Bien, entonces, vamos a hablar un poquito de “calidad”.
¿Qué es “calidad”? Nosotros trabajamos asegurando la calidad de un software, por eso es importante entender de qué estamos hablando, cuando hablamos de calidad. Entonces, ¿qué es para nosotros un software de calidad? Bueno, en principio, todo software tiene requisitos, entonces, si cumple con los requisitos, podemos decir que es de calidad, si no tiene fallas, podemos decir que es de calidad, digamos, son las cosas que uno, digamos, va a, a las cuales va a prestar atención, ¿no?. Pero bueno, si vamos un poquito más, más allá, lo que podemos ver es que hay como tres tipos de calidad, ¿no? Una es la calidad planificada, la dirección del proyecto planifica una calidad dice “bueno, esto lo que tenemos que lograr”. Pero, si bien tenemos la calidad planificada, también tenemos la calidad realizada, que no necesariamente es la misma que la planificada y, por otro lado, tenemos la calidad necesaria, la que se necesitaba para el proyecto, que no necesariamente es la planificada, ni tampoco la que se desarrolló.
Entonces, la conjunción de estos tres círculos, o sea, si uno estuviera superpuesto con el otro en forma perfecta, que no quede nada afuera, sería ideal, pero bueno, no es tampoco, digamos, que pase muy frecuentemente, entonces, esta intersección de los tres círculos es la “calidad óptima” y después bueno, tenemos, fíjense en las distintas áreas, que quedan, digamos, graficadas, con esto, tenemos distintos tipos de calidades, por ejemplo, hay una calidad realizada que es ineficiente, ¿si?, porque no era necesaria, se realizó una calidad que no se necesitaba.
Hay calidades que se planificaron y se realizaron correctamente, tal y como se planificó, pero son inútiles, es una calidad inútil, no sirve para nada. Hay una calidad que, esto es lo más grave, necesaria-no lograda y eso provoca insatisfacción. Y también puede pasar que haya una calidad realizada, que es necesaria, pero que es casual, porque, digamos, no se planificó, se realizó por casualidad. Para esto, entonces, acá en este gráfico tienen los distintos tipos de calidades sobre las que podemos, que nos podemos encontrar en un proyecto. Pero ya un poquito más formal y yendo a algo más, a una definición nosotros, en aseguramiento de la calidad, tenemos una definición que es la de QA, es el conjunto planificado y sistemático de todas las actividades necesarias para brindar una adecuada confianza que un producto o componente cumple con los requerimientos establecidos. Y, por otro lado, está el QC, lo que es el testing, la prueba del software, que es una de las actividades involucradas en el aseguramiento de la calidad.
Entonces, el texting está dentro de lo que nosotros llamamos “aseguramiento de la calidad”. Y “aseguramiento de la calidad” son un conjunto de actividades, una de las cuales es el testing, ¿se entiende?.
Entonces, vamos a ver la calidad según la norma ISO, esta norma 9.126, que dice que los factores que hacen a la calidad del software son: la funcionalidad, es decir, el software es completo, hace todo lo que tiene que hacer, o sea, si tiene que tener 10 funciones, hace las 10 funciones.
La fiabilidad: es correcto, ¿si?, no solo es completo, sino que las cosas que hace las hace bien; la usabilidad: es fácil de usar, o sea, si hace las cosas bien, pero después para usarlo es inentendible, bueno, no es de calidad. Entonces, la usabilidad es importante; la eficiencia: que lo haga, que, digamos, logre cumplir los objetivos de la mejor forma posible. Ustedes saben la diferencia entre eficacia y eficiencia. Eficacia es que logre el resultado y eficiencia, que lo logre de la mejor forma posible. Entonces, nosotros lo que queremos para que sea de calidad, es que logre de la mejor forma posible y si nosotros hacemos, bueno, una

