SOFTWARE ARCHITECTURAL STYLE

Ridhima Joshi, Lakshmi Sudheera Dama

**Abstract** This paper discusses the Software Architectural Style, its advantages, its uses, its types and an example. The paper further discusses about the main difference between the Architectural Style, characteristics of an architectural style, Architectural Instance and Architectural Pattern.

***Index Terms*—Client - Server Architecture, Components, Connectors, Software Architecture**

# INTRODUCTION[[1]](#footnote-0)

Software goes through various development life cycle phases namely; Requirements, Designing, Implementation, Testing and Maintainancence. The software architecture plays an important role in each and every phase of SDLC. The software architecture uses some styles to design and implement the system. Any style can be used or any two or more than two styles can be used at a time. How an architectural style is characterized is explained in this paper with the help of types of architectural style.

# What is Software Architecture?

The Software architecture is not different from an real architecture of a building, the only difference is that the software architecture depicts the architecture of a software system. Software architecture is said as the skeleton or the backbone of the system. It depicts the system’s overall structure. Software architecture is defined in terms of components, connectors and configuration. The components are the collections of modules, clients, servers, databases, layers. The connectors helps to know the interactions which takes place like the procedure calls, events, multicast, pipes, client-server protocol messages. The configuration explains the topology of the system, that means the collection of components connected to form the overall structure.

Good software architecture is achieved by never losing the sight of what the user wants.It helps to adapt new requirements easily and reduce the cost of productivity. Software architecture forms the basis of communication for the development team. Software architecture can influence software qualities like performance, security, reliability and availability. In short, it can be said that software architecture is the blueprint of a software system.

# Architectural Style

According to David Garlan & Mary Shaw [January 1994, CMU-CS-94-166] software architectural style is defined as:

“A family of systems in terms of a pattern of structural organization. More specifically, an architectural style determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of constraints on how they can be combined. These can include topological constraints on architectural descriptions (e.g., no cycles). Other constraints - say, having to do with execution semantics - might also be part of the style definition.”

Architectural style can be said as a shape of an application. It is defined in the terms of set of element types like modules, components, connectors and ports that can be used and set of constraints. The set of constraints restrict how the type can be used, such as system’s runtime topology, modules dependencies, data direction across the connectors and visibility of components.

In other words, styles represent families of similar systems which are related by shared structural and semantic properties. An architectural style typically defines a vocabulary of design element types as a set of component, connector, port, role, binding and property of types together with rules for instances of the type.[4]

Architectural style has 4 properties: Vocabulary of design elements, Configuration rules, Semantic interpretation and Analyses.

## Vocabulary of design elements: Design elements consists of components and connectors types such as pipes, filters, clients, servers, parsers, databases.

## Configuration rules: Configuration rules are the design rules or constraints. Configuration rules consists of topological constraints that determine the permitted compositions of the design elements. For example specifying that a client-server organization must be an n-to-one relationship.

## Semantic interpretation: Composition of design elements that enables a designer to understand the overall properties of a system. It can be said that the designer can understand who the data will flow in the system.

## Analyses: Analyses that can be performed on systems built in what style. For example, deadlock detection for client-server message passing.

# Characterising an Architectural Style

A particular architectural style can be defined by answering the following questions:

## What is the design vocabulary?

Design vocabulary consists of the types of design elements: components and connectors used while designing the architecture of a particular software. The design elements play an important role in architectural styles as the architectural styles are dependent on the design elements.

## What are the allowable structural patterns?

Structural patterns in software engineering helps to identify a simple way to understand the relationships between the entities. The structural patterns help to understand the relationships between the components and connectors. This question answers in what way the design elements are connected and what format of an architectural style is formed.

## What is the underlying computational model?

Computational model mainly consists of the complex algorithms which are to be implemented during the software development. Computational model helps in understanding the behaviour of the complex software. Few examples of complex models can be weather forecasting, air traffic controlling model, traffic signal model. As the behaviour and working of the system is understood, it helps in designing the architecture for that particular system.

## What are the essential invariants of the style?

The architectural styles have some similar properties. The similar properties help to determine an architectural style. For example in Pipes and Filters architectural style, filters do not know the identity of the other up stream filters or down stream filters.

## What are some common examples of its use?

This question answers where the particular architectural style is commonly used. For example the client and server architectural style is mostly used in networking related systems.

## What are its advantages and disadvantages?

The benefits and the limitations of an architectural style can help one decide when to use it.

