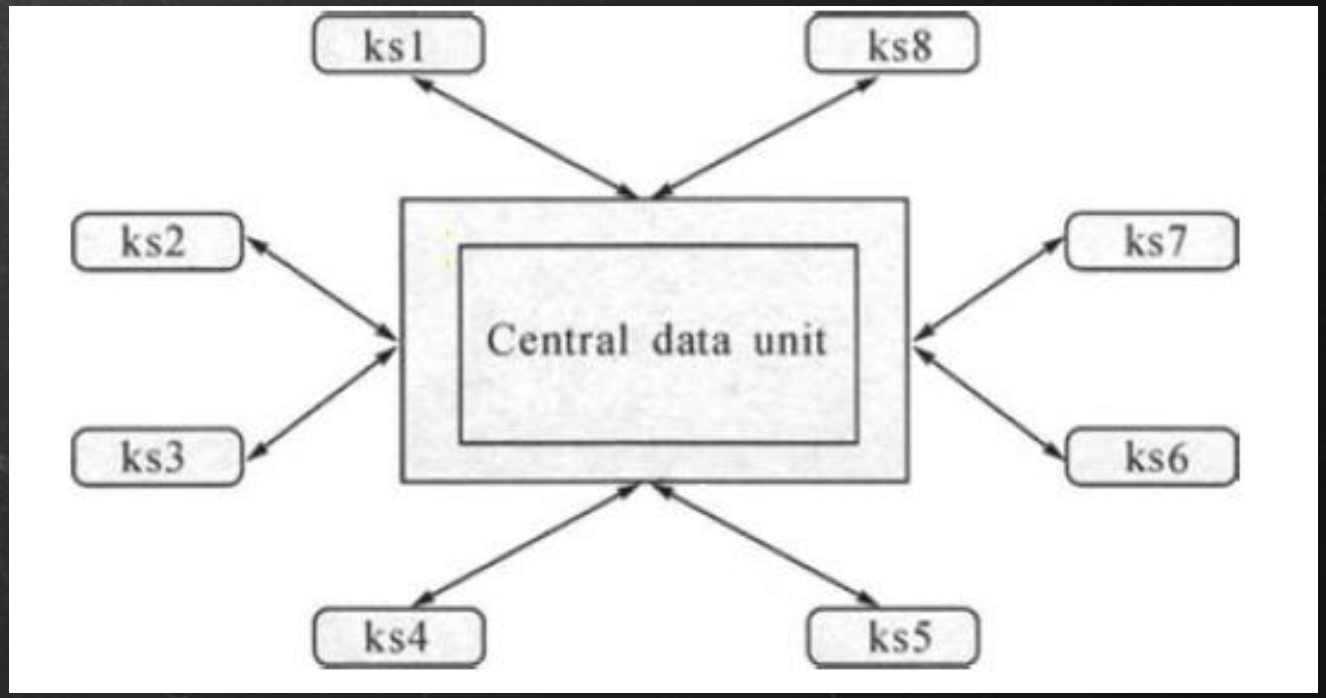
- Supports gradual abstraction in the system design process.
- Layer system has good extendability.
- Layer style supports software reuse.