calculadora, porque estamos con el tema de la calculadora, ¿no? y queremos hacer la suma de un millón más uno, ¿si?, imagínense que nosotros hacemos una calculadora que sume de a uno, uno más uno, más uno, más uno, más uno, un millón de veces, un millón una veces, para ver, para llegar al resultado, por ejemplo, la calculadora: ¿Sería eficaz?, sí. ¿Sería eficiente? No. Digamos, hay otra forma de hacerlo que no tenga que sumar, sumar, sumar y sumar.
Bien, que sea mantenible: que sea de fácil modificación, es decir, que si tengo que hacer una modificación sobre el software, sea posible hacerlo y que sea portable, adaptable a un nuevo entorno, quesea, digamos, que no sea tan restringido a un único sistema operativo, por ejemplo, a una única computadora. Lo que pasa es que, bueno, esto ya es más característica de diseño, ¿no?, porque a veces se diseñan de una manera que tiene que ser así y es únicamente para ese entorno.
Bien y, otra cosa que tenemos que conversar es: ¿Qué es asegurar la calidad contra controlar la calidad? O sea, nosotros decimos que queremos asegurar la calidad, ¿si?, pero bueno hay diferencias entre asegurar y controlar.
La calidad, en principio por definición, digamos, sepamos esto, la calidad no puede inyectarse al final del proyecto, o sea, nosotros no podemos, como pasaba antiguamente, recibir un software ya desarrollado y agregarle calidad ¿no?, la calidad no se puede agregar al final. Eso por un lado. Y, por otro lado, tenemos que saber que detectar errores en forma temprana, es decir, cuanto antes detectemos los errores, eso nos va a ahorrar esfuerzos, tiempo y recursos, ¿si?. Entonces, ¿qué significa esto?, que hay que empezar a testear lo antes posible. Nosotros nos podemos empezar a testear con el producto ya desarrollado, sólo empezamos a testear lo antes que se pueda, ¿si?, Incluso leyendo la documentación, la especificación funcional, leyendo ya arrancamos eligiendo lo que se quiere hacer y ahí empezamos a testear ya, a ver, si lo que se quiere hacer está bien, o tiene una falla. Entonces, hablábamos de QA, aseguramiento de la calidad, que está orientado al proceso. Es un procedimiento preventivo o el control de la calidad que está orientado al producto. Se explica uno que esté a un producto particular y es correctivo, ¿por qué?, porque buscar los errores y los pasos de desarrollo para que lo corrija no es que nosotros corrijamos algo desde testing nosotros no hacemos correcciones detectamos errores y lo pasamos para que ocurrirá otra.

Bien, entonces por qué necesario el testing, yo les quería comentar rápidamente algunos ejemplos, hay muchos que, algunos como más graves, pero ¿por qué es necesario el testing? Y bueno, en principio, digamos, sacar producto con errores, da mala imagen a nuestro equipo, hacen, que aprobo un re-trabajo y que sería innecesario, hace que la iimplantación de un nuevo desarrollo, o versiones de una misma aplicación sea difícil, sea tortuosa, y tiene un alto costo en correcciones, por esto que decíamos, ¿no?, si yo tengo que corregir, ya estoy haciendo un retrabajo que tiene un costo, costo de recursos humanos, costo en equipamiento, en licencias, muchos costos y esto acá, digamos, puse algunos ejemplos, en 1962 se quiso hacer la exploración de Venus y un programador dejó de poner una rayita, un guión en una instrucción, y la computadora no pudo leer correctamente uno de los programas y eso ocasionó que se derrumbe, que se caiga, perdió dirección, perdió dirección. Perdió dirección y cayó, y podía haber caído en cualquier lado, podría haber sido una catástrofe. También vieron que ahí hay programas de bolsa que compran y venden en forma automática, que están conectados con servicios de agentes de bolsa y se le pone una

lógica que cuando sube a una acción o cuando ve una tendencia alcista, compra, en cuanto ve que una tendencia en baja vende, y bueno, nada, pasó en un algoritmo que estaba mal programado que trabajó exactamente al revés, ¿si?, compró caro, vendió barato, y generó pérdidas por uno muchos millones de dólares.
También, digamos, en algún momento pasó que Rusia recibió en la guerra fría, una alerta de ataque de misiles y los estaban atacando, de hecho ven en la pantalla misiles que vienen hacia Rusia, pero bueno, no eran tantos misiles, eran pocos, la persona que estaba a cargo dijo “no puede ser, no puede ser, no puede ser”, porque, digamos que es obvio que si van a atacar con misiles a Rusia, no van a atacar por uno, dos o cinco misiles, van a atacar con mucho más. Entonces, les llamó la atención esto y no hizo nada, no hizo nada arriesgado, digamos, su seguridad, ¿no?, pero no hizo nada, no tomó ninguna acción, no contraatacó y la cuestión es había sido de que en realidad el radar, que estaba controlando si venían misiles o no, bueno, había sido engañado por los rayos del sol.
Así que bueno, nada, algunos ejemplos como para que ustedes entiendan lo importante que puede ser el testing y lo grave que puede ser que fallen los sistemas. Imagínense lo grave que puede ser que falle un semáforo, puede ocasionar una muerte. Entonces, hay muchos sistemas que son críticos, máquinas y sistemas de médicos, ¿no?, en hospitales y cada vez, cada vez, tenemos la informática en más lugares, entonces, nada, esto es muy crítico y bueno, tenemos que testear, testear es muy importante para asegurarnos que los software estén en buenas condiciones para ser operados.
Bueno, muy bien, vamos a continuar entonces con otros tema.