## What are some of the common specialization?

Architectural styles are specialized based on the properties of the styles such as Shared Memory, Distributed System, Messaging, Structure and Adaptable System.

# Types Of Architectural Styles

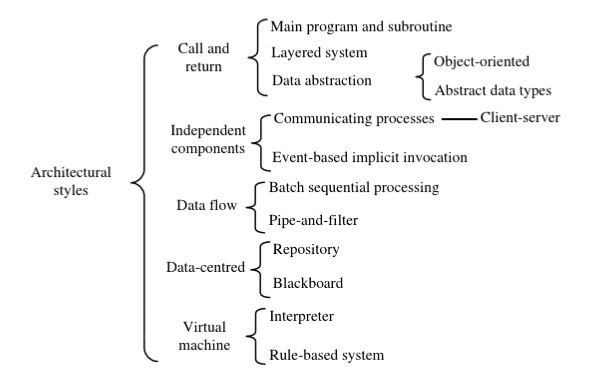


Figure 1: Architectural Styles

Architectural style types are Call and Return Architectural Style, Independent Components Architectural Style,

Data Flow Architectural Style, Data Control Architectural Style, Virtual Machine Architectural Style.

## 1. Call and Return Architectural Style

### What is the underlying computational model?

A software system which uses this style is decomposed into smaller components to deal with complexity and help achieve modifiability. A single thread has all the control. Each component executes only when it gets from another component and returns the control to that component when it terminates execution. Each component has a fixed entry where the executions of the components start, they are known as subroutines or procedures. The passes of the control from one component to another are called subroutine calls or invocations or procedure calls or function calls. Main program is the component that has the entrance to the whole program. Global variables or global state of the system is the shared data that all components can access.

### What is the design vocabulary?

A call and return architecture with a hierarchical structure is often called as main-program-and-subroutine with shared data. A group of subroutines that share a common data store is grouped together to form a module. Grouping subroutines into modules is granularity.

### What are the allowable structural pattern?

A common practice is to organise the connections between subroutines in certain patterns and to pack a number of interrelated subroutines into program units. This results in a number of subtypes of the style.

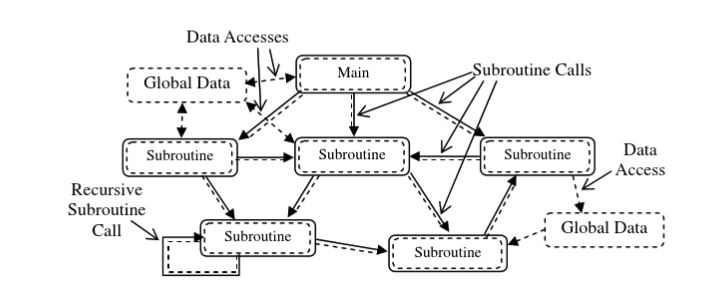


Figure 2: Call and return Architectural Style

### What are its advantages and disadvantages?

Advantages:

1. The decomposition of a complex program into subroutines enables programmers to separate the concerns in programming a large system into a number of less complex programming problems.
2. Encapsulation and modular programming enables developers to decompose large projects into manageable units and it easy’s the process of integrating them back into system.

### What are some of the common specialization?

Subtypes of Call and Return architectural style are Layered Systems Architectural Style and Data Abstraction Architectural Style.

### What are some common examples of its use?

CD Rental Shop Management is an day to day example of this style. [2]

## 1.1 Layered System Architectural Style

This style is commonly seen in system software and application systems.

### What is the underlying computational model?

The main point in this style is the way of organising subroutines into a simple topological structure.

### What is the design vocabulary?

Subroutines are organised into a number of groups called as layers.

### What are the allowable structural patterns?

The implementation of the subroutines of the layer depends on calls to the subroutines in the lower layers.

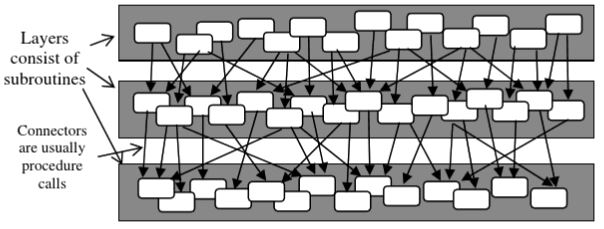


Figure 3: Layered System Architectural Style

### What are its advantages and disadvantages?

Advantages:

1. Support designs based on increasing levels of abstraction.
2. Changes in the function of one layer does not affect the other layers.
3. Helps to define standard layer interfaces upon which different implementers can build.

