En la clase anterior, el profesional invitado fue un analista de sistemas, él nos comentó y nos aclaró el tema de UML, además nos enseñó el tipo de metodologías que hay, como es el RUP, CMML y los casos de uso de estas metodologías.

El analista nos dijo que el UML es un lenguaje de modelamiento unificado que aterriza en una serie de diagramas que ayudan a la construcción y visualización de los sistemas (software). Una de las analogías que él hizo fue que los UML son como los planos de una casa, ya que los planos de la casa, son una especie de plasmación de las ideas de cómo quieres que sea tu casa; los UML, por su parte, son una serie de diagramas que surgen de plasmar nuestras ideas sobre el software que vayamos a desarrollar.

Ahora bien, continuando con esta analogía, el analista nos mencionó que un solo plano no tenía toda la información de la casa, sino que cada plano tendría una información específica y más detallada de la casa; por ejemplo, los planos de las instalaciones eléctricas, los planos sanitarios, etc.

Entonces al igual que en los planos para una casa, los UML no generan todos los diagramas, solamente lo que necesitas, habiendo así, varios tipos de diagramas UML. Los diagramas UML más conocidos que mencionó el analista son los diagramas de actividades, diagrama de secuencia, diagrama de caso uso, entre otros.

En cuanto al desarrollo de un proyecto de software, el analista habló sobre el SDLC (ciclo de vida de proyecto de software), nos explicó que en un proyecto existen dos dominios, el dominio de los sistemas (digital o computacional) y el dominio de los usuarios, haciendo interactuar nuestro mundo digital con el mundo real, transformando la realidad.

Ahora bien, respecto a los dos dominios mencionados anteriormente, el diagrama de clases se adecúa más al dominio de los sistemas, al computacional, ese mundo donde están los programadores; por otro lado, en el dominio del usuario, es recomendable presentar o realizar diagramas de caso de uso, puesto que este tipo de diagramas serán más entendibles para ellos

En relación al ciclo de vida de un proyecto de software, podemos decir que esto es hasta dónde puede durar o ser útil el software; para esto, nos ayuda mucho la programación orientada a objetos, permitiendo externder el ciclo de vida de un proyecto de software. Esto se hace utilizando la idea de objetos como la de encapsular las funciones de vida, lo que no se podía hacer con la programación estructurada.

Respecto a las metodologías, el analista nos dijo que existen varios tipos de estas, los cuales nos ayudan a optimizar el SDLC. Uno de las metodologías que se mencionaron en la exposición fueron la metodología RUP y SCRAMP.

La metodología RUP es una metodología que tiene más de 30 años desde su creación, suministra un enfoque para asignar tareas y responsabilidades dentro de una organización de desarrollo. Es una metodología de desarrollo iterativo para desarrollo de software, según el analista(expositor), se recomienda usarlo en proyectos largos y complejos.

Para desarrollar la metodología RUP se hace una matriz compuesta por fases y disciplinas que están regidas por el tiempo. Las disciplinas o flujos de trabajo de este proceso son los siguientes:

1.-Modelado del negocio.- Defines la terminología, aquí debes entender cómo es que funciona el negocio.
2.-Requerimientos.- Defines lo que quieres.
3.-Analisis y diseño.- Planteamientos de diseño y un mayor análisis de los que se requiere.
4.-Implementación.- Que se empiece la ejecución, en el caso de proyectos de software, será la parte de la programación.
5.-Test.- Son las pruebas que se realizan con tu software ya creado.
6.-Despliegue.- Dónde se va a alojar tu aplicación.
7.-Configuración
8.-Project Managment.- Gestión de proyecto, esto se realiza todo el tiempo
9.-Entorno

Ahora bien, otra metodología que mencionó el expositor fue el SCRUM, que es una metodología ágil en comparación con la metodología RUP, ya que en el caso del SCRUM se puede hacer las mismas fases del RUP de una manera más rápida, habiendo solo 3 flujos de trabajo en esta metodología. Esta metodología, el expositor lo recomienda para proyectos con un tiempo muy acotado.

