Identificación y Gestión de Riesgos en el Proyecto

Documento hecho por el equipo de trabajo 1: Esports
Desarrollo de Software en Equipos
Universidad de los Andes, Bogotá, Colombia
Fecha de elaboración: mayo 13 de 2023

1. Definición de riesgos

En el contexto del proyecto de software que el equipo se encuentra desarrollando, se discutieron distintos riesgos presentes o potencialmente posibles y, a continuación, se encuentran definidos:

1. **Si la curva de aprendizaje es más larga de lo esperado entonces se tendrá retrasos importantes en el cronograma del proyecto:**

Este es el primer riesgo que fue identificado, consiste en la dificultad inherente de aprender nuevas tecnologías que antes no se habían utilizado y que requieren un tiempo para comprender. En el caso del curso, se presenta una nueva tecnología y en una o dos semanas ya hay que entregar una práctica de la misma.

Probabilidad de ocurrencia: Alta, de hecho, ya se han visto consecuencias. Por ejemplo, aprender el framework 'Angular' junto a su estructura de ejecución ha sido complejo y debido a esto, se ha atrasado la entrega de resultados en las últimas semanas de trabajo (semana 12 y 13).

2. **Si se presentan problemas en el funcionamiento del código anteriormente escrito por el equipo, entonces habrá que dedicar tiempo adicional para realizar correcciones y esto puede retrasar en gran medida los avances semanales.**

Este segundo riesgo se relaciona con la calidad y correcto funcionamiento del código que se ha construido y subido al repositorio hasta ahora. Es posible que, en las últimas etapas de desarrollo del Front, al probar la aplicación, aparezcan errores indeseados relacionados con el código anteriormente entregado. Por ejemplo, podría haber un error no identificado antes en un servicio del back.

Probabilidad de ocurrencia: La probabilidad está en un nivel **bajo-medio**, pues en realidad la integración de software como 'Jenkins' y 'SonarQube' junto a las pruebas de ejecución del programa y sus elementos están precisamente pensados para mitigar esta realidad. Sin embargo, es posible que aparezcan problemas en partes del código que antes habían pasado desapercibidos debido a carencia de pruebas de integración.

2. Acción de mitigación

Para el primer riesgo: **La curva de aprendizaje de las nuevas tecnologías es más larga de lo esperado**, es posible llevar a cabo las siguientes acciones que reducen su probabilidad de ocurrencia:

- Dedicar tiempo todos los días para avanzar en el entendimiento de las herramientas de software. Por ejemplo, para el caso de Angular, se le puede dedicar 30 minutos o 1 hora diaria a seguir un curso en línea (Youtube o Coursera, por ejemplo) y lograr el fin de semana, con mejor comprensión del tema, desarrollar la parte de la entrega individual correspondiente.
- Pedir ayuda al profesor o monitor para entender ciertas particularidades del software que se está usando. Por ejemplo, se pueden ver clases de Angular en línea, guardar preguntas que se tengan y comentarlas al profesor en la clase o al monitor por medio del correo electrónico.

Para el segundo riesgo: **Presentación de problemas en el funcionamiento del código anteriormente escrito**, es posible llevar a cabo las siguientes acciones que reducen su probabilidad de ocurrencia:

- Elaborar pruebas de forma detenida que alcancen a cubrir la mayoría de los casos posibles dentro del código presentado y en el marco de los requerimientos que se están resolviendo.
- Instalar una herramienta de 'Linter' como el Linter de SonarQube en VSCode para garantizar la escritura de código de calidad, que siga buenas prácticas y esté menos propenso a fallar inesperadamente.
- No copiar y pegar código de los tutoriales del curso, escribir todo desde cero, para ser cuidadoso con el contenido.

3. Plan de contingencia

En caso que se materialicen los riesgos identificados, se pueden realizar las siguientes acciones para reducir su impacto negativo en el proyecto:

3.1 Si se materializa el riesgo: **La curva de aprendizaje de las nuevas tecnologías es más larga de lo esperado:**

1. Definir un plan de aprendizaje continuo, que incluya en toda una semana una dedicación horaria a aprender de forma individual la o las tecnologías que se dificultan.

2. Desde la semana siguiente al momento en que se presente el problema, comenzar a cumplir el plan de aprendizaje. Para esto, se puede dedicar de 30 a 60 minutos diarios todos los días y guardar las preguntas que surjan para resolverlas en clase o por correo.
3. Desde el comienzo de la semana, revisar los entregables que deben subirse al repositorio ese fin de semana, con la finalidad de prever la carga en tiempo que va a requerir el trabajo individual y grupal.
4. Si una entrega implica usar la tecnología cuyo aprendizaje se ha dificultado, iniciar su desarrollo al menos 3 días antes. Elaborar un plan de qué se hará primero y así en adelante.
5. Para el fin de semana, avisar al equipo a más tardar el Domingo por la mañana si hay problemas de entendimiento con la tecnología, para planear una posible solución o avisar al profesor para que pueda hacer una extensión del plazo de entrega.

3.2 Si se materializa el riesgo: **Presentación de problemas en el funcionamiento del código anteriormente escrito:**

1. Al detectarse el problema, el primer paso será entender la naturaleza del error. Esto puede incluir la revisión los diferentes 'Log' existentes en SonarQube, Jenkins o los mismos ambientes de edición del código.
2. Al detectar un error en código previamente escrito, se informará inmediatamente al equipo de trabajo. En caso de que el error se de en código escrito por una persona particular, pedirle que arregle el error y suba un commit a la rama principal.
3. La persona o personas cuyo código escrito presentan errores, deben trabajar de forma individual o conjunta para realizar correcciones y probar de nuevo cada cambio hecho.
4. Una vez que se han probado y verificado las correcciones, se pueden probar en el entorno de producción, las pruebas las pueden realizar todos los integrantes del equipo, para comprobar que todo fue arreglado.
5. Documentar todos los detalles del incidente, incluyendo la descripción del error, por qué ocurrió y cómo se solucionó. Esta documentación se puede compartir al equipo por el chat de WhatsApp para que todos tengan en cuenta estos cambios para parchar posibles imperfecciones en el código escrito anteriormente o evitar cometer los mismos errores en el futuro.