|  |  |  |
| --- | --- | --- |
| **Arquitectura de Software en 4+1 vistas** | | |
| **Nombre:** | Francisco Antonio Noj Avendaño | Foto.jpg |
| **Correo electrónico:** | [francisconoj@gmail.com](mailto:francisconoj@gmail.com) |
| **Tipo artículo:** | **Tutor académico** |
| **Fecha:** | 16/10/2015 |

**Resumen:**

<<Texto de resumen del artículo 3 a 5 líneas>>

**Palabras claves:**

Stakeholders, componentes, procesos, vista, retroalimentación.

**Introducción:**

<<Texto de resumen del artículo 10 a 20 líneas>>

Se puede decir que la Arquitectura de software (AS), es el diseño de alto nivel de cualquier sistema de software. Una vista del sistema que incluye los componentes que forman el sistema, la manera que estos componentes se comportan individual y colectivamente dentro del sistema. La AS muestra cómo se coordinan estos componentes para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión detallando cada componente del sistema.

Para representar la arquitectura de software existen varias formas tal como va evolucionando el software las maneras de representar dicho sistema van cambiando en este artículo se definirá el modelo 4+1 vistas, ya que es una de las más utilizadas en la actualidad.

El uso de múltiples vistas ayuda a que los interesados en el proyecto (stakeholders) puedan observar solo lo que les interesa de la AS. Los stakeholders pueden ser desde desarrolladores, ingenieros de software, administradores de proyectos, hasta los mismos usuarios finales, mostrándole a cada uno lo que le interesa ver. De esta manera se pueden manejar los requerimientos funcionales y no funcionales por separado.

Las vistas de este modelo son:

* Vista Lógica
* Vista de procesos
* Vista de desarrollo
* Vista física
* Y la vista +1 es la de escenarios

**Vista lógica**

En la vista lógica brinda los requerimientos funcionales, estos son los que el sistema debe brindar al usuario final, en los sistemas se busca que sea en su mayoría débilmente acoplado, con esto se logran una alta escalabilidad, ya que los módulos o componentes no dependen entre sí, si se hace un cambio en algún componente este no afecte el funcionamiento de los demás. Se logra a través de los principios de abstracción, encapsulamiento y herencia. Esta descomposición no solo se hace para potenciar el análisis funcional, sino también sirve para identificar mecanismos y elementos de diseño comunes a diversas partes del sistema.

En esta vista se presentan los diagramas: de clase y de paquetes.

**Vista de Procesos**

La vista de procesos toma en cuenta algunos requisitos no funcionales como performance y la disponibilidad. Se enfoca en asuntos de concurrencia y distribución, integridad del sistema, de tolerancia a fallas. La vista de procesos también especifica en que hilo de control se ejecuta correctamente una operación de una clase identificada en la vista lógica.

Se representan los flujos de trabajo paso a paso del negocio y las operaciones que realiza cada componente que conforma el sistema. En esta visa se presenta el diagrama de actividades.

**Vista de desarrollo**

La participación principal de la vista de desarrollo es la organización real de los modelos y componentes de software en el desarrollo del mismo. El software se divide en pequeños paquetes o subsistemas esto se realiza para que grupos de desarrollo tome ciertos paquetes para desarrollar. Los subsistemas tienen una organización jerárquica de capas, donde cada una brinda una interfaz estrecha hacia las demás capas, esto quiere decir lo mencionado anteriormente de estrechamente acoplado, para que cada subsistema pueda crecer independientemente de los demás y este no altere al sistema completo haciendo modificaciones en los demás módulos.

Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, ejecutables, etc. En esta vista se encuentran los diagramas: de componentes y de paquetes.

**Vista física**

En la vista física se representa como se distribuyen los componentes entre los diferentes equipos que forman la red del sistema, es incluyen los servicios que cada uno aporta dentro de la AS. Se toman en cuenta los requerimientos no funcionales del sistema, como los que podrían ser, disponibilidad, performance, escalabilidad, etc.

En esta vista se encuentra el diagrama de deployment.

**Vista +1**

Esta vista podría ser la más importante, ya que es una vista trasversal al resto de vistas, esto se logra a través de los casos de usos, desde un caso de uso podemos ver cómo se van uniendo las demás vistas.

No toda arquitectura de software requiere las “4+1” vistas completas. Las vistas que no son útiles pueden omitirse de la descripción de arquitectura, tales como la vista física si hay un único procesador, y la vista de procesos si existe un solo proceso o programa.

Para sistemas muy pequeños, es posible que las vistas lógica y de desarrollo sean tan similares que no requieran descripciones independientes. Los escenarios son útiles en todas las circunstancias.

Cuando se da un enfoque dirigido por los escenarios (casos de usos), es la funcionalidad más citica, estas funciones son las más importantes, la razón de existir del sistema y la mayor frecuencia de uso, estos riesgos deben ser mitigados de las siguientes maneras:

* Se escoge un pequeño número de escenarios para cierta iteración basado en el riesgo y la criticidad. Los escenarios pueden sintetizarse para abstraer una serie de requisitos de usuario o historias de usuario según sea la metodología a utilizar.
* Se bosqueja una arquitectura preliminar. Los escenarios se describen para identificar las abstracciones mayores (clases, mecanismos, procesos, subsistemas, paquetes, etc.) descomponiéndolos en secuencias de pares (objeto, operación).
* Los elementos de la arquitectura descubiertos se ponen en las 4 vistas de arquitectura: lógica, de procesos, de desarrollo y física.
* Se implementa la arquitectura, se prueba, se mide, y se analiza para detectar errores o potenciales mejoras en cada iteración.
* Se recogen las lecciones aprendidas haciendo feedback (retroalimentación) de cada iteración.

<<Desarrollo del artículo 2 a 4 páginas>>

**Conclusiones:**

* Conclusión 1
* Conclusión 2
* Conclusión 3
* Conclusión “n”

**Referencias:**

*(Estructura según guía de artículos)*

* Referencia 1
* Referencia 1
* Referencia 1

**Imágenes del artículo**

|  |
| --- |
|  |
| **Autor:** José Pérez (Prensa Libre) |
| **Dirección electrónica de la imagen:**  <http://www.prensalibre.com/internacional/EEUU-Migracion-elecciones-legislativas_0_1207079310.html> |