Example of a layered architecture: ISO/OSI network 7- layer architecture

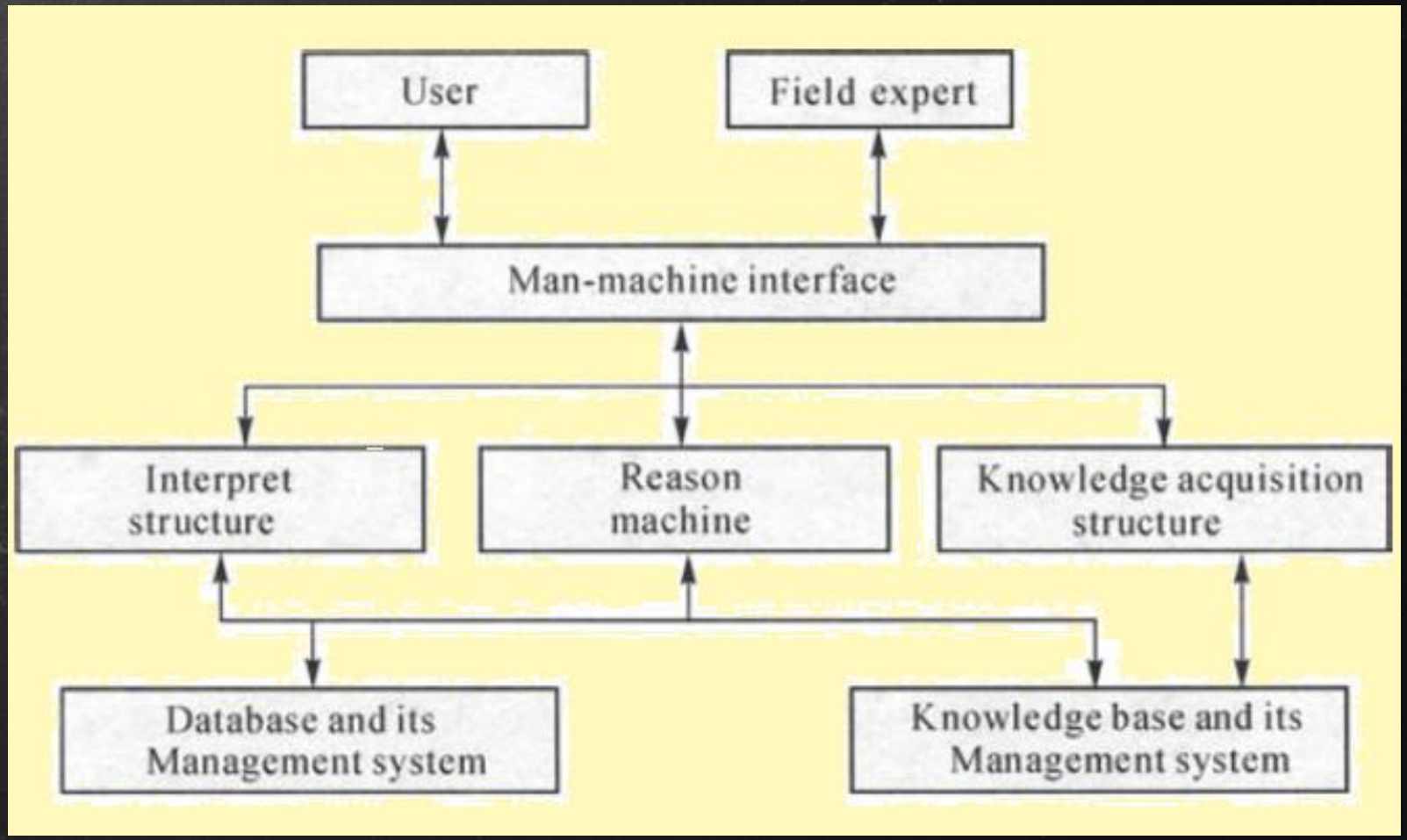
Data Sharing

- Also called repository style.
- System has 2 components:
  1. Central data unit component.
  2. Set of relatively dependent components.
- Central data unit called the repository shares information with all the other units.
- There are differences in the information exchange patterns.
- Thus there are 2 main control stratergies to deal with these information exchange patterns.



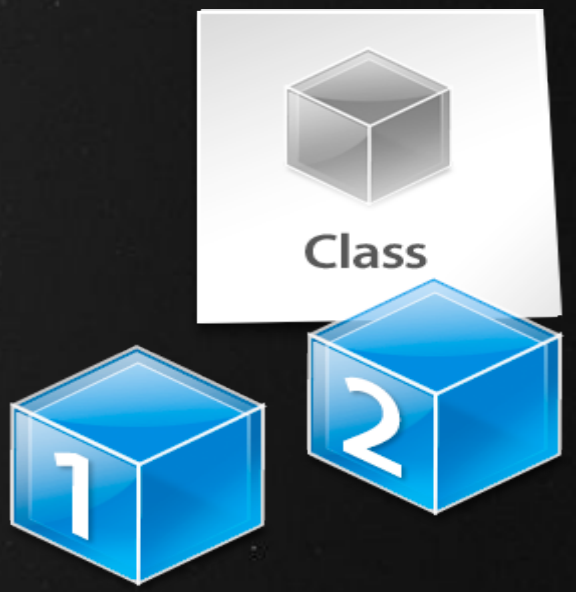
Black-board type repository model

The components:
ks-knowledge sources,
Central Data Unit,
Control Unit.

