V3
Éste gráfico nos está mostrando una idea de prueba de integración.
La prueba de integración es una prueba que realiza el área de Testing, el área de QA y que tiene que ver con probar cómo los componentes se relacionan entre sí, para generar una funcionalidad en el sistema. Entonces, el objeto es evidenciar defectos en las interfaces e interacciones entre componentes o sistemas integrados y son de importancia crítica.
Entonces, ahí ven que hay un círculo verde grande, el círculo verde grande dice “sistema” y después hay otros tres círculos verdes chiquitos, que son otros sistemas, entonces, nosotros tenemos que probar que este sistema, trabaje coordinadamente con otros sistemas, eso es una prueba de integración de los sistemas. Hay distintos sistemas, y que esos sistemas trabajen bien o sea, si yo les mando, los tengo que invocar, le tengo que mandar valores, tengo que recibir valores, que éste trabajen de forma coordinada, que trabajen correctamente, eso es una prueba de integración. Y otra prueba de integración es que dentro de un sistema están estos estos componentes que están en color azul, esos círculos azules, bueno, éstos son componentes. Entonces, esos componentes, primero hay que probar con pruebas unitarias, ya lo sabemos, pero además, hay que hacer una prueba de integración de estos componentes, que estos componentes trabajen entre sí en forma integrada, ¿si? Así como probamos que un sistema se integre con otro sistema, también probamos que los componentes se integren entre sí. Un componente quizás se tiene que integrar con tres o cuatro otros componentes, o con un montón más que no, pero con los componentes con los cuales tiene que trabajar, con los cuales se relaciona, bueno, que trabaje bien. Esa es la idea de la prueba de integración.
Entonces, la prueba de integración, prueba las integración de interfaces con funcionalidad, conecta a las partes más complejas y minimiza la necesidad de programas auxiliares.
Después, podemos decir que las pruebas de integraciones están orientadas a verificar que las partes de un sistema que funcionan bien aisladamente, lo hacen también en conjunto, esto es lo que venimos diciendo, ¿no?, sabemos que el componente funciona bien, bueno, a ver, en conjunto, ¿funciona bien? Entonces nosotros vamos a tener, imaginen un gráfico de componentes, que es esto que estamos viendo ahora, donde vemos qué componentes se relacionan con cuál otro, éste con éste, con éste, de manera jerárquica, uno llama al otro, el otro llama al otro. Entonces, las pruebas de integración que nosotros podemos hacer pueden ser incrementales, es decir, “bottom up” o “top down”, o sea de arriba a abajo, de abajo hacia arriba, es decir, arrancamos probando, este, acá vamos a ver la definición,¿no?, y las incrementales las tenemos acá. Entonces, los módulos que componen el software, se van acoplando progresivamente en conjuntos. Es decir, pruebo dos módulos, trabajan bien, bueno, le acoplo un tercero, ¿trabajan bien?, bueno, le acoplo un cuarto, luego de cada acoplamiento, se prueba la correcta interacción de los módulos y una vez verificado que el conjunto funciona, según lo previsto, se le suma un nuevo módulo y se vuelven a realizar pruebas, esto es lo que venimos diciendo. Tomamos, de éstos componentes tomamos un par y, ¿qué hacemos?, vemos si trabajan bien, lo probamos, bueno ,¿ok?, le acoplamos otro, lo probamos, esa es la idea.
