UNIVERSIDAD POLITECNICA DE CHIAPAS

ING. EN DESARROLLO DE SOFTWARE

CUATRIMESTRE:

5

GRUPO:

B

MATERIA:

Arquitectura de Software

PROFESOR:

Alonzo Macias

Actividad:

Investigación - Drivers - Estilos - Patrones

ALUMNO:

Mauricio Alejandro Ocampo Lopez

Fecha de entrega:

27/01/2022

DRIVERS ARQUITECTONICOS

¿Para qué son?

para enfocarse únicamente en los requerimientos de mayor influencia respecto de la forma que tomarán los elementos que componen las estructuras arquitectónicas. En el contexto de la creación de software, a este subconjunto de requerimientos se le conoce como drivers arquitectónicos. De esta forma, en el contexto del proceso de desarrollo de la arquitectura.

¿Para qué sirven?

puede verse como una ventaja pues es posible comenzar a realizar el diseño de la arquitectura antes de haber terminado de documentar todos los requerimientos. Ciertas metodologías de desarrollo como por ejemplo el Rational Unified Process recomiendan, de hecho, que se siga este enfoque. Retomando el ejemplo anterior, un caso de uso primario podría ser el realizar consultas del catálogo de productos. El criterio para elegir este caso de uso como primario es su importancia relativa a la satisfacción del objetivo de negocio y el hecho de que la consulta de catálogos involucra realizar conexiones hacia los sistemas de los fabricantes. Una restricción para dicho sistema podría ser que se usen librerías y herramientas open source.

Ejemplos

Los drivers arquitectónicos se clasifican en tres clases:

1. Drivers funcionales.
2. Drivers de atributos de calidad.
3. Drivers de restricciones.

Drivers funcionales: Esta clase de drivers, y particularmente los requerimientos de usuario primarios, son importantes porque proveen información relevante para llevar a cabo la descomposición funcional del sistema y asignar estas funcionalidades a elementos específicos en la arquitectura.

- Su relevancia en la satisfacción de los objetivos del negocio del sistema.
- La complejidad técnica que representa su implementación.
- El hecho de que representan algún escenario relevante para la arquitectura.

Drivers de atributos de calidad: son drivers de la arquitectura, independientemente de su prioridad. Pero su tiempo para realizar un diseño es demasiado acotado para producir un diseño inicial.

- Su relevancia en la satisfacción de los objetivos del negocio del sistema, es decir, su importancia para el cliente.
- La complejidad técnica que representa su implementación, esta es su importancia para el arquitecto

Drivers de restricciones: los requerimientos de este tipo no tienen prioridad, los tipos de requerimientos tienen una razón de ser, no a todos se les da la misma relevancia para el diseño de la arquitectura, aun si cuentan con una prioridad alta para el negocio. Por ello y por el hecho de el tiempo para realizar el diseño está por lo habitual acotado

ESTILOS DE ARQUITECTURA

¿Para qué son?

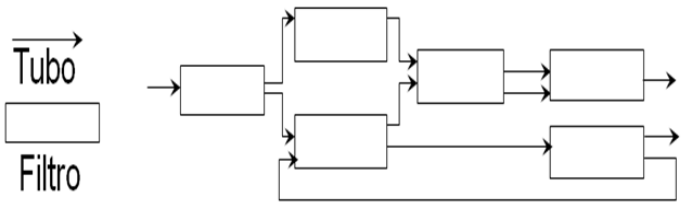
es un conjunto de decisiones de diseño arquitectural que son aplicables en un contexto de desarrollo específico, restringen las decisiones de diseño de un sistema a ese contexto y plantean como objetivo ciertas cualidades para el sistema resultante, Es un aspecto muy importante en la arquitectura del software, son patrones tanto en diseño como en arquitecturas.

¿para qué sirven?