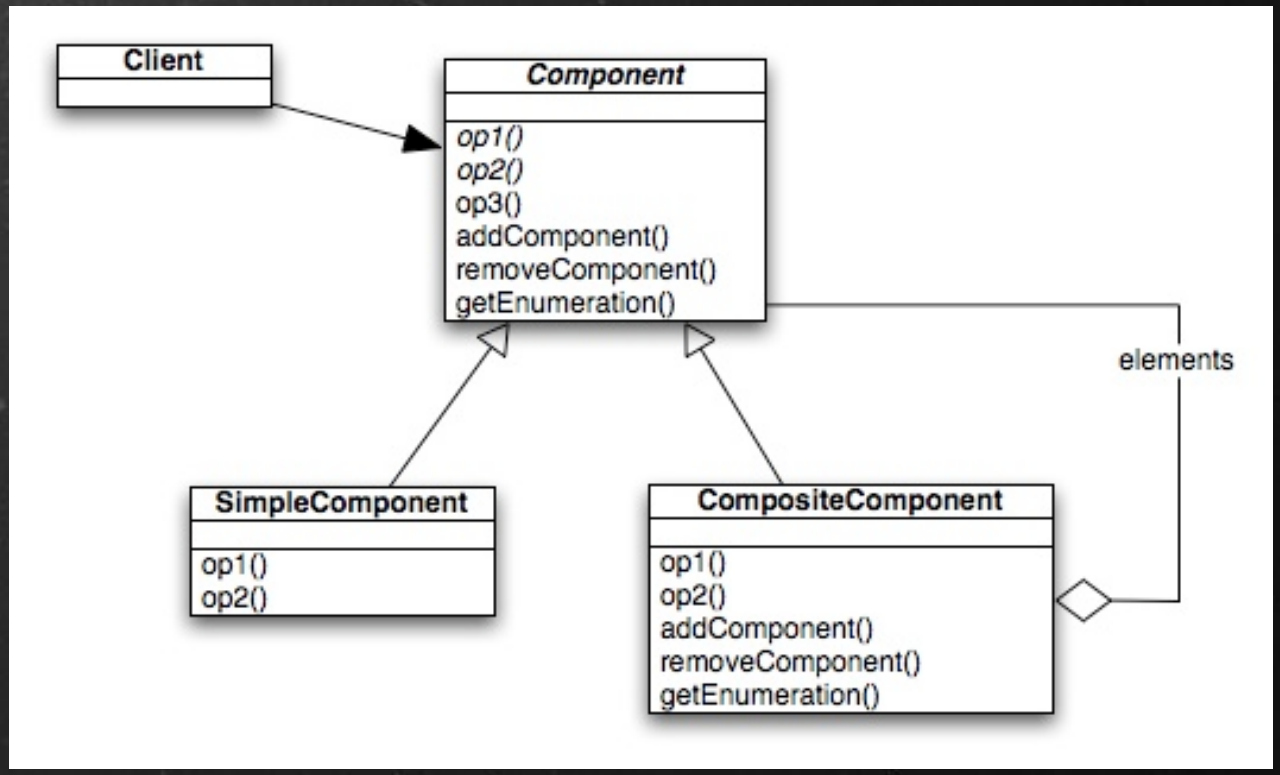


Example: Expert system

Object Oriented



- The key features are:
  - Data Abstraction.
  - Modularization.
  - Information encapsulation.
  - Inheritance.
  - Polymorphism.
- Objects in the problem are first recognized, then proper classes are constructed to represent these objects.
- Java - Object Oriented Programming, C - Procedural programming.



Example of Object Oriented Architecture: Described using a UML diagram.

Architecture Description Languages

- Computer language used to describe the software architecture.
- Shaw and Garland's description for ADL's includes-
  1. Components.
  2. Operators.
  3. Patterns.
  4. Closure.
  5. Specification.
- Different ADL's existing: ACME, AADL, Darwin, WRIGHT.

What makes a language an ADL?

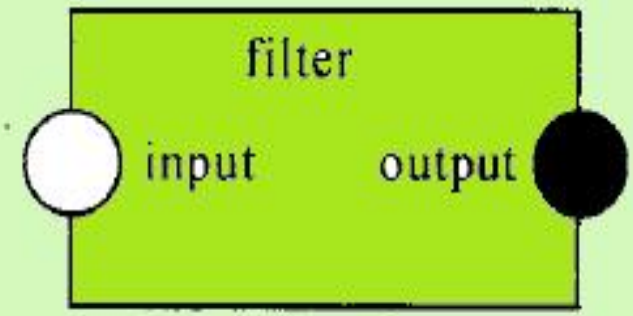
- Be suitable for communicating an architecture to all the stake holders.
- Support the tasks of architecture creation, refinement and validation.
- Provide the ability to represent most common architectural styles.
- Support analytical capabilities.
- Provide quick generating prototype implementations.

