Modelo “4+1” vistas de Kruchten (para Dummies)

El modelo “4+1” de Kruchten, es un modelo de vistas [1] diseñado por el profesor Philippe Kruchten (http://philippe.kruchten.com/) y que encaja con el estándar “IEEE 1471-2000” (Recommended Practice for Architecture Description of Software-Intensive Systems [5]) que se utiliza para describir la arquitectura de un sistema software intensivo basado en el uso de múltiples puntos de vista.

Foto de Philippe Kruchten

Vale, si por ahora no te has enterado de nada y no estas en 3 o 4 de carrera de Ingeniería del Software (o derivados) no te preocupes es normal, y si estas en 3 o 4 de carrera y aun así no te has enterado de nada, ¡Ponte las pilas YA! Porque estas cosas te deberían (por lo menos) sonar.

Bueno, vamos por pasos. Antes de entrar a explicar mas en detalle el modelo de kruchten vamos a explicar e intentar dejar claro algunos conceptos como por ejemplo qué es un sistema software, qué es una vista y qué es un punto de vista.

Lo primero es saber que es eso de “**un sistema software**”, el cual lo definimos con la siguiente “ecuación” (made in jarroba.com).

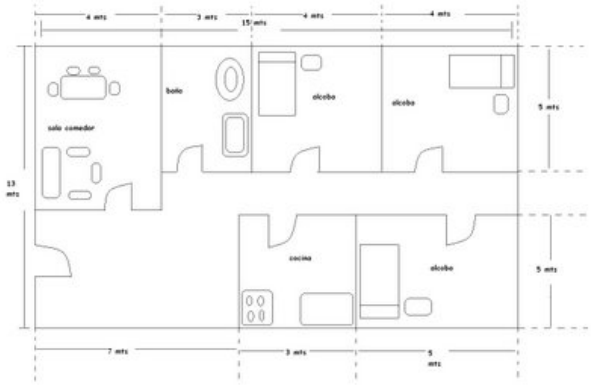
> **Sistema software = Hardware + Software**

Efectivamente, a grandes rasgos un sistema software es un software (mas o menos complejo) que “corre” en un determinado hardware (mas o menos complejo). Por ejemplo, todo el rollo de los “cajeros automáticos” es un sistema software ya que en un “hardware” que llamamos “cajero”, se ejecuta algún tipo de programa (software) el cual nos permite realizar determinadas gestiones.

Otra cosa de la que habla este modelo de Kruchten es sobre los conceptos de vista y puntos de vista, pues bien **una vista** no es mas que una representación de todo el sistema software desde una determinada perspectiva, y **un punto de vista** se define como un conjunto de reglas (o normas) para realizar y entender las vistas.

Bien, sino te ha quedado muy claro que es esto de las vistas y los puntos de vista, vamos a explicarlo con una sencilla analogía del mundo de la arquitectura (de la arquitectura de las casas, edificios y esas cosas):

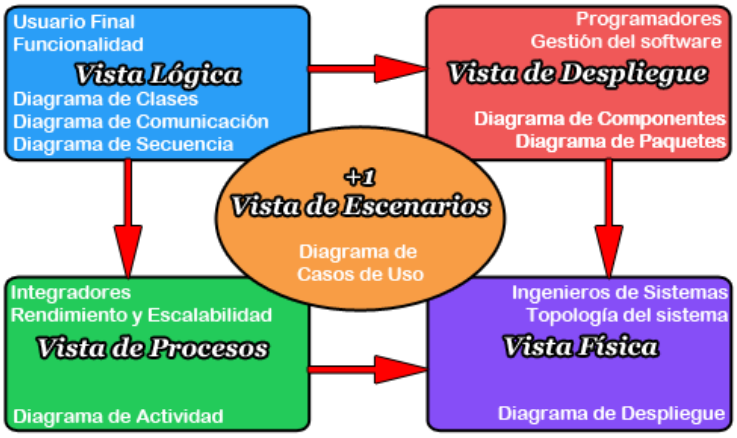
Si un arquitecto nos muestra un plano de una casa (como la de la siguiente imagen), nos esta mostrando **una vista** de la casa y como no tenemos ni idea de arquitectura, cuando nos explique o nos de un documento en el que explique que un determinado símbolo del plano representa a una puerta u otro símbolo representa una mesa, nos estará dado **un punto de vista** para que podamos entender el plano de la casa. Si mas tarde nos mostrase otro plano (o maqueta) de la casa, nos estaría dando otra vista de la casa y nos tendrá que explicar el nuevo punto de vista, es decir, que nos tendrá que explicar que significa cada símbolo u objeto de esa nueva vista.



