Review on Application of Pipe and Filter Architectural Style

Neha Rani

MBA-IT, SICSR(2015-2017)
Symbiosis International University

Abstract—Pipe and Filter Architecture Style is a very simple, yet powerful architecture that is also very robust. It is a subtype of Dataflow Architectural Style which is very prominently used architectural style in the industries for software development. This Architectural style has any number of components as filters which transform or filter data, before passing it on using connectors known as pipes to other components of the software architecture. The filters are all working in parallel. The architecture is simply sequenced and sometimes used for very complex structures. In this Paper Applications of Pipe and Filter Architectural style in organization in discussed with its methodologies and input-process-output. The paper would give an insight that how the Pipe and Filter Architectural style in being used in the industries, for what kind of Products, How Frequently, up to which complexity level and significance of its use in Software Application Development.

Keywords: Pipe, Filter, compiler, design, architecture, style, framework

I. INTRODUCTION

Software Architecture is the process of defining high-level structures of a software system, the specialization of creating such structure and documenting them for further development of the software. These structures are developed in accordance to the software system that has to be developed, requirements. The software designing evolves around the functionality that has to be developed into the software and so does the software architecture is defined for the same for software modeling.

Types of Software architecture

❖ Dataflow Architecture
Dataflow Architecture is a method for determining parallel calculation at a fine-grain level, typically as two-dimensional charts in which directions that are accessible for simultaneous execution are composed nearby each other while those that must be executed in grouping are composed one under the other.

❖ Batch sequential Architecture
Batch sequential architecture takes into consideration the on-interest provisioning of a multi-part work handling engineering that can be utilized for prompt or postponed organization of a heterogeneous, versatile "network" of specialist hubs that can rapidly mash through vast group preparing errands in parallel.

❖ Pipe and Filter Architecture
Pipe-And-Filter. An extremely basic, yet capable engineering, that is likewise exceptionally powerful. It comprises of any number of components (filters) that change or channel information, before passing it on by means of connectors (pipes) to different segments.

❖ Distributed Architecture
Distributed System Architecture permits different Process Knowledge Systems (PKS) to work as one over various units, destinations and offices over the venture. This single coordinated framework empowers consistent worldwide access to information, cautions, alarms, intelligent administrator control messages and history crosswise over gatherings of frameworks.
• **Client Server Architecture**
  The client/server design style portrays dispersed frameworks that include a different customer and server framework, and an interfacing system. The least complex type of customer/server framework includes a server application that is gotten to straightforwardly by various customers, alluded to as a 2-Tier design style.

• **Broker Architecture (CORBA)**
  Broker Architectural Style is a middleware design utilized as a part of circulated processing to organize and empower the correspondence between enrolled servers and customers. Here, item correspondence happens through a middleware framework called an article demand dealer (software bus).

• **Service Oriented Architecture**
  Service-oriented architecture (SOA) empowers application usefulness to be given as an arrangement of services, and the production of uses that make utilization of programming administrations. Administrations are approximately coupled on the grounds that they utilize principles based interfaces that can be summoned, distributed, and found. Administrations in SOA are centered around giving an outline and message-based communication with an application through interfaces that are application checked, and not component or object-based.

  ❖ **Interaction Oriented Architecture**
  Interaction-oriented architecture isolates the association of client from information reflection and business information preparing. The Interaction-oriented architecture disintegrates the framework into three noteworthy components − Data module, Control module, View presentation module.

• **Model View Controller Architecture (MVC)**
  MVC breaks down a given programming application into three interconnected parts that assistance in isolating the inside representations of data from the data displayed to or acknowledged from the client.

  ❖ **Data-centered Architecture**
  In data-centered architecture, the information is incorporated and got to as often as possible by different parts, which adjust information. The fundamental motivation behind this style is to accomplish integrality of information. Data-centered architecture comprises of various segments that convey through shared information archives. The parts get to a common information structure and are moderately free, in that, they associate just through the data store.

  • **Repository Architecture**
  In Repository Architecture Style, the data store is inactive and the customers (software components or modules) of the information store are dynamic, which control the rationale stream. The taking an interest segments check the data store for changes.

  • **Blackboard Architecture**
  In Blackboard Architecture Style, the information store is dynamic and its customers are aloof. In this manner the intelligent stream is controlled by the present information status in information store. It has a chalkboard segment, going about as a focal information storehouse, and an inside representation is manufactured and followed up on by various computational components.

  ❖ **Hierachical Architecture**
  Hierarchal architecture sees the entire framework as a chain of command structure, in which the product framework is deteriorated into sensible modules or subsystems at various levels in the order. This methodology is ordinarily utilized as a part of outlining framework programming, for example, system conventions and working frameworks.
• **Call and Return Architecture**
  The aim of this style is to reuse the modules and openly create singular modules or subroutine. In this style, a product framework is isolated into subroutines by utilizing top-down refinement as indicated by coveted usefulness of the framework.

• **Layered Architecture**
  Layered architecture concentrates on the gathering of related usefulness inside an application into particular layers that are stacked vertically on top of each other. Usefulness inside every layer is connected by a typical part or duty. Correspondence between layers is express and inexactely coupled. Layering your application suitably underpins a solid partition of concerns that, in turn, bolsters flexibility and maintainability.

