Estrategia Inicial de Pruebas

La verificación y la validación son dos áreas de CMMI que podemos abordar gracias a ésta estrategia inicial de pruebas, en la cual, se definen las características que requiere el departamento para alcanzar los objetivos de la misma. A lo largo de este documento se definen cada uno de los componentes de la estrategia inicial de pruebas junto con una breve explicación de lo que se debe de hacer en cada uno de los 13 puntos que se deben de cubrir. En **negritas** y en morado se encuentra cada una de las estrategias por las que opta y la justificación de por qué es la más adecuada para Cave Labs.

1. Estrategia de reclutamiento

Para establecer el equipo se debe de determinar el tipo de integrantes que realizarán las actividades de pruebas. En el caso de Cave Labs, todos los miembros de cada uno de los 3 equipos es parte del equipo de pruebas. La estrategia de equipo de pruebas por la que opta Cave Labs es la de **Especialistas Generalistas,** la cual consiste en armar un equipo de pruebas en el que todos los miembros tengan un entendimiento general del proceso de pruebas pero que también estén especializados en un área específica cada quien. También aplica ésta estrategia porque los miembros desean adquirir nuevo conocimiento y habilidades de éste proceso.

2. Estrategia de equipo

Para organizar al equipo se debe de decidir cómo los encargados de calidad y pruebas trabajarán como parte del equipo de entrega para que todo el departamento entienda la organización del equipo. La opción por la que opta Cave Labs es por la de **Whole Team Testers** debido a que es la estrategia que provee una colaboración mejorada y más rápida; además de que es la más adecuada para especialistas generalistas.

3. Nivel de detalle

Para establecer el plan inicial se necesita identificar el nivel de detalle de documentación que la estrategia inicial de pruebas tendrá; para así asegurarse de producir la cantidad adecuada de documentación. Cave Labs elige el nivel de detalle **Lightweight Overview** ya que es la opción que ofrece un nivel suficiente de detalle para la mayoría de los miembros del equipo, y es el más ideal para el ambiente en el que se está trabajando.

4. Aproximación de pruebas

Para generar una aproximación inicial de prueba se debe de identificar con base en las habilidades requeridas por el equipo de pruebas al igual que las herramientas requeridas. **Blackbox** será la aproximación tomada ya que nos

habilita probar una gran cantidad de escenarios y puede ser muy buen punto de partida para los esfuerzos de pruebas iniciales.

5. Estrategia de desarrollo

Para generar una aproximación inicial de diseño y programación se debe de identificar qué habilidades y herramientas de prueba necesita el equipo. Para esto, la opción por la que Cave Labs se inclina para ésta estrategia inicial es por la de **Test-after programming,** debido a que es más fácil para el equipo realizar este tipo de pruebas al requerir habilidades que cualquier miembro del equipo puede ejecutar sin problema alguno. Además es el más recomendado por los coaches del departamento para la fase inicial.

6. Estrategia de plataforma(s) de ambientes de pruebas

Para elegir la(s) plataforma(s) de entorno de prueba debemos de definir cómo desplegar las nuevas plataformas. La opción por la que más se inclina el departamento es por la de **Cloud-based** debido a que es la más potencial para el uso de una plataforma de pruebas con la ventaja de ser rápido y fácil de acceder. Además se trabajará usando GitHub para control de versiones.

7. Estrategia de ambientes de prueba equivalentes

Para elegir la estrategia de equivalencia de plataforma se debe de identificar la aproximación de qué tanto representarán nuestros ambientes de prueba nuestros ambientes de producción. La opción de **Production Equivalent** debido a que es la que provee un mayor nivel de garantía. De tal manera que nuestro ambiente de pruebas es una aproximación cercana o exacta a nuestro ambiente de producción, contando con las mismas condiciones de hardware y software.

8. Estrategia de pruebas no funcional

Para elegir la estrategia de prueba de requisitos no funcionales se deben de considerar que los requisitos funcionales tienen distintas categorías que deben de considerarse. Todas las siguientes serán cubiertas por el departamento:

Seguridad: Identificar problemas de seguridad antes de que sucedan.

Usabilidad: Descubrir errores de usabilidad mientras sean fáciles de identificar.

Rendimiento: Encontrar errores de rendimiento antes de lanzar la solución a producción.

Concurrencia: Asegurar que la solución trabaja de manera simultánea con diferentes usuarios.

Resiliencia: Asegurar que la solución opera por un largo periodo de tiempo.

9. Estrategia de suite de pruebas

Para automatizar las suites de pruebas debemos de tener bien definido como configurar la herramienta de regresión de pruebas. Para ello la **Single Regression Test Suite** es considerada como la más viable ya que apoya a las pruebas en situaciones directas y requiere la creación y el mantenimiento de un solo ambiente de pruebas.

10. Fuente de datos de prueba

Para obtener los datos de prueba debemos de considerar que esa información aporte a los esfuerzos de pruebas del equipo. Para nuestro caso, como volume testing no es la prioridad, **Original Solution Data** es la opción indicada: Es una fuente fácil para información de pruebas precisas y es la más indicada para el equipo ya que como bien dice la definición, “la información será insertada en vivo” por el equipo de pruebas.

11. Estrategia de automatización

Para automatizar las compilaciones se debe de determinar primero el nivel de automatización que se debe de implementar para la prueba. **Continuous Integration (CI)** es la opción más adecuada ya que una vez que algo se revisa, la solución se vuelve a construir. De tal manera de que si es necesaria, es útil ya que una vez que la solución es probada, el análisis del código se vuelve opcional.

12. Reporte de defectos

Para informar los defectos dentro del departamento todo el equipo debe de conocer las herramientas necesarias para resolver defectos. **Conversation** es la técnica identificada como la más eficiente para resolver el problema de manera rápida, ya que a pesar de reducir la documentación, permite informar de la manera más eficiente al desarrollador sobre el defecto para corregirse lo antes posible.

13. Estrategias de control de calidad

Para controlar los esfuerzos de calidad, el equipo busca controlar la calidad del trabajo que produce. ésto específicamente a través de la implementación de una técnica llamada **Non-solo work,** la cual consiste en trabajo en parejas para implementar buenas prácticas en conjunto y así poder tener un mejor aprendizaje, adquirir habilidades y comunicarse mejor. A pesar de que esto puede volver el desarrollo un poco más lento que otras estrategias, es el que lleva a un nivel de calidad más alto y a un menor costo de reparación de daños.

Referencia:

Develop Initial Testing Strategy. Disciplined Agile Delivery. Consultado el 15/02/18 de: http://www.disciplinedagiledelivery.com/process-goals/develop-initial-test-strategy/#TestApproaches