 **INSTITUTO TECNOLOGICO**

**DE CULIACAN**

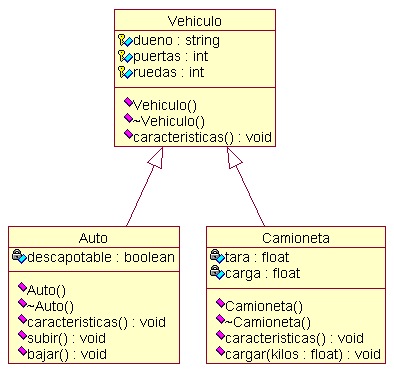
**ACTIVIDAD #6**

**Architectural Blueprints—The “4+1” View Model of Software Architecture de Philippe Kruchten**

**VISTA LOGICA:**

Esta vista se caracteriza por la aplicación de los principios de abstracción, encapsulación y herencia, esto sirve para identificar los mecanismos y elementos de diseño que poseen características similares en el sistema. Lo cual ayuda a los requerimientos funcionales del sistema, para que brinde adecuadamente los servicios a sus respectivos usuarios. De acuerdo a las características mencionadas, podemos ejemplificarlo mediante los diagramas de clases.

En los diagramas de clases se muestran una serie de clases que se agrupan por categorías, se relacionan de forma lógica, teniendo en cuenta, el uso, la asociación, composición y herencia. Dichos diagramas son de gran importancia a la hora de definir el comportamiento interno de los objetos de cada una de estas clases, ya que este tipo de vista tiene un estilo orientado a objetos que sirve para mantener un único modelo u objeto coherente en el sistema.



En la gráfica anterior podemos distinguir 3 clases, vehículo, auto y camioneta, cada una tiene sus atributos y métodos, con sus respectivos tipos de datos. Las dos ultima “auto” y “camioneta” están conectados a la clase vehículo con los conectores respectivos de la herencia, lo que indica que varios atributos y métodos pueden estar tanto en auto como en camioneta.

**VISTA DE PROCESOS:**

Aquí se destacan la concurrencia y distribución, integración del sistema y la tolerancia a los fallos, en otras palabras se encarga de los requisitos no funcionales del sistema. Esta vista la podemos dividir en niveles de abstracción, donde cada uno de ellos realiza alguna función específica. Un ejemplo podría ser el nivel más alto donde los procesos realizan acciones independientes. Cada proceso de la vista cuenta con un conjunto de tareas que pueden ejecutarse.

Podemos representar la vista de procesos mediante los diagramas de actividades. Los cuales consisten en separar las tareas independientes, las cuales se deben programar de forma individual. La división consiste separarlas por niveles, por ejemplos las que tienen mayor importancia en el sistema (tareas principales) se separan de las que tienen un grado de importancia menor. Las primeras se comunican a través de un conjunto de mecanismos de comunicación entre las tareas bien definidas

En las notaciones que se utilizan en esta vista varían según la acción que se desea representar, por ejemplo, los componentes en esta vista son los procesos con sus especificaciones, dentro de esos procesos hay otros procesos que se denotan dentro de el en una representación gráfica, y a su vez algunos procesos pueden repetirse, lo cual se denota como un proceso repetitivo o bien una flecha circular indicando que debe volver a procesarse. En el caso de los conectores también varían dependiendo la indicación que le da un proceso a otro, como los mensajes, tanto en un sentido como en ambos.

**VISTA DE DESARROLLO:**

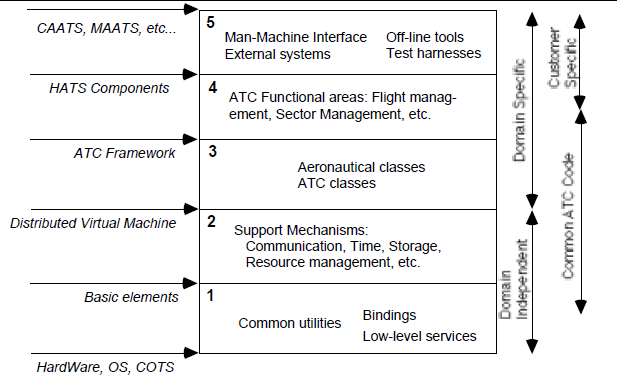
Esta vista sirve como base para la asignación de requisitos, para el razonamiento sobre la reutilización del software, portabilidad y seguridad; a su vez también sirve como base para para una nueva línea de productos. Además posee una característica peculiar, en cuanto al software, pues este se guarda en pequeñas bibliotecas (subsistemas), que se organizan en capas de forma jerárquica, donde cada capa proporciona una interfaz; de esta manera un grupo reducido de desarrolladores pueden llevar acabo el sistema software, o bien una parte del mismo.

La forma de representar esta vista arquitectónica es mediante los diagramas de módulos y subsistemas, con esta representación se pueden mostrar las relaciones de importación y exportación. Antes de poder describir por completo la arquitectura (vista) de desarrollo, previamente se debieron identificar todos y cada uno de los elementos que conformaran el software.

Como se menciona en el primer párrafo, esta vista se organiza por capas (adoptando dicho estilo), cada una de las capas posee una responsabilidad definida, se recomiendan de 4 a 6 capas por subsistema. Siguiendo la regla de diseño nos encontramos con que algunos subsistemas pueden depender de otros siempre y cuando estén en la misma capa o por debajo y así tener una liberación sencilla de la estrategia “capa por capa”.

Al igual que en las vistas anteriores, la representación de los componentes en los diagramas varían dependiendo la actividad que vayan a realizar, cuenta con tres distintas, una para las capas, otra para los subsistemas y por ultimo para los módulos, en caso de los conectores solo existe uno que hace referencia a la dependencia de compilación.

Un ejemplo de la vista de desarrollo (tomado del archivo de referencia) es el siguiente:



Podemos observar que la arquitectura cuenta con 5 capas, cada una con una responsabilidad específica y que de ella dependen otros subsistemas. Según los subsistemas de los que dependen las primeras dos capas son las que interactúan con el hardware, constituyendo así la infraestructura del mismo. En el avance de las capas el sistema se va acercando más a la interacción ya con el usuario, como lo es la interfaz gráfica, etc.

**VISTA FISICA:**

Se toman en cuenta los requisitos no funcionales del sistema, como por ejemplo la disponibilidad, confiabilidad, desempeño, etc. En esta vista el sistema se ejecuta sobre varios nodos de procesamiento, o también podemos llamarlos hardware, que se relacionan con los elementos identificados proporcionados por las vistan anteriores. Aquí se especifican varias configuraciones físicas; una para el desarrollo y las pruebas, o bien para el despliegue del sistema en plataformas distintas.

Para representar gráficamente la vista física contamos con elementos basados en la vista de procesos, que permiten cambios en el mapeo sin afectar el código fuente del sistema.

**Referencias:**

Act 6 4+1view-architecture