

**Universidad Tecnológica de Aguascalientes**

**Carrera:**

**Ingeniería en Desarrollo y Gestión de Software**

**Materia:**

**Arquitectura de Software**

**Investigación de Frameworks de Arquitectura**

**4+1 Kruchten**

**Profesor:**

**Araujo Gonzalez Miguel Antonio**

**Alumnos:**

## **Nombre de Los Alumnos:**

## **Calzada Rodriguez Brayan Alexis**

## **Rios Valencia Onyx Ricardo**

## **Gonzalez Limon Guadalupe del Rocio**

## **Gomez Dominguez Pedro Alejandro**

## **Garcia Resendiz Oscar Renato**

**28 de Enero de 2022**

**Introducción**

El modelo 4+1 de Kruchten, es un modelo de vistas diseñado por el profesor Philippe Krutchten y que encaja con el estándar IEEE 1471-2000 (Recommended Practice for Architecture Description of Software-Intensice System) que se utiliza para describir la arquitectura de un sistema de software intensivo, basado en el uso de múltiples puntos de vista.

Las vistas suelen describir el sistema desde el punto de vista de diferentes interesados, tales como usuarios finales, desarrolladores o directores de proyecto.

La vista lógica describe el modelo de objetos del diseño cuando se usa un método de diseño orientado a objetos. Para diseñar una aplicación muy orientada a los datos, se puede usar un enfoque alternativo para desarrollar algún otro tipo de vista lógica, tal como diagramas de entidad-relación.

La vista de procesos describe los aspectos de concurrencia y sincronización del diseño.

La vista física describe el mapeo del software en el hardware y refleja los aspectos de distribución.

La vista de desarrollo describe la organización estática del software en su ambiente de desarrollo.

**Desarrollo**

**El Modelo de 4+1 Vistas**

La arquitectura del software se trata de abstracciones, de descomposición y composición, de estilos y estética.

También tiene relación con el diseño y la implementación de la estructura de alto nivel del software.

Los diseñadores construyen la arquitectura usando varios elementos arquitectónicos elegidos apropiadamente. Estos elementos satisfacen la mayor parte de los requisitos de funcionalidad y performance del sistema.

Lo que propone Kruchten es que un sistema de software se ha de documentar y mostrar con 4 vistas bien diferenciadas y estas 4 vistas se han de relacionar entre si con una vista más, que es la denominada vista +1.

Estas 4 vistas las denomino Kruchten como: vista lógica, vista de procesos, vista de despliegue y vista física y la vista +1, que tiene la función de relacionar las 4 vistas citadas, la denomino vista de escenario.

Diagrama

Descripción generada automáticamente

Cada una de las vistas ha de mostrar toda la arquitectura del sistema software que se este documentando, pero cada una de ellas ha de documentarse de forma diferente y ha de mostrar aspectos diferentes del sistema software.

**Vista lógica:** El estilo usado para la vista lógica es el estilo de orientación a objetos. La principal guía para el diseño de la vista lógica es el intentar mantener un modelo único y coherente de objetos a lo largo de todo el sistema, para evitar la especialización prematura de las clases y mecanismos particulares o de un procesador.

**Diagrama

Descripción generada automáticamenteVista lógica:** Se representa la funcionalidad que el sistema proporcionara a los usuarios finales. Se ha de representar lo que el sistema debe hacer, las funciones y servicios que ofrece. Para completar la documentación de esta vista se puede incluir los diagramas de clases, de comunicación o de secuencia de UML.

**Vista de Despliegue:** Se centra en la organización real de los módulos de software en el ambiente de desarrollo del software. El software se empaqueta en partes pequeñas –bibliotecas de programas o subsistemas– que pueden ser desarrollados por uno o un grupo pequeño de desarrolladores. Los subsistemas se organizan en una jerarquía de capas, cada una de las cuales brinda una interfaz estrecha y bien definida hacia las capas superiores.

**Diagrama

Descripción generada automáticamenteVista de Despliegue:** Se muestra el sistema desde la perspectiva de un programador y se ocupa de la gestión del software; o en otras palabras, se va a mostrar cómo está dividido el sistema software en componentes y las dependencias que hay entre esos componentes. Para completar la documentación de esta vista se pueden incluir los diagramas de componentes y de paquetes de UML.

**Vista de Procesos:** La arquitectura de procesos toma en cuenta algunos requisitos no funcionales tales como la performance y la disponibilidad. Se enfoca en asuntos de concurrencia y distribución, integridad del sistema, de tolerancia a fallas

**Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteVista de Procesos:** Se muestran los procesos que hay en el sistema y la forma en la que se comunican estos procesos; es decir, se representa desde la perspectiva de un integrador de sistemas, el flujo de trabajo paso a paso de negocio y operacionales de los componentes que conforman el sistema. Para completar la documentación de esta vista se puede incluir el diagrama de actividades de UML.

**Vista Física:** Se muestra desde la perspectiva de un ingeniero de sistemas todos los componentes físicos del sistema, así como las conexiones físicas entre esos componentes que conforman la solución (incluyendo los servicios). Para completar la documentación de esta vista se puede incluir el diagrama de despliegue de UML.

**Vista Física:** Esta toma en cuenta primeramente los requisitos no funcionales del sistema tales como la disponibilidad, confiabilidad (tolerancia a fallas), performance (throughput), y escalabilidad. El software ejecuta sobre una red de computadores o nodos de procesamiento (o tan solo nodos).

**Diagrama

Descripción generada automáticamente+1 Vista de Escenarios:** Esta vista va a ser representada por los casos de uso software y va a tener la función de unir y relacionar las otras 4 vistas, esto quiere decir que desde un caso de uso podemos ver como se va ligando las otras 4 vistas, con lo que tendremos una trazabilidad de componentes, clases, equipos, paquetes, etc., para realizar cada caso de uso. Para completar la documentación de esta vista se puede incluir el diagrama de casos de uso UML.

Una vez que se explicaron las vistas que propone Kruchten y la forma de documentarlas, se puede apreciar que es un modelo bastante bueno para documentar la arquitectura de un sistema software complejo, ya que todos los Stakeholders pueden entender el sistema software que se este desarrollando desde diferentes perspectivas.

Cabe mencionar que Kruchten no dice como se ha de documentar cada vista, sino que es lo que hay que documentar en cada una de ellas, es decir que cuando se diga que la vista lógica se puede documentar de forma grafica con un diagrama de clases de UML, no quiere decir que esa vista se tenga que documentar con ese diagrama, sino que ese diagrama por sus características puede documentar esa vista.

Para la utilización del modelo 4+1 vistas de Kruchten para la documentación de arquitectura de sistemas software se debe de seguir el estándar IEEE1471-2000.

**Opiniones**

**Oscar Renato:**  Para mí, aprender sobre este modelo me fue de gran utilidad ya que me enseño cosas que desconocía sobre la arquitectura de software, sobre cómo se puede trabajar de distintas maneras, esto me enseño que los ingenieros de sistemas se enfocaron en la vista física, y luego en la vista de procesos, además de que realmente me permitió encontrar una clara definición acerca de la arquitectura del software y además me enseño los distintos procesos que se llevan a cabo para estas fases.

**Bibliografía**

<https://jarroba.com/modelo-41-vistas-de-kruchten-para-dummies/>

<http://materias.fi.uba.ar/7510/practica/zips/Modelo4_1Krutchen.pdf>