Bien, comenzamos entonces la unidad 3 y vamos a mostrarles, a ver, bien, en esta unidad vamos a aprender cómo se diseñan los casos de prueba. Seguimos viendo casos de prueba, porque como les decía anteriormente, los casos de pruebas son lo más importante que hacemos en nuestro nuestro trabajo diario como testers.

Entonces, ¿qué vamos a repasar en esta unidad? Vamos a ver los principios del testing, un poquito de teoría que tiene que ver con la certificación STQB, es una certificación que se puede hacer para certificar como testers y bueno, eso tiene un poco de teoría que volcamos también en este curso. Vamos a ver cuáles son las características de un "test case", vamos a ver, vamos a hablar, digamos sobre el diseño de "test case" y vamos a ver algunos ejemplos. Bien, entonces en este vídeo “Principios del testing”.

Son siete principios, el principio uno dice que la prueba muestra la presencia de defectos. La prueba puede mostrar que los defectos están presentes, pero no puede probar que no hay defectos, nosotros no podemos hacer una prueba y después de probar decir: “bueno, no hay errores”. No podemos asegurar que hay errores. Podemos asegurar que hay errores si los encontramos y podemos decir también, que no encontramos nosotros más errores, por lo menos, en estos, con estas pruebas que hicimos.

Pero la prueba reduce la probabilidad de los defectos no descubiertos restantes en el software, si, o sea que reduce la posibilidad de que haya más errores, pero incluso si no se encuentran defectos, no es una prueba de corrección, ¿está bien?, entonces, la prueba no demuestra que no hay errores, demuestra los errores que hay y que no se encontraron más pero no significa que no haya más.

Principio número dos, que la prueba exhaustiva es imposible, o sea, probar todo, todas las combinaciones de entradas y de precondiciones no es posible excepción de casos triviales, o sea, ofrece un sistema muy sencillo únicamente, si no, es imposible probar todas las combinaciones de casos que pueda haber, todos los escenarios posibles, es muy difícil que entonces en vez de la prueba exhaustiva, se usa el riesgo, yo les comentaba en un vídeo anterior, se usa el riesgo y prioridades para enfocar los esfuerzos de prueba. Entonces, uno prueba lo que es más peligroso y prueba lo que es más probable que pase. Está bien, pero se prueba lo más importante, sí, pero no se prueba exhaustivamente porque no se puede.

Bien, principio tres, la prueba temprana. Las actividades de prueba deben comenzar tan pronto como sea posible en el ciclo de vida de desarrollo del software o del sistema, y deben enfocarse en los objetivos definidos, ya que cuanto antes comencemos a mirar, esto también lo hemos dicho, cuanto antes comencemos a hacer las pruebas, muchos menos costos va a tener, después el error si yo encuentro un error en forma temprana es más barato, si lo encuentro cuando ya está en producción es un error caro.

Bien, después el tema del agrupamiento de defectos. Se dice que un número pequeño de módulos, contiene la mayoría de los defectos y ¿esto es porque es así?, bueno, porque hay un número pequeño de módulos que son los que tienen las reglas de negocio más

importantes, lo que hace que sea más complejo. La parte más compleja siempre está en un número pequeño de módulos. No es complejo todo el sistema, siempre ahí como un núcleo de cosas más complicadas y ahí donde suelen estar los defectos pero bueno, esto quiere decir que probando en ese pequeño módulo, estos pequeños módulos, podemos encontrar muchos de los defectos o los más importantes. Bien, principio cinco, lo llamamos la paradoja del pesticida, ¿si?, significa que, ¿vieron como el tema del pesticida, que uno tira pesticida, por ejemplo, en plantas para matar los insectos, pero los de insectos terminan acostumbrándose al pesticida? y ya llega un momento que no le hace efecto ese pesticida, bueno, esto es algo parecido. En las mismas pruebas se repiten una y otra vez, es decir, nosotros siempre probamos lo mismo, eventualmente en el mismo conjunto de caso de prueba no encontrará más cualquier nuevo, bah, o sea, a ver, si siempre estamos probando lo mismo va a ser difícil encontrar cosas nuevas. No decimos, no digo que es imposible, sesiguen encontrando pero es más fácil encontrar cosas nuevas si probamos cosas distintas, encontrar errores nuevos pero haciendo otras pruebas, si probamos siempre lo mismo y bueno, es difícil que encontremos errores nuevos.

Bien, y los últimos dos principios que uno dice el principio sies, que era pruebas dependiente del contexto, la prueba se hace diferentemente en diversos contextos desde ellos. O sea, si yo tengo un contexto de seguridad, por ejemplo, en un banco, que es muy importante la seguridad, bueno, voy a probar de una manera, si no es tan importante la seguridad, voy a probar de otra manera, ¿está bien?, entonces, el contexto hace también a la prueba, a cómo basar la prueba. Y el principio siete habla de la falacia de la ausencia de errores. Dice que encontrar y arreglar defectos no ayuda si el sistema construido es inutilizable y no cumple las necesidades y expectativas de los usuarios, o sea, a ver, nosotros podemos quitarle todos los defectos a un sistema, pero si, por quitarle los defectos hacemos que este sistema sea inutilizable y no haga lo que tiene que hacer, bueno, eso no sirve, no sirve, ¿está bien?

Bueno, bien, entonces estos fueron los siete principios del testing y ahora vamos a continuar entonces con los otros temas a ver en esta unidad.