Bien, el proceso se repite hasta completar la aplicación y cada conjunto parcial o total, debe verificar los requerimientos funcionales de rendimiento y de seguridad definidos. Y después tenemos unas, bueno, otras pruebas de integración incrementales, que tenemos la “bottom up”, que va de abajo hacia arriba, que prueba primero los componentes de bajo nivel y luego se avanza hacia los de mayor nivel, entonces, es decir, arranca por los componentes más chiquitos y después va subiendo, subiendo, subiendo al final, o sea, la síntesis, ¿no? como

sería, la mayor, pero, se puede realizar “stubs”, se pueden utilizar “stubs”, que son como emuladores para simular un componente que todavía no está disponible. Nosotros imagínense que tenemos que probar algo que tiene cinco componentes, pero no están los cinco componentes desarrollados. Hay tres componentes desarrollados y los que no. Bueno, esos dos, en realidad, podemos hacer un emulador. Imaginen que hay un componente que es una impresión, una impresión de un ticket, que llama a un servicio web y que entregue un número de comprobante electrónico y que asegura hace un montón de cosas o que hace una impresión especial en una impresora fiscal, lo que sea, bueno, eso es un módulo complejo, que tiene mucho código, pero nosotros podríamos decir, “bueno, vamos a emularlo” le vamos a llamar a esto y que nos devuelva siempre “ok”, que nos devuelva siempre un número, que nos devuelva siempre el número de ticket realizado, y después lo desarrollamos, ahora sabemos que si lo enfocamos, nos da una respuesta, nos devuelve un número, aunque sea ficticio que lo acabamos de crear, es un número secuencial, no importa, pero ese es un “stub”, ese es un simulador, no tenemos el código desarrollado, pero hacemos algo medio hueco que lo llamamos y nos devuelvo algo, y listo, a efectos de la prueba de los otros componentes, sirve. Nos sirve para probar ese componente, por supuesto, porque no está desarrollado, pero sirve para acompañar la prueba de otros componentes.
Y “Top down” lo mismo, se puede utilizar “Drivers” para simular datos de entrada, en caso de no tenerlos al momento de las pruebas, o sea si nosotros necesitamos hacer una prueba “Top down”, estamos arrancando por la funcionalidad más alta. Imaginen que, tenemos una pantalla de carga de datos y bueno, ¿y cómo hacemos para probar?, bueno, se puede hacer, utilizar “Drivers”. “Drivers” son unos programas que pueden ayudarnos a automatizar las pruebas, entonces, podemos meter datos dentro de ciertos campos para que eso, digamos, haga que la prueba se realice.
Y, por último, tenemos las “pruebas integración no incrementales” como ésta que es más “Big bang”. ¿Que hace “Big bang”? Se acoplan todos los módulos de una sola vez, se reduce la cantidad de pruebas, es decir, a ver, tenemos todos los módulos, los acoplamos todos juntos, no estamos acoplándolos en forma gradual, como veníamos haciendo, que teníamos dos módulos, probamos, si está bien, incorporamos el tercer módulo, no, todos juntos, todos a la vez. Se reduce la cantidad de pruebas, se dificulta la detección de posibles errores, porque acoplás todo y da un error y, ¿dónde está el error?, ¿en cuál módulo está el error? Requiere que todos los módulos del producto hayan sido desarrollados y probados en forma unitaria, ¿si?.
Entonces, decimos, si el proyecto es de corto plazo, hay poco tiempo para aplicación de la prueba y realizar las correcciones, bueno, se puede usar un “Big bang”, pero normalmente no es recomendable integrar todos juntos, porque si hay algún problema, es muy difícil de rastrear.
Vamos a hacer un ejemplo, tenemos componente “uno”, “elevar un número al cuadrado”, componente “dos”, “sumar dos números”,¿si?, ¿se acuerdan que teníamos estos componente? Bueno, vamos a testear, vamos a llamar al componente “dos”, que es sumar dos números, con los parámetros: “cinco al cuadrado” y “ocho”, entonces, esto te va a dar un resultado, ¿si?, va a dar un resultado. ¿Cuál sería el resultado de esto, no? Sería 33, no 32, acá tienen algo para decir, “encontré un bug”, ¿okey? Sería 33.
Entonces, “cinco al cuadrado”, ¿si?, lo que va a hacer, si nosotros pasamos como parámetros “cinco al cuadrado”, lo que va a hacer, va a ser llamar al componente uno, que es elevar un número al cuadrado, o sea, que al llamar al componente dos y pasarle parámetros,

en el pase de parámetros, estoy llamando al componente uno. Entonces, ahí estoy viendo cómo trabajan integrados el componente dos y el componente uno.
Bueno, hasta acá entonces lo que es prueba de integración, ahora vamos a ir a trabajar sobre pruebas de sistema.