Bueno pues vistos los conceptos de lo que son las vistas y los puntos de vista, y habiendo explicado que es un sistema software, uno ya se puede hacer a la idea de que va el modelo “4+1” vistas de Kruchten para la descripción de arquitecturas de sistemas software ¿NO?.

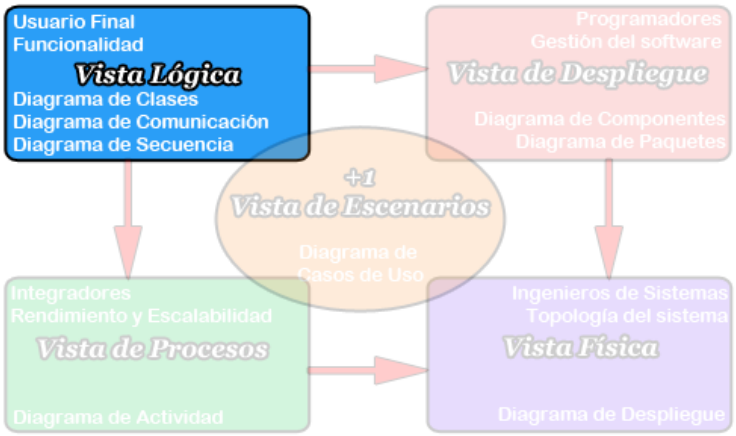
Pues sí, lo que propone Kruchten es que un sistema software se ha de documentar y mostrar (tal y como se propone en el estándar IEEE 1471-2000) con 4 vistas bien diferenciadas y estas 4 vistas se han de relacionar entre sí con una vista más, que es la denominada vista “+1”. Estas 4 vista las denominó 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 denominó vista de escenario.

Cada una de estas vistas ha de mostrar toda la arquitectura del sistema software que se esté documentando, pero cada una de ellas ha de documentarse de forma diferente y ha de mostrar aspectos diferentes del sistema software. A continuación, pasamos a explicar que información ha de haber en la documentación de cada una de estas vistas.



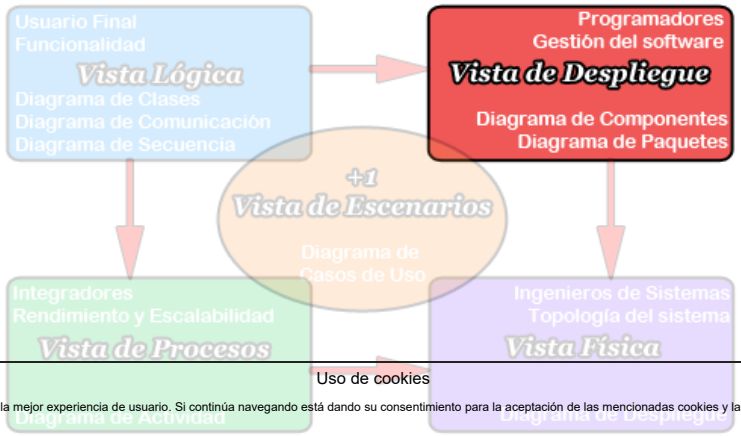
(https://jarroba.com/wp-content/uploads/2012/03/Kruchten.png)

Vista Lógica: En esta vista se representa la funcionalidad que el sistema proporcionara a los ***usuarios finales.*** Es decir, se ha de representar lo que el sistema debe hacer, y las funciones y servicios que ofrece. Para completar la documentación de esta vista se pueden incluir los diagramas de clases, de comunicación o de secuencia de UML.



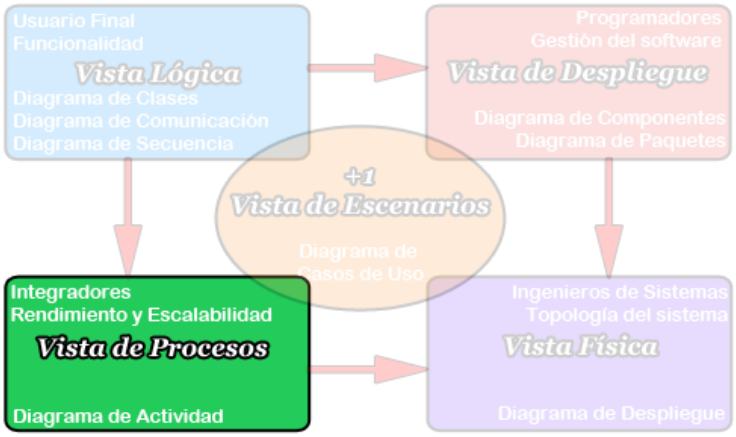
(https://jarroba.com/wp-content/uploads/2012/03/logica.png)

Vista de Despliegue: En esta vista 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 como esta 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.