Sin embargo, lo que tienen en común ambas metodologías son los requerimientos, si no hay requerimientos el dominio del sistema no tiene conexión con el dominio del usuario.

Según el expositor el requerimiento es un proceso de averiguar lo que se debe construir, pero para esto, se necesita desarrollar otro tipo de competencias, fuera del dominio computacional, que pertenecen al campo de inteligencias múltiples.

Como especie de comentario diré que estas inteligencias múltiples fueron desarrolladas por el psicólogo Howard Gardner, el cual dijo que existen 8 tipos de inteligencias, Inteligencia lingüístico-verbal, I. lógica-empresarial, I. espacial, I. musical, I. corporal cinestésica, I. intrapersonal, I. interpersonal, I. naturalista.

Bueno, volviendo al tema de los requerimientos, el expositor hacía mención de las inteligencias múltiples, puesto que a veces la persona que está en el dominio computacional (sistemas) no tiene muy desarrollada la inteligencia interpersonal, la cual te ayuda a comunicarte mejor con el usuario o cliente, y así saber lo que este último quiere. De ahí, si los del dominio computacional no entienden lo que el cliente quiere, entonces el proyecto fallaría debido a que el proceso de requerimientos no fue realizado correctamente.

Entonces si nosotros queremos acercar al dominio computacional con el dominio del usuario, o sea queremos que ambos se comprendan, se entiendan; para ello nosotros debemos usar el requerimiento funcional.

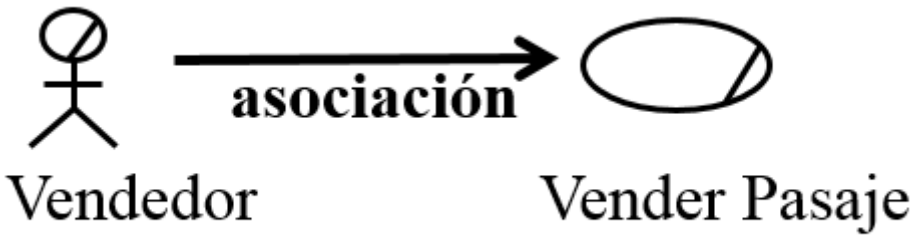
El requerimiento funcional es aquel que está ligado expresamente a un requerimiento del usuario, o sea esto surge de la conciencia del usuario. En cambio, el requerimiento no funcional es aquel que ha pasado por un análisis del dominio computacional, y son ellos los

que determinan si ese requerimiento va o no. O sea esto es algo que se descubre que no necesariamente surge de un requerimiento del usuario.

Todo lo anterior, en relación a los requerimientos, es tratado por la ingeniería de requerimientos. La cual en palabras sencillas, estudia el proceso por el cual se transforman los requerimientos declarados por los clientes, a especificaciones precisas, no ambiguas, consistentes y completas del comportamiento del sistema.

Por último los casos de uso, describen el comportamiento de un sistema desde el punto de vista del usuario, o sea es cómo vas a plasmar ese requerimiento funcional que hace tu cliente en un lenguaje no computacional, en otras palabras en un lenguaje natural accesible al usuario.

Veremos un gráfico, que usó el expositor, para explicar mejor lo que es un caso de uso:



Los casos de uso, tienen la notación de óvalo en este tipo de diagramas y la notación del actor vendría a ser la de una persona, tal como se muestra en la imagen.

En este caso “vendedor” sería el actor y el caso de uso vendría a ser “vender pasaje”, que sería el nombre de requerimiento; y la flecha es el tipo de asociación que hay entre estos(actor y caso de uso). Con este tipo de diagramas se busca que el usuario comprenda lo que el dominio computacional ha hecho, es por ellos que se hace ese tipo de diagramas, puesto que no todos los usuarios saben de códigos y están inmersos en el mundo de la programación.