Hay un tipo de prueba que llamamos “prueba de componente”, también se la llama “prueba unitaria”, se realiza sobre una unidad de código claramente definida.
Yo les hablaba de unos modulitos, como unos módulos de programación, son como pequeños módulos, pequeños componentes, que son programitas chiquititos que hacen una única cosa. Entonces, cuando se tiene un problema complejo con un desarrollo de software, lo que se hace es atomizarlo, dividirlo en todos problemas chiquititos, esos son pequeños componentes. Entonces, lo que hace la prueba de componentes, justamente, es probar los componentes, probar es ese pequeño, esa pequeña porción de código de programación, ¿si?. Entonces prueba la lógica interna del componente, por ejemplo, una función, una función, no sé, elevar a una potencia. Entonces, se prueba el manejo de errores de acciones que, se prueba que se cumpla, que ese componente cumpla con la especificación. Entonces, finalmente, lo realiza el área que construyó el módulo. Esta prueba no es una prueba que hagan los testers, sino que es una prueba que hace normalmente el desarrollador. Entonces, el desarrollador, cuando codifica, cuando escriben su código, escribe también una prueba, que lo que hace es, justamente, invocar a su propio componente, pasándole unos valores y validar el resultado sea correcto, porque, a ver, un módulo de programación, un pequeño componente, va a tener una entrada y una salida, entonces, va a tener un ingreso de datos, nosotros le vamos a dar datos de ingreso y vamos a poder evaluar el resultado. Esto como una fórmula, ¿no?, yo alimento con ciertos datos y en función de la alimentación que yo le doy, puede evaluar la salida, ¿si? Ese componente es una caja negra para nosotros, no sabemos qué hace adentro o tenemos poca idea de lo que hace adentro, pero no tenemos forma de ver el código, ¿no?, nosotros como testers, el desarrollador sí lo hace. Pero, la idea es, si vamos a configurar una prueba de componentes, el desarrollador sabe qué valores le puede pasar a ese componente y en función de esos valores, qué pasa, qué resultado va a devolver ese componente.
Entonces, escribe, además de escribir el componente, se escribe una pequeña prueba del componente, donde los llama, lo invoca pasándole valores y verifica que el valor recibido sea el correcto. Entonces, ahí se asegura que ese componente funcione. Por eso es una prueba que hace el desarrollador, a veces lo hace el desarrollador con un tester, pero en general lo hace el desarrollador solo, se basa en el diseño detallado, comienza una vez codificado y compilado el módulo, y revisado el módulo, y los llamamos que son “pruebas de caja blanca”, o sea, para nosotros un componente, para nosotros como testers, un componente sería de caja negra, pero para un desarrollador es de caja blanca, porque el desarrollador es el que acaba de escribir el código, sabe lo que hace el sistema, ¿está bien? Entonces, lo prueba de una manera, digamos, conociendo el código, conociendo por dónde puede ir el flujo.
Entonces la prueba de componente, la prueba unitaria, es atómica, es decir, se prueba la mínima funcionalidad posible. Los testers en condiciones “if/else” deben separarse en dos casos, o sea, si tenemos “if/else”, si hay una condición, entonces hace esto y si no hace lo otro, bueno, tenemos que probarlo dos veces con la condición para que vayan por un lado y con la condición para que vaya para otro lado, o sea, si dentro del código del componente, hay una bifurcación, bueno, tengo que probar con valores que nos lleven para un lado y para el otro lado de la lógica y este es debe ser independiente, o sea, no depende de otro componente, se prueba este componente solo, e inocuo, porque no tiene que forzar ni romper nada y es un test que tiene que probar el funcionamiento, no tiene que romper la base de datos.
Entonces, la prueba componente la hace el desarrollador de software y la prueba de integración, la hace el equipo de pruebas. La prueba integración, ¿que sería?, es probar que los componentes entre sí funcionen, entre sí, se comuniquen entre sí, que entre sí, cuando

un componente llama a otro, cuando un ocmponente necesita datos de otros, que eso funcione, eso sí lo hace el testing, pero la prueba componente no la hace testing.
El test debe ser repetible, o reutilizable, o sea, hay que poder probar cuantas veces sea, porque el desarrollador prueba, encuentra que hay un error, corrige, prueba, ¿si?, prueba varias veces; automatizable, porque se puede poner dentro del mismo código, que bueno, una vez que se termina de ejecutarse, que se realice sólo la prueba y rápido, porque bueno tiene que ser algo que demore mucho, por supuesto.
Entonces, vamos a dar un ejemplo, vamos elevar un número al cuadrado. Vamos a tener el componente, “elevar un número al cuadrado”, entonces, si yo tengo un componente que hace esto, justamente, ¿qué le paso como parámetro de entrada?, un número, para que lo eleve al cuadrado y se espera que devuelva el número elevado al cuadrado. O sea, yo le paso tres, espero que me devuelva un nueve, ¿se entiende?. Y por ejemplo, también, podría ser otro componente, que sería sumar dos números, ¿si?. O sea, vamos a sumar dos números ¿qué le paso como parámetro?, dos números de entrada, distintos, el tres más el cinco y como devuelva como resultado la suma de los dos números, ¿si?, ¿está bien?.
Eso, esa es la idea de este, de lo que es prueba del componente. Que les digo, dentro de toda la escala del programa que hay, es la única prueba que nosotros no hacemos, que nosotros no hacemos, pero bueno, se los tengo que comentar.
Muy bien ahora sí, vamos a ir a ver las pruebas que nosotros sí hacemos.