Sirven para establecer un vocabulario común, donde se dan nombres a los componentes y conectores, así como a los elementos de datos, Establecen un conjunto de reglas de configuración, como la topología del sistema Definen una semántica para las reglas de composición de los elementos Posibilitan el análisis de los sistemas construidos sobre el estilo. Favorecen al tratamiento estructural, la investigación, la teoría. También para reutilizar en situaciones semejantes en un futuro

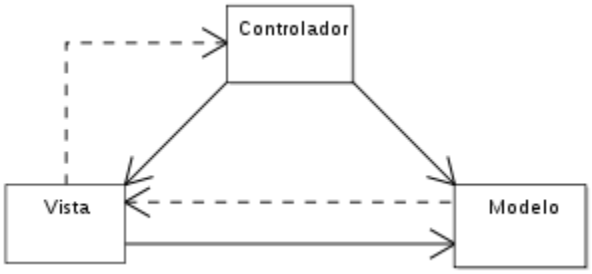
Ejemplos

Tuberías y filtros:



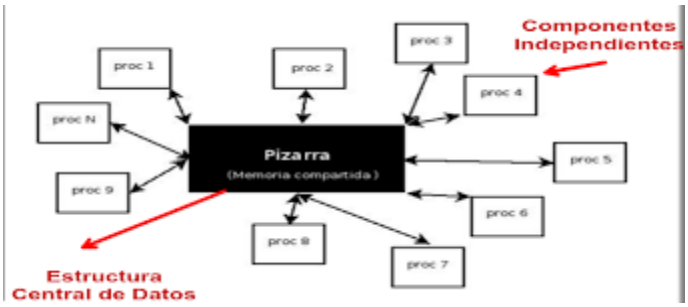
sirve para tomarse totalmente independiente de otros, este los datos no se alteran durante su transmisión, los tubos crean un contexto en el que operan los filtros, pero estos no operan de manera independiente de ellos.

Modelo-Vista-Controlador:



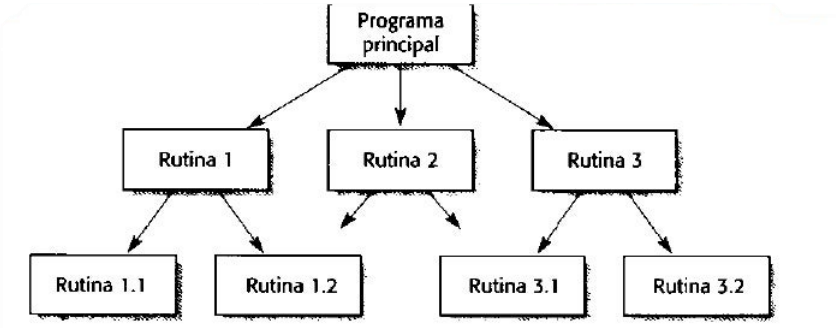
El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado. En la parte de la vista, como bien lo dice el nombre maneja la visualización de la información. En el controlador, este interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

Pizarra:



Sirve para una administración centralizada, de una forma adecuada, manejando grandes volúmenes de datos.

Estilo de llamada-retorno



Es usualmente sincrónicos, tiene parámetros de llamado, argumentos de respuestas. Sus variantes son programa principal, arquitecturas RPC, basadas en objetos, basadas en mensaje/recursos, y en servicios.

Patrones de diseño

¿Para qué son?

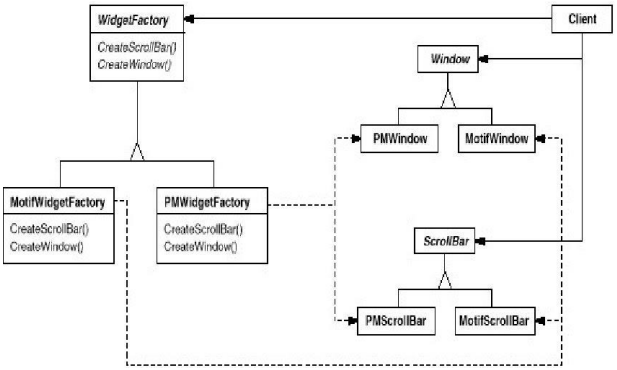
Es conjunto de decisiones de diseño que se pueden aplicar a un problema de diseño recurrente y que pueden parametrizarse para diferentes contextos donde ese problema de diseño aparece.

¿Para qué sirve?

