CUADERNO DE PRÁCTICAS

(TERCERA ENTREGA)

Planificación y Gestión de Proyectos Informáticos

Máster Profesional en Ingeniería Informática

Universidad de Granada

Curso académico 2022/2023

Índice

Práctica 8

Gestión de calidad

Duración

1 sesión

Objetivos

- Profundizar más en las actividades a realizar por un equipo de gestión de calidad: "no es solo hacer pruebas".

- Definir un plan de acción para unas situaciones concretas que suponen un cambio sobre el proyecto actual y la forma de trabajar del equipo.

Tareas

1. Leer detenidamente los siguientes artículos como material complementario para este tema y resumir en media página cuáles son las tareas principales de gestión de calidad de un proyecto informático.

   http://www.calidadytecnologia.com/2014/04/Gestion-Profesional-Calidad-Proyecto.html

   https://iveconsultores.com/sistema-de-gestion-de-calidad/

2. Basándonos en el ejemplo base de las prácticas, se pide la gestión de la calidad del proyecto sobre un análisis de las peticiones de cambio del anexo, proponiendo un plan de acción para cada una de ellas.

Anexo práctica 8

Peticiones de cambio que surgen dentro del proyecto, durante el desarrollo.

		Petición de cambio
1	Recursos humanos	Se necesita contratar durante dos meses a un programador extra. No afecta al tiempo de entrega.
2	Recursos materiales	Se necesita contratar un servidor más potente para dar soporte a la administración de usuarios o procesamiento de datos
3	Tiempo: retraso	Las actividades relacionadas con implementación durarán dos semanas más de lo planificado inicialmente.
4	Costes: aumento	El sueldo mensual de los trabajadores se incrementa en un 2%.
5	Requisitos	Las aplicaciones desarrolladas deben tener los colores y logos del cliente, lo exige su normativa.
6	Requisitos	Hay un nuevo requisito funcional por parte del cliente. Proponed uno concreto que afecte en vuestro proyecto al cambio de diseño en la arquitectura del sistema para poderlo afrontar.
7	Diseño, Metodología	Se va a utilizar IFML para el modelado de la interfaz de usuario
8	Alcance, Metodología	Se decide que una persona del equipo supervise si los objetivos del proyecto se están cumpliendo o no durante su desarrollo y que ésta revise todos los entregables.
9	Pruebas o incidencias, Metodología	Se va a crear un sub-equipo específico para gestionar las incidencias de las aplicaciones creadas e instaladas por la empresa de desarrollo y se van a hacer pruebas de regresión tras cada cambio que se realice.

Completa la siguiente tabla con un plan de acción de calidad para cada petición:

	Acción para garantía de calidad: Qué se hará y porqué	**Cuándo**	**Consecuencias acción de calidad**
1			
2			
…			

En la primera columna deben ponerse **actividades relacionadas con la calidad**. Ejemplos de ellas son: formación, reuniones de coordinación, informes de resultados y de toma de decisiones, revisiones de resultados, documentación de los cambios, seguimiento de estándares, seguimiento de metodologías ágiles, comunicación con el cliente, etc.

Las **consecuencias de la acción** dependen del momento en el que se realicen (**cuándo**), por ejemplo, no afecta lo mismo a la calidad de un proyecto si se contrata a una persona con poca formación al inicio del proyecto que en los últimos meses.

En la columna de **consecuencias de la acción** para cada petición, se debe indicar si afecta a tiempo, costes, alcance, metodología de trabajo, etc., y cómo afecta (por ejemplo, se incrementan los costes del proyecto en x, se retrasa la entrega del proyecto en y, etc.).

Práctica 9

Gestión de la configuración del software

Duración

1 sesión

Objetivos

- Utilizar y profundizar en varias herramientas de gestión de cambios y/o configuración del software. En este caso, como ejemplo, vamos a utilizar Git para la gestión de cambios del software, y Jira para el seguimiento de incidencias.