Disadvantages:

1. It is not easy to implement layered style to all systems.
2. It may be difficult to find the right levels of abstraction.

### What are some common examples of its use?

The common examples are communication protocols, application area like database systems and operating systems.[2]

## 1.2 Data Abstraction Architectural Style

### What is the underlying computational model?

In the object-oriented style, dynamic binding enables a variable to refer to objects of different classes. A call statement which may invoke different procedures at different times during one execution of the program is achieved through introduction of inheritance relation between program unit classes.

### What is the design vocabulary?

The data to be used and operated the same way as data of preliminary data types directly defined by programming languages is called as an abstract data type. The instances of the abstract data types are the components called as managers.

### What are the allowable structural patterns?

Classes are encapsulations of data and operations. A class in object oriented talks to other class by being generalisation or specialisation of another. Inheritance and polymorphism play an important role in this style.

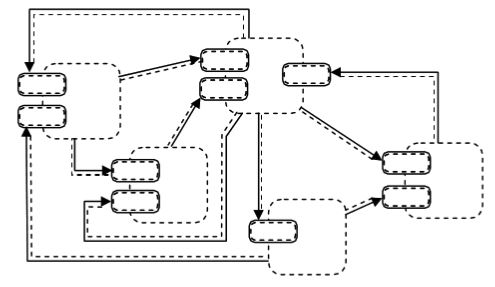


Figure 4: Data Abstraction Architectural Style

### 

### What are its advantages and disadvantages?

Advantages:

An object hides its representation from its clients, hence it is easy to change the implementation without affecting the clients

Disadvantages:

The identity of an object should be known in order to interact with the other object..

### What are some of the common specialization?

The subtypes of this architectural style are Object - Oriented Architectural Style and Abstract Data Types Architectural Style.[2]

## 2. Independent Components Architectural Style

### What is the underlying computational model?

Independent component architecture style consists of a number of components that communicate through messages. They send data to each other but do not contro each other. The message may be passed to named participants or passed among unnamed participants in case of using the publish paradigm.

### What is the design vocabulary?

All components must be executable. If the component is active it is called as processes or else just modules.

### What are the allowable structural patterns?

There is no general pattern for the whole class of independent component architectural style with regard to the topological structures of the systems.

### What are the essential invariants of the style?

The only means of information exchange among the components is message passing.

### What are its advantages and disadvantages?

Advantages:

1. This style achieves modifiability by decoupling various portions of the computation. It enables concurrent executions of the components, even parallel executions of components on various computers over a network.

2. It eases component integration.

Disadvantage:

A component cannot assume the receivers will respond to it, when it announces an event or send message to another component.

### What are the common specializations?

The subtypes of this architectural style are Event-based implicit invocation Systems, Interrupt Driven Process Systems and Multicast Message with Dynamic Binding Systems

### What are some common examples of its use?

Typical example is the client server architecture. It consists of a number of clients and one or more servers.[2]

## 2.1 Communication processes

These are the classic multi-process systems.

### What is the underlying computational model?

The components of a system are active processes. They communicate with each other through fixed channels of communications. They pass messages to each other to synchronise their executions and to achieve mutual exclusion in accessing shared resources.

### What is the design vocabulary?

The components are called processes and the communication with each other through communication channels and ports, where messages are sent and received.

### What are the allowable structural patterns?

The general structure is a network of processes connected by communication channels. In this style, there is no event handler that is responsible for redistributing the messages among the processes. The communicating processes systems have more complicated flow of information between processes rather than only having more or less simple patterns of data flow. The messages passed between processes not only contains data to be processed, but also contains requests of performing certain tasks.

### What are some of the common specializations?

Client-server architecture is a subtype of this style. Server serves data to more than one clients across the network. Client requests the server and server processes the request, synchronously or asynchronously.[2]

## 2.1 Event-based Implicit Invocation Architectural Style

### What is the underlying computational model?

Event systems make use of a message manager that manages communication among the component, invoking a component when a message arrives for it.

### What is the design vocabulary?

Publishing the data is known as a component’s announcement of data. Announcement is also known as an event. A message manager is called as event handler. Event handler records all components that register their interests in certain events. When an event takes place, the event handler distributes the data to those registered components.

### What are the allowable structural patterns?

The interface of a component contains a collection of events that it can rise and a collection of procedures or functions that other components can call directly. Hence, components may communicate with each other directly through procedure of function calls.

