ENSAYO BASE TEORICA

El Agilísimo

El agilismo es una respuesta lógica a los problemas de la evolución social y tecnológica, que se ha ido planteando, para poder comprender el agilismo habría que repasar un poco de la historia del software ya que es una diciplina que lleva muy poco tiempo y que es muy reciente a nuestra época actual.

El primer computador que corrió con almacenar programas de forma digitalmente fue en 1948, ya que como sabemos es una diciplina que está muy reciente y que no ha transcurrido mucho tiempo como para exigir cambios y estabilidad. Siguiendo con la ley de more los componentes del hadware acaban duplicando su capacidad, por lo que en poco tiempo van saliendo maquinas mas potentes y reducidas en su tamaño, haciendo y procesando miles de datos y reduciendo el costo del hadware, ya que cada vez se saquen computadoras más eficientes, esto nos permite ver que para las computadoras ya no hay restricción como la militar y científica.

Al extenderse el hadware se van generando mas soluciones y mas inquietudes a los sistemas que son más complejos generando nuevas necesidades a una velocidad vertiginosa que implica a los desarrolladores de software, son muy pocos los que se atreven a crear nuevas aplicaciones dándole respuesta a las necesidades son retos que no se resuelven con precisión por que los proyectos no son claros y llegan tarde. En 1968 se acuña el término de ingeniería de software en la NATO (SOFWARE ENGINERRING CONFERENCE) y también en esta conferencia se acuña el termino de crisis en el software para definir los problemas antes mencionados.

Hay que tener en cuenta que anterior mente los tiempos de vida en un producto eran más largos y duraderos que le generaban a las empresas beneficios y era muy rentable, pero fue en la época de los 80 que esto cambio los tiempos de vida de los productos son más cortos ya que van saliendo nuevas aplicaciones al mercado que solo duran unos pocos meses y se van quedando afuera, las empresas tienen que adaptarse a estos cambios, ya que los productos de software no son predecibles ni inmutables ya que los procesos industriales no tienen el mismo efecto ya que algunos son aplicados en máquinas y otro en las personas. Como objetivo es de ver el agilismo como una respuesta a una necesidad.

Los métodos en cascada ha servido como bloque de construcción como ciclo de vida y su visión es el desarrollo del l software tiene que seguir una secuencia de fases y dentro de cada fase o etapa hay metas , y las actividades que van dentro de cada una satisfacen las metas, el ciclo de vida tiene varias actividades como la ingeniería y análisis del sistema que consiste que el trabajo empieza con los requisitos de todos los elementos del sistema, el análisis de los requisitos del software, donde el ingeniero de software debe comprender la información del

software la función y el rendimiento de las interfaces requeridas, el diseño que traduce los requisitos en una representación del software, la codificación donde el diseño debe traducirse en una forma legible para la máquina, la prueba en donde se revisa que se de los resultados que se esperan y el mantenimiento en donde el software sufre cambios para corregir los errores si se encuentran.

Hablando de cifras reales en los procesos de software en estudios que se realizaron por un grupo de profesionales de Massachusetts en 1985 con el fin de analizar las causas de los fracasos en los sistemas informáticos, las cifras no son alentadoras ya que en 1994 el 30% de los proyectos eran cancelados, el 53% eran proyectos problemáticos y el 16% eran de los proyectos exitosos, en el 2004 cambiaron un poco las estadísticas los proyectos exitosos crecieron el 29% y los proyectos fracasados en 71%. Las causas de que los proyectos no triunfen son que no hayan los suficientes recursos, la poca participación de los usuarios, incompetencia tecnológica, objetivos poco claros, los cambios frecuentes en los requerimientos, requerimientos incompletos, falta de soporte ejecutivo, cronogramas irreales y expectativas no realistas. De las causas antes mencionadas la mayoría son fallas humanas. El agilismo es una respuesta l los fracasos del método en cascada, se busca corregir los errores que se dan en el software reconociendo que los fabricantes se equivocan y que el software es propenso a los errores, la finalidad del agilismo es de corregir los problemas clásicos de los programas que le dan valor al as personas que conforman el proyecto y así mismo satisfaciendo las necesidades de los clientes y al desarrollador.