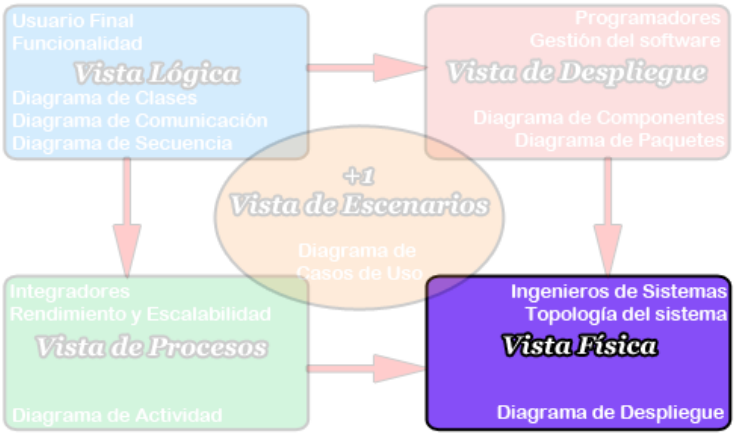
(https://jarroba.com/wp-content/uploads/2012/03/Despliegue.png)

Vista de Procesos: En esta vista 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 actividad de UML.



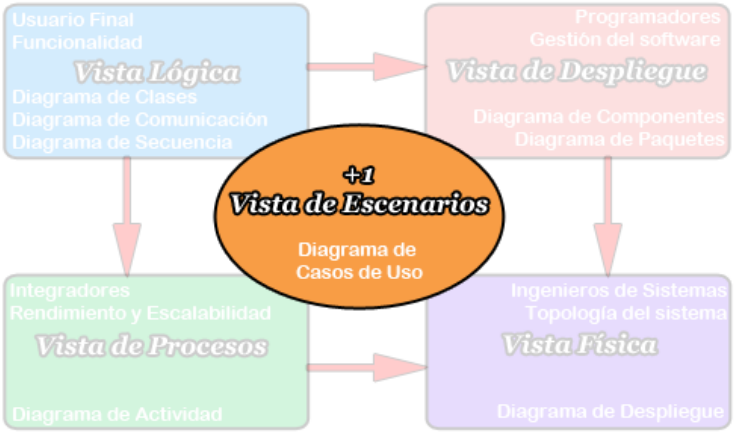
(https://jarroba.com/wp-content/uploads/2012/03/Procesos.png)

Vista Física: En esta vista 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.



(https://jarroba.com/wp-content/uploads/2012/03/fisica.png)

“+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 van 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 pueden incluir el diagramas de casos de uso de UML.



(https://jarroba.com/wp-content/uploads/2012/03/+1.png)

NOTA MUY IMPORTANTE: Hay que dejar claro que Kruchten **no dice de que manera se ha de documentar cada vista**; sino que es lo que hay que documentar en cada vista, es decir que cuando se diga que la vista lógica se puede documentar de forma gráfica 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.

Kruchten no define en ningún sitio “puntos de vista” para hacer vistas, solo define la info que ha de tener cada vista. Lo que pasa que a día de hoy mucho software se hace bajo el paradigma de la programación orientada a objetos. Y casualmente existe una cosa que se llama UML (Lenguaje de Modelado Unificado), que mira tú por donde, representa vistas con diagramas que están formados por “símbolos”, que pertenecen a un LENGUAJE llamado UML (UML es un Lenguaje no una metodología) que conoce (o ha de conocer) todo ingeniero software. Y eso qué quiere decir, pues que un diagrama es una vista, con un punto de vista que ya está muy bien definido en UML. ¡¡A qué ahora las cosas empiezan a cuadrar!!. Pues eso, que con UML nos ahorramos el tener que documentar los puntos de vista al estar ya documentadas.

Y otra cosa también muy importante, es que **cada uno puede representar las vista “como le de la gana” siempre y cuando estén reflejadas en el punto de vista** como ocurrirá en muchos casos en la vista física.

Una vez explicadas 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” (personas interesadas en el proyecto, desde el usuario del sistema hasta el “manda más” de la empresa) pueden entender el sistema software que se esté desarrollando desde diferentes perspectivas.

Para terminar, solo recomendar la utilización de este modelo “4+1” vistas de Kruchten para la documentación de arquitecturas de sistemas software siguiendo el estándar “IEEE 1471-2000”.