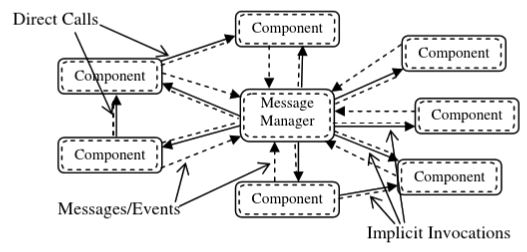


Figure 5: Event-based Implication Invocation Architectural Style

### What are its advantages and disadvantages?

Advantages:

1. This style strongly supports reuse and software evolution.
2. Component replacement is easy without affecting the interfaces of other components of the system.

### What are some common examples of its use?

Common examples can be found in database management systems to ensure consistency. in GUI systems to separate presentation of data, in programming environments to integrate various tools.

## 3. Data Flow

Data flow architectural style is mostly used in domains where data processing has an important role.

### What is the underlying computational model?

This style is viewed as a series of transformations on successive pieces of input data. Data enters the system and flows through one component at a time until it reaches the output or a data source.

### What is the design vocabulary?

Data flow architectural style consists of a collection of components called as processing elements or units. A processing element starts its computation when the inputs are available. After processing the input data it produces output. Then this output is fed as input to the other processing elements. The connections and interactions between these processing elements are known as flows or data flows.

### What are the allowable structural pattern?

Data flow architectural style can be viewed as a directed graph where processing elements are the nodes and data flows between the elements are the arcs.

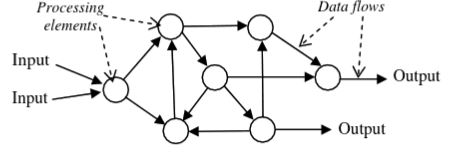


Figure 6: Data Flow Architectural Style

### 

### What are the essential invariants of the style?

1. The only interaction between processing elements is by the data flows.
2. Each processing element must be on the path from input to output.

### What are the advantages and disadvantages?

Advantages:

1. Components are highly independent.
2. The process elements can be executed concurrently even on different computers.
3. The process elements are capable of processing continuous streams of data and provide continuous controlling and monitoring functions of real time process control systems.
4. Components are reusable and can be easily integrated into other systems of the same architectural style.
5. At the component level, the processing elements can be modified without requiring modifications of other processing elements provided that formats and types of the data flows into and out of the element are compatible with the original ones.
6. At the structural level, new processing elements and data flows can be easily added into the system to extend the system’s functionality.

Disadvantages:

If synchronisations are necessary, the design and implementation of this architectural style will be difficult as the synchronisation of the computations of more than one processing element will be needed.

### What are some of the common specializations?

There are two main subtypes of data flow architectural style: Batch Sequential Processing Architectural Style and Pipe-and-Filter

### What are some common examples of its use?

The most common example for this type of architectural style can be a web based application. The client side that is the user’s computer is connected to the web server through the internet. The server processes the sequences of user’s requests in the form of messages which are passed as requests to a database server for processing database queries. The database queries are processed by the html page generator to produce an html file which is sent back to client side in the form of an HTML page to display on the computer screen of the user.[2]

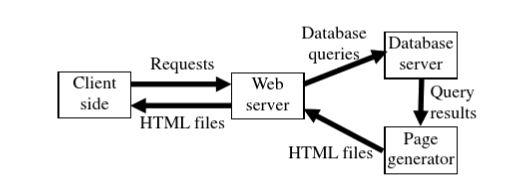


Figure 7: Web-based application

## 3.1 Batch Sequential Processing

### What is the underlying computational model?

Components are independent programs.Components are executed sequentially that is one component runs to completion before the next starts. The data is transmitted between components as a whole batch rather than a stream of data elements.

### What is the design vocabulary?

The computational components are called as processing steps or phases.

### What are the allowable structural patterns?

This style usually consists of a finite number of steps connected linerly, but some systems contain loops that repeat some processing steps for a finite number of times.

### What are its advantages and disadvantages?

Advantages:

Batch sequencing process architectural style avoids complicated issues related to the synchronisation between components by executing them one after another.

Disadvantages:

As the components do not execute concurrently. The performance of the system may be less satisfactory than other data flow systems.

### What are some common examples of its use?

Compilers are the most common batch sequential processing architectural styles.[2]

## 3.2 Pipe- and- Filter

### What is the underlying computational model?

Each component incrementally reads in streams of data on its inputs and incrementally produces streams of data on its outputs. The outputs from one component are directly fed to another component as its inputs. The connectors serve as conduits for the streams, transmitting outputs of one filter to another filter’s input.