Tareas

- Vamos a simular el proceso de seguimiento y control en la implementación de tareas de un proyecto utilizando Git. Para ello, el equipo debe crear un proyecto en una plataforma online como GitHub. El proyecto puede ser público o privado; en caso de ser privado, debe dar acceso al profesor (jmoyano@ugr.es) para que pueda acceder.
    - El proyecto tendrá una rama principal *main*, donde se irán incorporando los cambios, una vez que funcionen.
    - Inicialmente, simularemos la implementación de dos tareas de forma paralela. En cada tarea deben participar, como mínimo, dos miembros del equipo. Además, cada una de las tareas tendrá asignada un responsable.
        - La información de las ramas existentes, los miembros del equipo que trabajan en cada rama, y el responsable de cada una de ellas, deberá ir en un fichero *README.md* que se inicializará en la rama *main* (lo hará uno de los miembros del equipo, que actúe como responsable del proyecto).
        - Una vez inicializado el fichero README.md, sería interesante dirigirse al repositorio en GitHub y ver cómo ha quedado la portada del proyecto.
    - El responsable de cada tarea creará una rama a partir de la rama principal, donde se inicializará dicha tarea. Esta rama tendrá un nombre concreto que identifique la tarea (p. ej.: *TareaA*). Para simular la inicialización de la tarea, el responsable creará al menos dos ficheros de texto con cualquier contenido (estos ficheros simularán los ficheros de código).
        - En este punto, tras inicializar las dos ramas, sería interesante entrar al repositorio en GitHub y: a) ver cuántas ramas hay; b) ver qué avance hay en cada rama; c) analizar por qué es así.
    - Cada miembro del equipo que participe en una tarea trabajará sobre la rama correspondiente; es decir, en la rama *TareaA* se irá desarrollando lo relativo a dicha tarea, independientemente de las otras tareas paralelas en el proyecto. Para simular este avance en la implementación, cada miembro modificará uno o varios de los ficheros de texto en dicha tarea. Cada miembro debe simular como mínimo, dos *commits* dentro de la rama para dar por finalizada su contribución.

- La primera de las tareas finalizará antes, por lo que una vez todos los miembros han finalizado con la implementación de la primera tarea, el responsable de la tarea se encargará de unir la rama de la tarea concreta con la rama principal (suponemos que la segunda de las tareas sigue en desarrollo).
    - En este punto, sería de nuevo interesante volver al repositorio en GitHub y analizar ahora cómo se encuentra la rama *main*.
    - También es interesante utilizar el comando *git hist* (ver diapositivas del curso de git) para visualizar el avance.
- Creamos una nueva rama para una tercera tarea a partir de la rama principal (nótese que esta rama incluirá ya la implementación de la *TareaA*, pero no de la *TareaB*).
    - El responsable de la *TareaC* inicializará la tarea creando la nueva rama, modificando el README.md con la información de la nueva tarea, y creando los ficheros correspondientes para esta tarea.
    - Seguimos el mismo proceso anterior: los miembros participantes en esta tarea hacen varios *commits* simulando el desarrollo de la tarea.
- Respecto a esta nueva tarea, vamos a simular la resolución de conflictos. Para ello, al menos dos de los miembros que estén trabajando en la tarea, modificarán el mismo fichero localment. Cuando ambos intenten hacer *push* de sus cambios, ocurrirá el conflicto. A aquel miembro que le aparezca el conflicto (será el segundo que intenta hacer *push*) deberá solucionarlo, y publicar los cambios en el repositorio.
- En algún momento, las dos tareas que seguían en desarrollo se completan, y se unen en la rama *main*. El responsable de cada una de las tareas será el encargado de hacerlo.
- Una vez finalizadas todas las tareas, crearemos una versión del software (*git tag*) como hito del proyecto, que esté accesible mediante una etiqueta.
    - Sería interesante visualizar esta versión en GitHub.
- En este punto, conectamos nuestro repositorio GitHub con un panel de gestión del proyecto en Jira.
    - Creamos un proyecto en Jira.
    - Conectamos con nuestro repositorio en GitHub (Panel izquierdo → Desarrollo → Código).
    - Creamos una incidencia en nuestro proyecto. Nos fijamos en la clave de la incidencia, que será del tipo *CLAVE-XX*.
    - Volviendo a Git, un miembro cualquiera del equipo (que actuará como responsable de la incidencia) creará una nueva rama, incorporando la clave de la incidencia en su nombre, como por ejemplo: *git checkout -b <CLAVE-XX>-<branch-name>*.
    - Simulamos un *commit* dentro de dicha rama para solventar la incidencia, habiendo modificado uno o varios ficheros. Recuerde incluir la clave de la incidencia en Jira también en el *commit*,