Bibliografía

[1] Articulo de Philippe Kruchtens “*Architectual Blueprints – the “4+1” View Model of Software Architecture*”: http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf (http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf)

[2] Wikipedia: http://en.wikipedia.org/wiki/4%2B1_Architectural_View_Model (http://en.wikipedia.org/wiki/4%2B1_Architectural_View_Model)

[3] “*Software Engineering: Principles and Practice*”, 3rd Edición de Hans van Vlient. May 2008.

[4] http://www.notei.com.mx/index.php?id=81 (http://www.notei.com.mx/index.php?id=81)

[5] http://standards.ieee.org/findstds/standard/1471-2000.html (http://standards.ieee.org/findstds/standard/1471-2000.html)

12 thoughts on “Modelo “4+1” vistas de Kruchten (para Dummies)”

josue (http://--) dice:
17/10/2019 a las 14:36 (https://jarroba.com/modelo-41-vistas-de-kruchten-para-dummies/#comment-269163)

Gracias por informacion

Responder

raul b dice:

21/02/2018 a las 14:46 (https://jarroba.com/modelo-41-vistas-de-kruchten-para-dummies/#comment-200156)

Los diagramas relacionados a la capa lógica no son correctos… Los diagramas de clases y secuencia son diagramas de mucho detalle. La vista lógica es la primera que se diagrama y en ese momento nosotros no sabemos que componentes (Vista de implementación o desarrollo)vamos a tener y mucho menos las clases a implementar (Imaginemos que la solución tenga 1000 clases… cuantos diagramas saldrían?). En esta vista lo ideal es diagramar componentes a muy alto nivel (que representen responsabilidades claves) y sus relaciones. La idea es comenzar a visualizar los diferentes patrones/estilos arquitectónicos (cliente-servidor, MVC, N-tier, N-Layer, Pipe n Filters…), que se van a usar en el sistema.

 Responder

pablo levipil dice:

29/08/2016 a las 03:59 (https://jarroba.com/modelo-41-vistas-de-kruchten-para-dummies/#comment-56193)

me ayudo mucho y esta muy bien explicado

good job

 Responder

Harold Campos dice:

28/04/2014 a las 21:20 (https://jarroba.com/modelo-41-vistas-de-kruchten-para-dummies/#comment-212)

Genial, la publicación, dejaron en claro muchos conceptos, usando un lenguaje no muy técnico y de fácil comprensión, aunque pues esa era la idea, es para Dummies.

 Responder

Angel dice:

29/08/2013 a las 22:50 (https://jarroba.com/modelo-41-vistas-de-kruchten-para-dummies/#comment-211)

Me quedó más claro, gracias por la información. Saludos

 Responder

Angel dice:

27/08/2013 a las 20:56 (https://jarroba.com/modelo-41-vistas-de-kruchten-para-dummies/#comment-209)

Por que el Diagrama de Despliegue no puede servir para documentar La Vista de Despliegue?
No se está confundiendo la Vista de Despliegue con la Vista de Implementación?
y la Vista Física con la Vista de Despliegue?
Saludos

 Responder

Richard [Admin Jarroba] (http://www.jarroba.com) dice:

28/08/2013 a las 14:06 (https://jarroba.com/modelo-41-vistas-de-kruchten-para-dummies/#comment-210)

Hola Angel.
Para documentar las vistas, puedes utilizar los diagramas que quieras, siempre y cuando lo reflejes en el punto de vista que corresponderá a esa vista, por tanto si consideras que para desarrollar la vista de despliegue vas a utilizar un diagrama de despliegue, lo documentas en tu punto de vista y listo.

Respecto a lo que comentas, tienes mucha razón sobre lo que dices de la vista de despliegue y de la vista de implementación. No las estoy confundiendo, de hecho en la entrada comento que es lo que hay que documentar en cada una de esas vistas que es lo que importa y lo que se ha de tener en cuenta. El problema esta en las traducciones al ingles sobre como denominan a esas vistas que efectivamente las ponen como «Implementation View» y «Deployment View», pero vamos «Deployment View» es lo que en castellano llamamos vista física e «Implementation View» es vista de implementación aunque mucho la llamamos vista de despliegue, que efectivamente da lugar a la confusión, pero como digo lo que importa es lo que hay que documentar en esas vista y no importa tanto el nombre (o su traducción).

Gracias por la matización.

SL2

 Responder

jarroba dice:

28/12/2012 a las 13:10 (https://jarroba.com/modelo-41-vistas-de-kruchten-para-dummies/#comment-208)

Buenas Carlos.
La palabra «trazabilidad» no es la palabra correcta. Lo correcto seria haber dicho «coherencia entre vistas» ya que si somos bastante técnicos la palabra trazabilidad en ingeniería del software significa otra cosa.