Nos sirve para soluciones conceptuales a problemas recurrentes a la hora de diseñar, Un aspecto importante a resaltar es que los patrones son soluciones conceptuales. Esto significa que no pueden usarse de forma directa, sino que deben ser “instanciados”, es decir, adecuados al contexto y al problema específico que se busca resolver

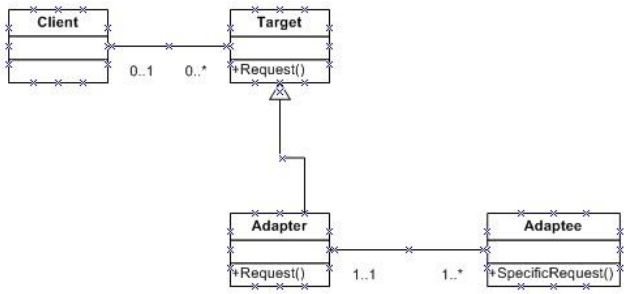
Ejemplo

Patrones creacionales:



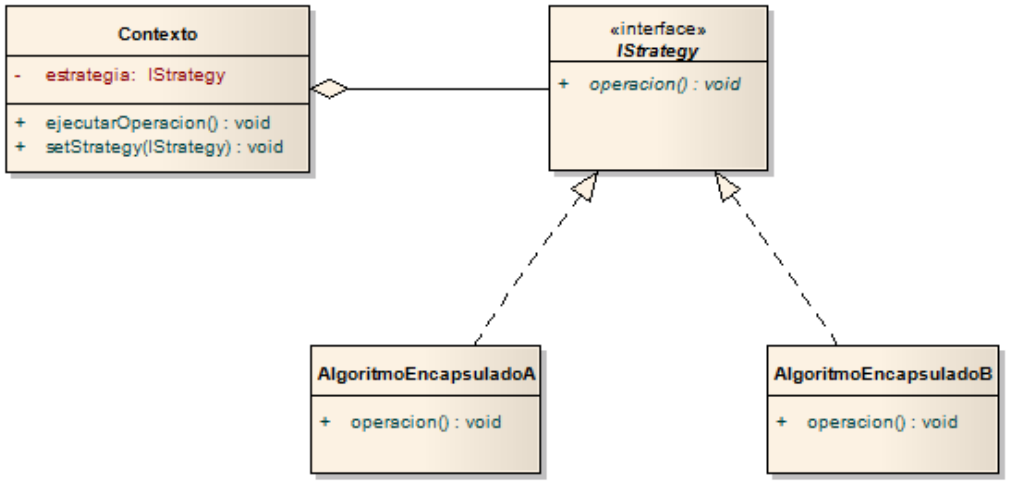
Los patrones de creación proporcionan diversos mecanismos de creación de objetos, que aumentan la flexibilidad y la reutilización del código existente de una manera adecuada a la situación. Esto le da al programa más flexibilidad para decidir qué objetos deben crearse para un caso de uso dado.

Patrones estructurales:



Estos facilitan soluciones y estándares eficientes con respecto a las composiciones de clase y las estructuras de objetos. El concepto de herencia se utiliza para componer interfaces y definir formas de componer objetos para obtener nuevas funcionalidades.

Patrones de comportamiento:



El patrón de comportamiento se ocupa de la comunicación entre objetos de clase. Se utilizan para detectar la presencia de patrones de comunicación ya presentes y pueden manipular estos patrones**.**

Conclusión

Con la información antes dada podemos ver, primero que los drivers son como un apoyo, ya que, con estos requerimientos, seguimos una arquitectura, con esto teniendo en cuenta tendremos una estructuración correcta del sistema permitirá satisfacer la mayoría de estos drivers. En estilos y diseños puedo decir que es casi lo mismo con sus claras diferencias siendo que los estilos establecen un conjunto de reglas se basan investigación, la teoría en lo que los patrones deben ser adecuados al problema para llegar un resultado en específico.

Bibliografía

https://www.researchgate.net/profile/Perla-Velasco-Elizondo/publication/281137715_Arquitectura_de_Software_Conceptos_y_Ciclo_de_Desarrollo/links/57144e1408aeebe07c0641ab/Arquitectura-de-Software-Conceptos-y-Ciclo-de-Desarrollo.pdf

http://blog.osoft.es/index.php/2016/03/24/estilos-y-patrones-de-arquitecturas-de-software/

https://slideplayer.es/slide/25533/

https://profile.es/blog/patrones-de-diseno-de-software/