Pipe and filters pattern
(Patrón de tuberías y filtros)

Contexto:

Este patrón o tipo de arquitectura se centra en que, a partir de datos de entrada, después de pasar por medio de procesos o funcionalidades individuales y encapsuladas, llamadas filtros, dichos datos se transporten a otro filtros o puertos de salida a través de tuberías de forma secuencial.

¿Cuándo se utiliza ?

Este patrón se utiliza cuando queremos implementar en nuestro sistema, un flujo secuencial,sincronizado y automatizado de ciertos pasos.
A partir de esto podemos generar un alto grado de mantenibilidad y reutilización, ya que los filtros al ser independientes de otros filtros adyacentes en su proceso, se pueden sustituir o mejorar fácilmente sin afectar absolutamente nada dentro del sistema.
También es útil cuando se tiene que trabajar con una gran cantidad de datos.

¿Cuándo no se utiliza?

Cuando no se quiere la implementación de un sistema simplista o se necesite hacer un programa de mayor inteligencia, ya que este patrón no tiene una forma clara de tomar decisiones, como por ejemplo; a qué filtro es mejor irse, o cuál lo hará más rápido, etc.
Además de que hay una gran pérdida de eficiencia cuando se presentan ciclos, ya que no se tienen en cuenta los procesos “rio arriba”.
Y es totalmente inútil cuando se desea implementar en un sistema que necesite la interacción con el usuario.

Ventajas:

- Mantenibilidad
- Reutilización
- Se puede trabajar un gran volumen de datos
- Automatización de procesos
- Bajo acoplamiento
- Paralelismo
- Fácil implementación
- Incremental

Desventajas:

- No permite interacción con el usuario.
- No se puede implementar en programas muy largos.
- Es muy básico, y no se pueden implementar procesos inteligentes o complejos.
- Puede haber duplicación de funciones.
- Ineficiente cuando hay ciclos.

¿En qué se puede utilizar?

- Es muy utilizado en sistemas operativos *Unix,* para concatenar comandos.
- Para la construcción de arquitecturas de compiladores.
- En la arquitectura de computadores.
- Para trabajar con *Big Data*.
- Transformación de datos.

Bibliografía

https://jegadeesansite.files.wordpress.com/2018/01/sei-series-in-software-engineering-len-bass-paul-clements-rick-kazman-software-architecture-in-practice-addison-wesley-professional-2012.pdf

https://codesitio.com/recursos-utiles-para-tu-web-o-blog/cursos/curso-de-java-flujos-de-datos-en-java/ (Se tomó como base para realizar el ejemplo).