La coherencia entre vistas muestra las posibles relaciones que hay entre las vistas. Como en cada una de las vistas se tiene que representar TODO el sistema se tiene que poder relacionar los elementos en cada una de las vistas.

Por ejemplo y a un alto nivel de abstracción podemos decir que un caso de uso «C» de la vista de escenarios se relaciona con una clase «L» de la vista lógica, con un proceso «P» de la vista de procesos, con una capa «D» de la vista de despliegue y con un elemento «E» de la vista física. Puede ser que las relaciones sean con mas elementos en cada vista, pero la idea es que como en cada vista esta representado todo el sistema se tiene que relacionar que partes de las vistas hacen lo mismo, o que partes intervienen para cumplir un cierto requisito funcional del sistema.

Voy a corregir el comentario anterior para que no lleve a engaño.

Gracias por escribir.
SL2

 Responder

Carlos dice:

28/12/2012 a las 12:41 (https://jarroba.com/modelo-41-vistas-de-kruchten-para-dummies/#comment-207)

Buenos dias

Queria preguntarte sobre la trazabilidad en los escenarios, como relacionas las vistas con los escenarios individuales, si sigue un esquema preciso o simplemente se menciona que componentes de las vistas en estan implicados en un escenario en concreto

Un saludo y gracias por la información del 4+1

 Responder

jarroba dice:

26/12/2012 a las 11:41 (https://jarroba.com/modelo-41-vistas-de-kruchten-para-dummies/#comment-206)

Buenas Eversor

Quizas no exprese bien lo que dije, cuando puse lo de: «con UML nos ahorramos el tener que documentar los puntos de vista al estar ya documentadas», estaria mejor dicho decir que nos «ahorramos» tener que INVENTARNOS una notación para el punto de vista cuando trabajamos bajo el paradigma de la orientación a objetos ya que se puede utilizar el lenguaje UML para ello.

Los puntos de vista siempre han de documentarse y poner que notación se utiliza para ello. En el caso de que se utilize el lenguaje UML poner los simbolos UML que se utilizan. Evidentemente para un Ingeniero Informático que sepa UML al ver una vista representada en UML no tendra que ver el punto de vista al conocer ya el punto de vista, pero otros que no sepan UML tendran que ver que significa cada simbolo de la vista.

En resumen para cada una de las 4 vistas hay que:
1.- Definir los puntos de vista (se utilice o no UML)
2.- Documentar y representar cada una de las vistas

Para el caso de la vista «+1» a parte de hacer los puntos 1 y 2 hay que relacionar esa vista con cada una de las otras 4 vistas (~~trazabilidad~~ Coherencia entre vistas).

Gracias por tu comentario
SL2

 Responder

Eversor dice:

25/12/2012 a las 16:31 (https://jarroba.com/modelo-41-vistas-de-kruchten-para-dummies/#comment-205)

En el IEEE 42010 se puede leer: «An architecture viewpoint is characterized by […] a set of stakeholders interested in how they are addressed […]»

¿Cómo puede ser que no sean necesarios los «puntos de vista» por el simple hecho de usar UML? Puedo entender que usando UML te ahorres el engorro de tener que definir cada modelo que uses, pero poco más.
¿Estoy pasando algo por alto?

Un saludo

 Responder

Alejandro dice:

02/09/2012 a las 22:51 (https://jarroba.com/modelo-41-vistas-de-kruchten-para-dummies/#comment-204)

Más claro, agua. Muy bien explicado todo !!

 Responder

Deja una respuesta

Tu dirección de correo electrónico no será publicada. Los campos obligatorios están marcados con *

Comentario

Nombre *

Correo electrónico *

Web

Publicar el comentario

Este sitio usa Akismet para reducir el spam. Aprende cómo se procesan los datos de tus comentarios (https://akismet.com/privacy/).

(/)

Siguenos en:

(https://www.youtube.com/user/Jarrobaweb) (https://twitter.com/JarrobaWeb) (https://www.facebook.com/jarrobaWeb)

(https://github.com/jarroba?tab=repositories) (https://plus.google.com/103352032205227460108/)

Contacto: jarrobaweb@gmail.com (mailto:jarrobaweb@gmail.com)

VISITAS (HTTPS://JARROBA.COM/VISITAS/)

MAPA DEL SITIO (HTTPS://JARROBA.COM/SITEMAP_INDEX.XML)

By Reimon & Richard