### What is the design vocabulary?

Filters are the components and pipes are the connectors in the pipe and filter architecture style.

### What are the allowable structural patterns?

There is no structure restriction in addition to that of the restrictions of data flow architectural style.

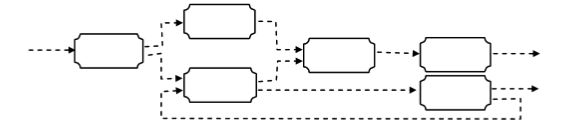


Figure 8 : Pipe-and-filter Architecture Style

### What are its advantages and disadvantages?

Advantages:

1. Pipe and filter architectural style allows the designer to understand the overall input output behaviour of a system as a simple composition of the behaviors of the individual filters.
2. Any two filters can be used together, provided they agree on the data that are being transmitted between them. Hence they are reusable.
3. New filters can be added to existing systems, and old filters can be replaced by improved ones easily.
4. As data is incrementally processed, the system is enabled to process long streams of input data.
5. The style supports concurrent execution, hence each filter can be implemented as a separate task and potentially executed in parallel with other filters.

Disadvantages:

1. The style leads to a batch organisation of processing. Although the filters can process data incrementally, they are inherently independent, so the designer must think of each filter as providing a complete transformation of input data to output data.
2. The style may be hampered by having to maintain correspondence between two separate but related streams.
3. Depending on the implementation, the style may force a lowest common denominator on data transmission, resulting in added work for each filter to parse and unparse its data. This can lead to loss of performance and to increased complexity in writing the filters themselves.

### What are some common specialisations?

Pipeline architectures are pipes and filters systems in which there is no cycle of data flows.

### What are some common examples of its use?

Mostly the examples are written in Unix shell. Unix provides a runtime mechanisms for implementing pipes.

Eg : ls -l | more

Combines ls and more through the usage of pipe.[2]

*4. Data Centered*

### What is the underlying computational model?

The systems in which the access and update of a data store is an apt description.

### What is the design vocabulary?

Components in this style is a data store, in which the data is stored and can be accessed by other components known as shared data. The components which access the shared data for the functionality and to communicate with each other are called as clients.

### What are the allowable structural patterns?

The shared data is the central means of communication among the clients.

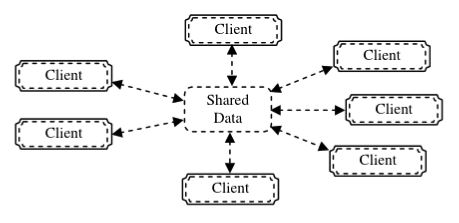


Figure 9 : Data Centered Architecture Style

### What are its advantages and disadvantages?

Advantages:

1. The clients are relatively independent of each other.
2. The data store is independent of the clients.
3. New clients can be easily added.

Disadvantages:

Modifying the representation of data stored in the shared data can be problematic as it may affect all clients in the system.

### What are some common specialisations?

The subtypes of this style are Repository and Blackboard.

*5. Virtual Machine*

### What is the underlying computational model?

Virtual machines are software systems that simulate a computer system, with some functionality which is not native to the hardware and/or software on which it is implemented.

### What is the design vocabulary?

There are three passive data components namely program, data and internal state and one active component: the interpretation engine. Program component stores the ‘program’ being interpreted which is an abstract representation of the process of data. The data component stores the information to be processed. The internal state stores the current state of the interpretation engine. The interpretation engine selects an instruction from the program being interpreted, updates the program’s data according to the instruction and updates its internal state.

### What are the allowable structural patterns?

A software system using this style usually consists of four components interconnected.

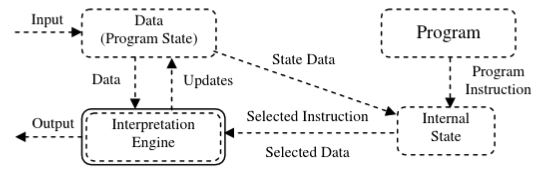


Figure 10 : Virtual Machine Architectural Style

### What are its advantages and disadvantages?

Advantages:

1. It is portable.
2. It adds flexibility through the ability to interrupt and query the program via executing a program using interpreter.

Disadvantages:

Modifications can be introduced at run-time but there is a performance cost.

### What are some of the common specialization?

The subtypes of Virtual Machine Architectural style are Interpreter Architectural Style and Rule-Based System Architectural Style.