Así como el antiguo modelo también hay etapas de análisis, desarrollo y pruebas, se ejecutan repetidas veces y se les denominan iteraciones, que duran de dos a seis semanas en la cual se habla con el cliente para analizar los requerimientos, se codifica nuevo código, y se mejora el código ya existente, se realizan las pruebas y se le muestran los resultados al cliente. La esencia del agilismo es de adaptarse a los cambios aplicando las diversas técnicas obteniendo los resultados satisfactorios si darle lugar al margen de error y al caos

En la situación actual los países como Gran Bretaña Estados Unidos Japón China tienen un modelo del mundo anglosajon,que se han adaptado a los cambios, en las universidades se les enseña a los estudiantes a ser profesionales, a que se enfrenten a los problemas actuales de la informática y de crear software, de dedicarse a satisfacer a los usuaria ya que el software es una herramienta que mejora la calidad de vida y el uso que le da el usuario.

Los roles que hay en un equipo de trabajo es de saber distinguir las obligaciones y las responsabilidades de cada integrante del equipo, una persona puede asumir varios roles en un mismo día, los papeles más comunes son:

El dueño del software: es el cliente quien indica que es lo que quiere

El cliente: es el dueño del producto y el usuario final

Analista de negocio: También es el dueño del producto por que trabaja con el cliente al mismo tiempo y lo satisface en todas sus necesidades

Desarrolladores: Toman la información del analista de negocio escriben el código, implementan la soluciona deben tener conocimientos de diseño y de usabilidad

Administradores de sistemas: Se encargan de los servidores y los servicios que necesiten los desarrolladores

En todo el mundo se habla de arquitecto de software es el encargado de analizar, tomar las decisiones y escoger de cómo se va hacer el diseño y además de hablar directamente con los clientes, en el agilismo los desarrolladores son arquitectos porque también toman decisiones y escriben código.

El agilismo nos cuesta porque no tenemos en cuenta que hay que esforzarse para crear software no da pereza y no creemos en lo que hacemos todo es antinatural y que no tenemos tiempo para aprender nuevas cosas, también porque nos quedamos en una sola cosa o se terminan los proyectos y además no nos adaptamos a los cambios que todos los días son nuevos en los sistemas, tenemos que aceptar que el software puede ser mejor y que se pueden hacer mejores códigos para mejorar las necesidades y errores que surgen.

Desarrollo dirigido por test (tdd)

Es una técnica de diseño e implementación de software que se centra en tres pilares en implementar las funciones justas del cliente, minimizar el número de defectos que tenga el software y la producción de software modular que se adapte al cambio y que sea reutilizable, tdd convierte al programador en un desarrollador, es la respuesta de varias dudas como las de saber cómo se empieza ,como se hace que es lo que hay que implementar y lo que no, no se trata de escribir código a lo loco si no de diseñar lo que esta requerido

Es importante saber si el diseño requerido es para una aplicación móvil una página web para poder escoger las herramientas adecuadas conformes a las exigencias. El algoritmo tdd tiene tres pasos que son:

***ESCRIBIR LA ESPECIFICACION PRIMERO**: Expresar en código una vez ya se tenga en claro cuál es el requisito.

***IMPLEMENTAR EL COGIGO QUE HACE FUNCIONAR EL EJEMPLO**: Escribir el ejemplo y codificarlo

***REFACTORIZAR**: Es modificar el diseño sin alterar su comportamiento, en este paso se algoritmo tdd se rastrea el código en busca de líneas duplicas y se eliminan refactorizando y se revisa que todo el código cumpla con los requisitos del diseño