Metodología de Trabajo: GiTecScrum
Proyecto Diagnostico
I Semestre 2015
Algoritmos y Estructuras de Datos II

DESCRIPCIÓN GENERAL DE LA METODOLOGÍA GiTecScrum

Esta metodología esta adaptada para estudiantes de Nereo, esta utiliza Scrum como base para obtener la información a programar, UML, la cual se usará para modelar sprint por sprint, que serán tomadas como etapas de desarrollo, para evitar problemas de dependencias internas dentro del proyecto mismo, y dividir el trabajo en partes iguales. Además se incluirá el manejador de código para que el proyecto este impecable en su rama principal, aprovechando al máximo la funcionalidad de ramas del git (la cual es de las mas rápidas entre los manejadores de código). Seguido de esto, la etapa de desarrollo con prueba y testeo antes de subir el archivo, además de realizar “scrum dailys”, y por ultimo la prueba y testeo final del proyecto buscando bug´s y arreglándolos o reportándolos (en el caso de que no pueda repararse).

DESCRIPCIÓN POR PASOS

Obtención y aclaración de datos.

Será leído el documento que explica el proyecto en sí, se estudiará hasta tener una lista de historias completa (“story” de scrum) que sea comprensible para el grupo. Anotar las horas de disponibilidad por programador y los roles que cada uno tomará, además de anotar un horario fijo para tener control sobre el manejo de horas.

Priorizar tareas.

Tras analizar los datos del documento que contiene la explicación proyecto, y tener la lista de historias, se deben priorizar según su dependencias, las que no tengan dependencias van de primero, y así hasta escalonar y llegar a la etapa que mas dependencias tenga, la prioridad va del 1 al 3 siendo 1 la prioridad mas alta y por ende tratarla de primero y 3 la prioridad mas baja la cual se tratara de último, si las dependencias pertenecen a un grupo concreto, estas pueden ser escritas por un solo programador para evitar problemas de merge, pues si se requieren clases para crear otra, el programador estará tentado a

escribirla y crear un conflicto en la rama principal, aunque no inmediatamente, que puede ser el caso, sería problemático perder el tiempo solucionando este tipo de problemas.

Modelado según la prioridad.

Según la prioridad de las historias se modelarán en UML todas aquellas que tienen prioridad 1, luego la 2 y por último la 3, cuando el modelado de prioridad 1 ha sido terminado y documentado, se puede empezar con la etapa de desarrollo. Mientras se programa, también se adelanta el modelado, esto con el fin de evitar problemas de dependencias y además adelantar la programación sin tener mayor problema pues lo que se esta escribiendo de código serán las partes mas elementales que no forma un programa pero argumentan la base de uno, (explicar aquí).

Programación y manejo de código.

Cada clase hará lo que se solicita en el diagrama UML, además de tener las caretas o interfaces de cada una de las clases. Además se escribirá una clase por commit, la cual no se crea en la rama principal del proyecto si no en una rama diferente, cuando la clase haya sido terminada se prueba, se agrega a la rama principal,la clase puede tener diferentes commits en etapa de desarrollo de la misma, pero no pueden agregar se los commit en la rama principal, si no en una rama alterna. En el manejo de código cada vez que se agrega una clase o se cambia, **tiene que usarse otra rama de trabajo diferente a la rama principal,** esto con el fin de evitar al MÁXIMO PROBLEMAS DE MERGE.

Testeo, arreglos y reporte de bug´s

Se buscan fallos en el sistema, bajo condiciones especificas y tiene que ser grabado cada 20 minutos, para buscar las condiciones especificas que causan el problema, se recomienda usar kazam, con el codec de *h264 mp4* y usar *grabar pantalla completa*.