### What are some common examples of its use?

Common examples are interpreters, syntactic shells, rule-based systems and command language processors. The most common example is Java Virtual Machine, allows Java to be platform independent.[2]

# Views of Architectural Styles

The architectural style is can be understood in three ways:

1. Architectural Style as Language
2. Architectural Style as a System of Types
3. Architectural Style as a Theory

The above views explains architectural style in terms of the software architecture properties: design vocabulary, configuration rules, semantic interpretation and analysis.[10]

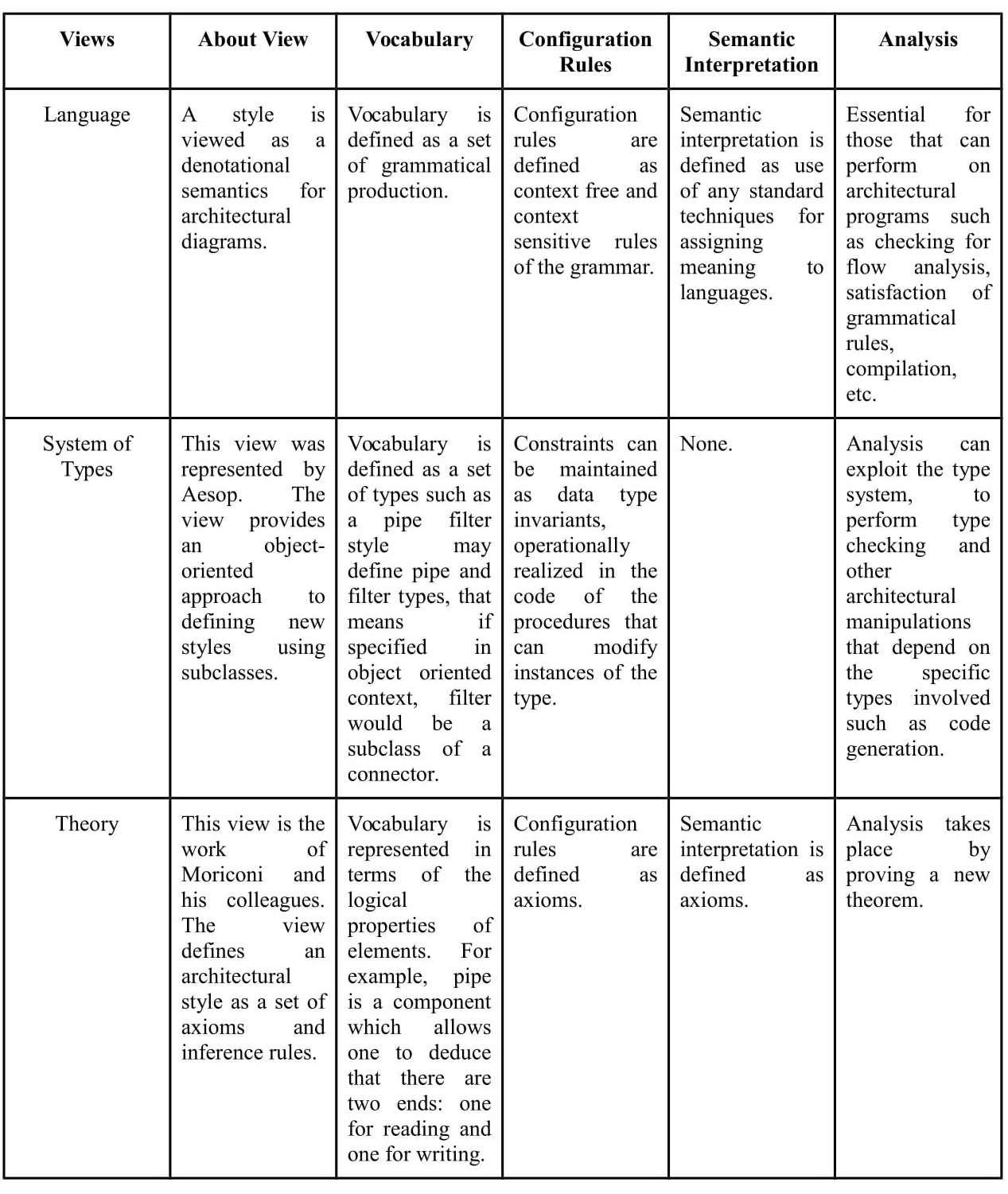


Figure 11 : Views of Architectural Styles

# Architectural Style Versus Pattern

Architectural styles generally provide guidance and analysis for building a broad class of architectures in a specific domain. Architectural styles defines the components and connectors. It answers to question ‘what’ and is less domain specific.

