TEORIA BASICA CAP I Y II

JOHAN SEBASTIAN JIMENEZ TOBARIA

PROGRAMACION 1

ANDRES ESLAVA

TECNOLOGICA DEL SUR

En este escrito se hablara acerca de la base teórica, que sirve para analizar resultados obtenidos en el trabajo; el agilismo que se puede definir definir como una respuesta lógica a los problemas que han ido evolucionado social y tecnológicamente y el desarrollo dirigido por test también se explicara el modelo las cifras, el manifiesto ágil en que consiste la situación actual, los roles el algoritmo TDD, implementación del código refragmentación, refactorizar, condiciones y recomendaciones, TDD con diferentes tecnologías y en medio de un proyecto, por ultimo realizaremos unas conclusiones.

Desde mi punto de vista el agilismo y el TDD son muy importante si los empezamos a incluir en nuestras vidas pues estos tienen una manera organizada que nos ayuda a mejorar la forma de trabajo y la forma de implementarlo en el ámbito del desarrollo del software.

El agilismo para poderlo entender un poco más es necesario saber un poco acerca de historia sobre el desarrollo de Software ya que fue en 1948 se corrió el primer programa en una máquina, en 1968 se emplea el termino NATO donde nos permite definir los problemas que se presentan en el ámbito del desarrollo y estos a su vez desarrolla tres áreas de conocimiento las cuales son las siguientes.

* Ingeniera del software
* Gestion Predictiva de proyectos
* Producción basada en procesos

La ingeniería del software nos ayudó a implementar la necesidad de una disciplina científica, mientras que la gestión predictiva nos ayuda a planear ejecutar y planear a través de un proceso sistemático.

Para poder realizar un buen desarrollo de software se deben seguir una secuencia de fases, que contribuyen a tener resultados satisfactorios y que su arquetipo tiene diferentes actividades como lo son:

* Ingeniería y análisis del sistema
* Análisis del requisitos del software
* Diseño
* Codificación
* Prueba
* Mantenimiento

En este modelo nos indica que hay una ventaja y que se caracteriza por su sencillez pues este sigue unos pasos a la hora de desarrollar el software. Estos siguen un flujo secuencial, sus requisitos son difíciles de entender al principió y los clientes tienen que tener paciencia hasta llegar a las etapas finales.

La mayoría de profesionales en 1985 crearon un grupo con el fin de recolectar información de proyectos fallidos en información, y se centraron en la implementación de paquetes.

Este grupo revelo 50.000 proyectos fallidos y según standish hay 10 causas para que un proyecto fracase, donde siete de los factores son fallas humanas.

Kent Beck y 17 participantes se reunieron para hablar sobre el desarrollo de software donde muchos de estos sabían las desventajas del modelo de cascada, y a su vez los posiciono como líderes en la industria del desarrollo del hardware. El manifiesto ágil tiene cuatro principios como lo es los individuos e interacciones, el software que funciona, colaboración con el cliente y responder ante el cambio.

El agilismo es un enfoque práctico que da respuestas a fracasos y frustraciones donde hoy en día el desarrollo de software adquiere cada vez más presencia, todos trabajan unidos formando una piña donde el cliente también es parte de ella y su estrategia es el trabajo en grupo. Su esencia es la habilidad para adoptarse a cambios donde se ejecutan diversas técnicas llevando una disciplina para obtener resultados satisfactorios.

La situación actual muestra que muchas personas se han dado cuenta que necesitan un cambio y muchos viven en el mundo anglosajón. En España se enfocan en que los estudiantes sean profesionales capaces de enfrentar su éxito, y en el caso de la informática hace que no se imponga ya que los profesores los instruyen solo a la creación del software, hoy en día muchas empresas contratan personal muy joven y con poca experiencia, también es importante reconocer cuando se le saca el máximo provecho a la situación.

La palabra ágil está siendo mal empleada y esto hace causar falsas ideas sobre su verdadero significado pues estos adoptan una metodología, muchos grupos en internet ayudan y profesionales ofrecen formación y entrenamiento, los roles del equipo ayudan a las personas a realizar un mejor trabajo donde muchos de ellos adoptan distintos roles con el paso del tiempo. Los papeles más comunes en un proyecto de software son los dueños del producto, el cliente, los analistas de pagos, los desarrolladores, arquitectos y administradores de sistemas.

El TDD es una técnica que se implementa en el software para obtener una cobertura muy alta y que convierte al programador en un desarrollador. Hay muchos argumentos claros y directos que nos indican porque es beneficioso manejar el TDD en el libro de Kent y a su vez nos muestra algunas razones con otras figuras de la industria como lo son:

* Calidad del software
* Código reutilizable
* Trabajo en equipo
* Confianza
* Comunicación
* Adquisición de un rol más interesante e inteligente
* Revisión del proyecto

El algoritmo TDD y su esencia es sencilla de poner en practica con mucho entrenamiento esta solo tiene tres pasos los cuales ayudan notablemente a la mejora como lo son la especificación que al tener claro el requisito es expresarlo en forma de código, la implementación consiste en codificar lo mínimo para que se cumpla y refactorizar es modificar el diseño sin alterar su comportamiento. También hay que tener en cuenta las consideraciones y recomendaciones una de ellas es la ventajas que tiene un desarrollador experto frente al principiante ya que el experto se diferencia en el modo en el que implementa el test, las tecnologías desconocidas al implementar el TDD es realizar una función que nos aporte conocimiento, donde se respeta el tiempo que se emplea en aprender con la herramienta y avanzar cada vez que haya confianza en esta, es solo cuestión de aplicar el sentido común para aprender a usar la herramienta.

El TDD en medio de un proyecto es una manera de utilizarlo pero con más recomendaciones ya que este tiende primero a realizar un test y luego realiza la codificación, esta trata con un código legado donde se da por entendido que no tiene ningún test y suelen tener dos API, también pueden sobrecargar métodos para que el código legado y su nueva versión convivan.

En conclusión podemos implementar estas técnicas en las nuevas tecnologías y en la forma de desarrollar software siempre que tengamos una disciplina y una manera organizada para obtener resultados satisfactorios y a su vez tener mas crecimiento personal.