• **Virtual Machine Architecture**
  Virtual Machine architecture imagines some usefulness, which is not local to the equipment and/or programming on which it is executed. A virtual machine is based upon a current framework and gives a virtual deliberation, an arrangement of characteristics, and operations.

❖ **Component-Based Architecture- Component level design**
  Component-based architecture portrays a product building way to deal with framework outline and advancement. It concentrates on the deterioration of the outline into individual utilitarian or consistent parts that uncover very much characterized correspondence interfaces containing techniques, occasions, and properties. This gives a more elevated amount of deliberation than article situated configuration standards, and does not concentrate on issues, for example, correspondence conventions and shared state.

❖ **Implicit-Asynchronous Communication Architecture**
  The thought behind implicit summon is that as opposed to conjuring a methodology straightforwardly, a segment can report (or communicate) one or more occasions. Different parts in the framework can enlist an enthusiasm for an occasion by partner a method with the occasion. At the point when the occasion is reported the system itself summons the greater part of the methods that have been enrolled for the occasion. In this manner an occasion declaration "implicitly" causes the summon of methodology in different modules.

• **Message-based Architecture**
• **Event Based Architecture**

### II. PIPE AND FILTER ARCHITECTURE

Pipe and Filter Architecture provides the structure for the such systems which involves processing of stream of data, for example: OSI model, TCP/IP networks, ISO, Compiler Design phase. In each processing step the data passes through a filter component and adjoining the pipe through each processing filter units are joined to make the activity complete. The filter is used to transform or filter data. The pipes are used as connector to connect to other components and other filters. All filters get processed at the same time and all filter works in the processor as the form of threads. All threads are processed consecutively and all filters are getting processed virtually at the same time. Following is the diagrammatic representation for pipe and filter software architecture style.
III. NEED AND IMPORTANCE OF PIPE AND FILTER SOFTWARE ARCHITECTURE

Pipe and Filter software architecture a sequential flow architecture model for the processing of data fetched in such software application (software application modeled in pipe ad filter architectural style). It is very flexible in approach. The ideals of the actuality are that the information component was prepared effectively in the past rounds, the likelihood of disappointment abatements for each resulting round of preparing. Along these lines, the likelihood preparing the component effectively by a few channels is an expanding capacity of the quantity of information preparing emphasis. Which increases the functionality, reliability, usability and efficiency of the pipe and filter software architecture. The pipe-and-channel style disentangles framework support and improves reuse for the same reason-channels remain solitary, and we can regard them as secret elements. Both pipes and filters can be progressively created: Any blend of channels, associated by funnels, can be bundled and appear to the outside world as a channel. Because a filter can prepare its contribution to disengagement from whatever remains of the framework, a pipe and-filter framework is effortlessly made parallel or conveyed, giving chances to improving a framework's execution without adjusting it. This enhances the maintainability and portability of this architecture.

This design is extraordinary on the off chance that you have a great deal of changes to perform and you should be extremely adaptable in utilizing them, yet you need them to be robust.

IV. EXAMPLES OF PIPE AND FILTER ARCHITECTURE

1. A example of Interpreter by Ashish kumar, Thapar University, India follows as:

   An interpreter basically used to produce a virtual machine. It is suitable for applications in which most appropriate machine is not directly available. Java language uses interpreter. When a java program is compiled with the help of JVM (Java virtual machine) JVM create a byte code which is an intermediate code not binary code. Now this byte code can be run on any machine. It has four states- one for engine and rest three are memories. These states are:
   1) Interpreter engine for executing the program.
   2) Current state of interpreter engine.
   3) Program being interpreted.
   4) Current state of program being interpreter. [1]

2. Computational Model

   Robert Allen and David Garlan, Carnegie Mellon University, states Computational model is not too restrictive, is it too general? An obvious restriction is to model filters as finite state machines. However, this leads to some rather unfortunate consequences. An important property of a pipe and filter architecture is that it is hierarchical: a pipe-filter system can be viewed, recursively, as a filter in another system. If the computational power of a filter is limited to a finite state machine, however, this property no longer holds. Since a pipe can hold an arbitrary amount of data, a system can assume an infinite number of states; there is no way to map an arbitrary system to a finite state filter. [2]

3. Compiler Design

   A compiler transforms a program written in an high-level programming language from source code into item code. Software engineers compose programs in a structure called source code. Source code must experience a few stages before it turns into an executable system. The initial step is to pass the source code through a compiler, which interprets the high-level language instructions into object code.
Marchese states that-The Application of Pipe and Filter in Compiler design is stream data processing and transformation

The Approach used: pipeline of generators, transformers, filters
The Advantages of using Pipe and Filter architecture for compile design:
  – Independence of the processing elements
  – Power of composing independent elements
  – Reusability/replaceability of elements. [3]

Compiler Pipeline:

V. CONCLUSION

Pipe and Filter yet very simple, basic architecture fulfil the needs for design complex systems and the systems that need lot of interaction with the CPU of the computer. Its components and filters align the interaction which helps the system to run efficiently and speedily. It divides the multifaceted structure into simple ones to give an simplicity for designing large and major, highly complex and intricate systems.

REFERENCES