Patterns focus on solving smaller and more specific problems within a given or multiple styles. Patterns need not be architectural, they deal with solutions to lower level programming mechanisms rather than system structuring issues. Patterns define the implementation strategies of those components and connectors. It answers to question ‘how’ and it is more domain specific.[1][3]

*Questions to determine a style or a pattern:*

Which architectural pattern is best for a given situation depends on which requirements has the highest priority,

### 1. Maintainability:

How easy or difficult is it to add an additional processing component, for instance to filter certain words? How easy or difficult is it to change the input format, such as adding line numbers? In the Pipe-filter pattern for instance, adding a filter is very easy, but changing the input format might be hard.

### *2. Reusability*:

Can individual components be reused in other systems? In this case, the Pipe-and-filter pattern enhances reusability because of the uniform data format that is used.

### *3. Performance:*

Is the response time small enough? Is overall resource behavior (for example memory usage) acceptable? Patterns that make use of parallelism, such as the Pipe-filer pattern and the Eventbus pattern, will have better performance. On the other hand, starting a complex system like the event bus system, or transforming data in every filter using a different data structure, may lower performance.

### *4. Explicitness*:

Is it possible to provide feedback to the user? Per stage? This is not possible in the Pipe-filter pattern for instance.

### 

For the KWIC example, there is no difference between the different solutions, but had a Master-slave pattern been applied, fault-tolerance would have been enhanced.

The list of requirements and their priorities will vary for every system. No rigid guidelines can be given to tell you which pattern will be the best in every case. Much also depends on the implementation of the pattern. Independent processes, for example, may be implemented using threads or using processes on different machines. The balance between communication and computation, the capacity of the processors involved and the speed of communication between machines, among others, will decide which implementation will have the best performance. [11]

# Architectural Style Versus Architectural Instance

An architectural style, however, narrow the constraints on the form and structure of a family of architectural instances. For example, pipe filter" architectural style might dene the family of system architectures that are constructed as a graph of incremental stream transformers. Architectural styles prescribe such things as a vocabulary of components and connectors, topological constraints, and semantic constraints. Styles range from abstract architectural patterns and idioms such as client-server, to concrete reference architectures such as the ISO OSI communication model.

An architectural instance refers to the architecture of a specific system. Box and line diagrams that accompany system documentation describe architectural instances, since they apply to individual systems.

# Uses of Architectural Style

There are different types of architectural styles explained in further sections. An architecture of a system is not only made up of a single architectural style, but it can be a combination of different architectural styles. Below are some examples of the uses of an architectural style.

1. Desktop applications may use client-server architecture style and component-based architectural style. The client-server architectural style will be required if a client wants to send a request to a server. The component based architectural style can be used to decompose the design of the application further into independent components that opens the appropriate communication interfaces.
2. Web application can use layered architectural style, so that the presentation logic layer, business logic layer and the data logic layer will be separated from each other.

# Advantages of Architectural Style

1. Architectural style promotes design reuse and can lead to code reuse .

The designs can be used in similar kind of designs for systems. Solutions with well understood properties can be reapplied to new problems. Architectural styles promotes design reuse, hence it further helps in reuse of code for similar kind of systems or modules. Often the invariant aspects of an architectural style lend themselves to an implementation that can be shared or reused.

1. Architectural style can provide the basis for system understanding and style specific visualization.

Architectural styles leads to designing of software architectures and diagrammatic representation of softwares not only help the developers to understand, characterize and implement the new system but also to non technical people involved in the project like stakeholders.

1. Constraining the design space often permits specialized style specific analyse such as deadlocks, latency, throughput, etc and choices.
2. Use of standardized architectural styles supports interoperability.
3. Architectural style allows a designer to exploit recurring patterns of organization such as topological configuration or specific organization of components such as MVC in object oriented systems.
4. Architectural style provides a context within which certain kinds of design integrity can be enforced.
5. Architectural style forms a basis for supporting reuse of architectural building blocks and patterns.

The benefits of the architectural styles vary as per the architectural style and as per the needs of the system. [8]

# Example

The most common example to explain an architectural style is the WWW browser. It uses the Client- Server Architectural style, which is a subtype of Communicating Process Architectural Style.