Darwin

- Declarative Language.
- Describes the organization of software in terms of components, their interfaces and their binding components between them.
- Provides general purpose notations for specifying the structure of the system.
- Focuses on specification of distributed software system.
- Supports the specifications of dynamic structures.


```
Component Server{
provide p;
}
Component Client{
require r;
}
Component System{
inst
A: Client B:Server
bind
A.r — B.p
}
```

Client Server System in Darwin




```
Component filter{
        provide output<stream char>;
    require input<stream char>;
}
```

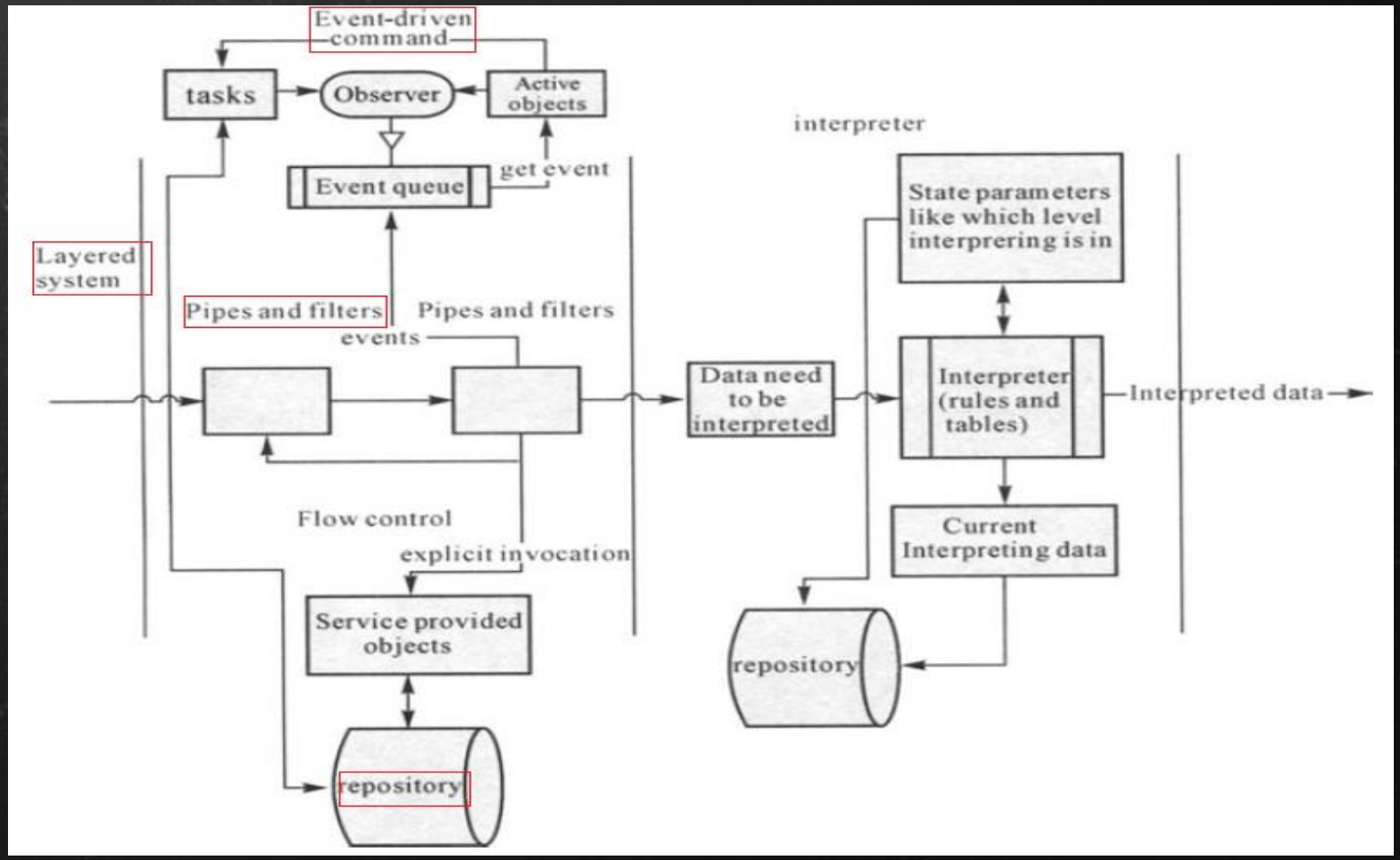
Filter Component in Darwin

Conclusion-I

- Common attribute in all the architectural slides - extendibility.
- Good software - closed for change, open for extension.
- Each style has its good quality attributes at the cost of sacrificing other quality attributes.
  - Pipes and filters style has bad interactivity while event driven style has good support for user interactivity.
  - In event driven style its hard to share common data, while repositories has advantage of data sharing.

Conclusion-II

- Maximum benefit of software architectural styles can be achieved by the integration of different styles.



References

- Software Architecture - Zheng Qin, Jiankuan Xing, Xiang Zheng.
- Garfixia Software Architecture - Patrick Van Bergen.
- Art of Software Architecture: Design methods and Techniques - S.T. Albin.