como por ejemplo: *git commit -m "<CLAVE-XX> <summary of commit>"*. Posteriormente, el responsable de la incidencia unirá la rama con la rama principal.

        - Vea como desde Jira se puede observar, por ejemplo en los paneles de incidencias, la información de las ramas y *commits* relacionados.
    - Por último, volvemos a crear una nueva versión del software donde se ha solucionado ya la incidencia.

Entregables de esta práctica

- Enlace al repositorio de GitHub. Recuerde que, si el repositorio es privado, debe haber dado acceso al profesor anteriormente. Incluya además los enlaces a las dos versiones del software etiquetadas.
- Enlace al proyecto en Jira. De igual modo, debe haber incluido al profesor (jmoyano@ugr.es) en el proyecto para que pueda entrar.

Práctica 10

Seguimiento y control de proyectos. Retrospectiva.

Duración

1 sesión

Objetivos

- Entender el uso de las retrospectivas como herramientas útiles para la mejora continua de procesos.
- Reflexionar sobre lo acontecido durante la ejecución del proyecto con el objetivo de evaluar las prácticas utilizadas y ajustar su uso en el futuro.

Tareas

- Busque información en Internet sobre formas de "conducir" una retrospectiva de un proyecto, también conocida como "análisis post-mortem" del proyecto.
- Utilice la información obtenida para elaborar una plantilla que permita documentar las lecciones aprendidas y las propuestas de mejora que se obtienen durante la realización de una retrospectiva.
- Rellene la plantilla diseñada con las lecciones aprendidas durante la ejecución del proyecto de prácticas. En particular, su retrospectiva debe responder a preguntas como las siguientes:
    - ¿Cuál era el plan al comienzo del proyecto? ¿Cómo cambió a lo largo de su ejecución?
    - ¿Qué sabe ahora que le hubiese gustado saber al comienzo del proyecto? ¿Cómo habría cambiado el proyecto de haberlo sabido antes?
    - ¿Qué aspectos salieron rematadamente mal durante el proyecto? ¿Por qué?
    - ¿Qué fases o etapas del proyecto habrían necesitado más tiempo para poder ejecutarse de una forma más adecuada, teniendo en cuenta las restricciones temporales ya impuestas?
    - ¿En qué fase se tuvieron que repetir tareas ya realizadas? ¿Cómo podría haberse evitado?
    - ¿Qué herramientas se utilizaron durante el proyecto? ¿En qué funcionaron bien? ¿En qué funcionaron mal? ¿Cómo cambiaría el uso de herramientas de cara a proyectos futuros?
    - ¿Fue efectivo el proceso utilizado a lo largo del proyecto? ¿En qué aspectos funcionó razonablemente bien? ¿En qué aspectos habría que mejorarlo?
    - ¿Cuáles son los aspectos más relevantes de este proyecto que resaltaría para compartirlos con el gestor del proyecto y el resto de su equipo de cara a proyectos futuros?
- Especifique, al menos, tres propuestas de mejora de cara a futuros proyectos. Sus sugerencias se tendrán en cuenta de cara a futuras ediciones del máster.

Entregables de esta práctica

- Ampliación del documento con la retrospectiva del proyecto y las propuestas de mejora.