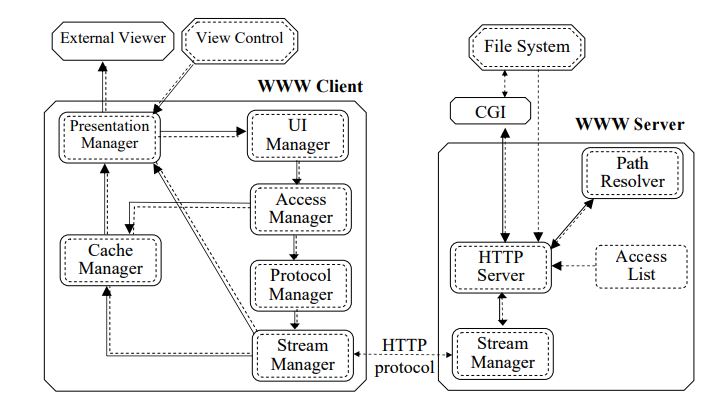


Figure 12: WWW Client - Server Structure

There are six components in this example: external viewer, client side (WWW browser), web server, view control, CGI, and a file system.

The external viewer is a program used to view types of contents of a web page like postscript viewer. They are active runtime processes.

The view control component maintains a configuration file and aids the mapping of document types to external viewers.

The server side contains the file system which maintains a collection of files in the format of HTML and other data. The files are retrieved by the WWW server when requested by the client.

CGI executes the scripts embedded in HTML files and provides extended functionality of the system for implementation of web based applications.

The client side that is the WWW browser can be decomposed further into UI, presentation manager, access manager, cache manager, protocol manager and stream manager.

UI manager handles the look and feel of client’s user interface.

Presentation manager delegates information display to UI manager or external viewers to view resources that are not directly supported by the UI manager.

Access manager accepts the information requests in the form of URL’s captured by the UI manager and determines if the request URL exists in the cache.

Cache manager is a collection of retrieved files and passes a file to the presentation manager if it is requested by the access manager.

Protocol manager determines the types of request for information captured by access manager and invokes an appropriate protocol to service the request.

Stream manager uses the protocol invoked by protocol manager to communicate with the server in order to obtain the requested information.

The server side is also decomposed into components: Stream manager, Access list, path resolver, HTTP server.

Stream manager communicates with the stream manager on the client side to receive information request and send back the requested information.

Access list stores a list of clients that are authorized for the documents.

Path resolver resolves the location of the requested documents in the files.

HTTP server ensures the transparent access to the file system where the source documents are stored.[9]

# Conclusion

The paper explains briefly about software architecture, architectural style by defining the four properties of each style and how to characterize the architecture styles. It also explains in detail about different architectural styles that are commonly used with their benefits and limitations. Paper highlights comparisons between architectural style and architectural pattern and architectural.

Authors have also provided an example that briefly explains how an architectural style is determined..

1. Appendix

CGI: Common Gateway Interface

UI: User Interface

WWW: World Wide Web

1. Acknowledgment

We would like to thank Dr. Sharon White for the guidance, support and help while writing the paper.

1. References
2. “Structural pattern,” *Wikipedia*, . Available: https://en.wikipedia.org/wiki/Structural\_pattern.
3. Zhu, Hong. Software Design Methodology : From Principles to Architectural Styles, Elsevier Science & Technology, 2005. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/uhcl/detail.action?docID=269543
4. R. T. Monroe, A. Kompanek, R. Melton and D. Garlan, "Architectural styles, design patterns, and objects," in IEEE Software, vol. 14, no. 1, pp. 43-52, Jan/Feb 1997. doi: 10.1109/52.566427
5. *Chapter 3: Architectural Patterns and Styles*. Available: https://msdn.microsoft.com/en-us/library/ee658117.aspx.
6. “Content Strategy Starts with a Message Architecture,” OpenView Labs, 05-Feb-2015. Available: https://labs.openviewpartners.com/building-your-content-strategy-with-a-message-architecture/#.WsmelYjwbIU.
7. “Architectural Styles” Available: http://shapingsoftware.com/2009/02/09/architectural-styles/
8. “Software Architecture and Architectural Styles” Available: https://pdfs.semanticscholar.org/presentation/bbeb/36e1a01c3991dfef72dbbeaf7ccbdbf95fe6.pdf
9. David Garlan, “What is Style?” Carnegie Mellon University
10. “Architectural Patterns” Available: https://www.ou.nl/documents/40554/349790/IM0203\_03.pdf

1. [↑](#footnote-ref-0)