Y ahora vamos a hablar de las “Pruebas Complementarias”. Son pruebas que no tienen que ver con el funcionamiento del sistema, digamos, con la funcionalidad del sistema, con lo que el sistema tiene que hacer, con si hace bien los cálculos, por ejemplo, no tiene que ver con eso. Tiene que ver con otras cosas, otro tipo de pruebas. Por ejemplo, la prueba de carga o prueba de volumen, ¿qué hace?, verifica que el sistema soporta los volúmenes máximos definidos en la cuantificación del requerimiento. Cuando uno dice “¿qué es lo que tiene que hacer el sistema?”, da ciertos parámetros, dice “Bueno, a ver, este sistema necesita como requerimiento dos gigas de espacio en disco”, cierta capacidad procesamiento, cierta cantidad de transacciones, “soporta hasta 20 usuarios”, bueno, bien, entonces, estas definiciones nosotros lo que vamos a hacer es probar, llevarlas al máximo. Si soporta hasta 20 usuarios, bueno, vamos a probar con 20 usuarios concurrentes, o sea, 20 usuarios trabajando a la vez. Vamos a poner a 20 personas a trabajar todas juntas o a emularlo de alguna forma con el software que simule que haya 20 personas trabajando, pero vamos a probarlo en su, en su definición. Si está definido que el espacio almacenamiento es hasta dos gigas, vamos a ocupar esos dos gigas, vamos a trabajar, digamo,s a llevar el software al máximo, a su máxima capacidad, hasta donde dice el requerimiento, ¿está bien?.
Después tenemos otra prueba, que es la “Prueba de Estrés”. Esta prueba de estrés lo que hace es someter al sistema excediendo los límites de su capacidad de procesamiento, de almacenamiento, teniendo en cuenta situaciones no previstas originalmente, es decir, decíamos “Bueno, está pensado hasta 20 usuarios”, bueno, probemos con 500 usuarios, a ver qué pasa, o sea, vamos a tratar de estresarlo, vamos a llevarlo más allá del máximo para el que está pensado. No estuvo pensado para 500 usuarios, estuvo pensado para 20, vamos a llevarlo a 500 a ver qué pasa, ¿si? Vamos a exigirle de más a este software, vamos a estresarlo, ¿está bien? Vamos a lelvar a que ocupe no dos gigas de disco, si necesita, no sé, memoria, cierta cantidad de memoria, bueno, vamos a hacer que trabaje de manera tal que requiera más memoria y a ver qué pasa, a ver qué pasa, si quiere consumir más de memoria y no la tiene, ¿se pone muy lento?, ¿qué pasa?, ¿cómo reacciona el sistema? Esto lo lleva mucho más allá de los límites, para poder para poder decir “Bueno, a ver, está pensado para 20 usuarios, pero si trabajan hasta 100 usuarios, trabaja bien, por más que fue pensado para 20, más de 100 usuarios empieza a caer mucho la performance y más de 200 usuarios, el sistema se cuelga”. Pero esa definición, la podemos sacar con una prueba de estrés.
Después tenemos las “Pruebas de Rendimiento”, o performance, ¿si? Comprueban la rapidez con la que el sistema responde a una funcionalidad, qué tan rápido es el sistema, qué tan, qué tan veloz, qué tan veloz en distintas circunstancias.
Después, “Pruebas de Estabilidad”. Mide la rapidez de respuesta trabajando en forma continuada por largos períodos. Hay sistemas que trabajan muy bien en periodos cortos, pero si necesitan mucho tiempo, quizás empieza a consumir cada vez más memoria, más memoria, más memoria y llega un momento que se pone muy lento. Entonces, estas pruebas de estabilidad, lo que hacen es ver si el sistema es estable.
Después están las “Pruebas de Robustez”, que evalúan la reacción a datos de entrada erróneos, ¿qué pasa si le damos datos que no sirven?, por ejemplo, ahí subo una foto, en vez de una foto, le subo una base de datos de cinco gigas, bueno, ¿qué va a ser el sistema?, ¿cómo reacciona el sistema?, ¿cómo reacciona ante eso?, es la prueba de robustez.

“Prueba de Cumplimiento”, si cumple con las normas y los estándares, si cumple con las, puede haber ciertos estándares que haya que cumplir, por ejemplo, “Normas ISO” o alguna, alguna especificación muy particular que tenga algún, no sé, algún área médica o este, bueno, no sé, distintas cosas que puede ser que sea necesario contemplar en el software, y ver si cumple eso, ¿no?, reglamentaciones.
Bien, por ejemplo, en bancos, bancos que tienen algunas medidas de seguridad particulares, bueno, este tipo de cosas.
Y, también tenemos como prueba complementaria lo que es la “prueba de regresión”. ¿Qué hace la prueba regresión? Verifica que, luego de introducido un cambio en el código, la funcionalidad original final no haya sido alterada, esto ya lo conversamos en algún momento, ahí tienen una pequeña historieta, donde ven que hay un dibujito de un chico corriendo un bicho, parece una mosca, este lo atrapa lo saca afuera de la casa y, no sé si lo llegan a ver, que cuando saca a este insecto, entran un montón más, bueno, estas cosas son las que suelen pasar cuando uno, por corregir un “bug”, genera muchos más, entonces, por eso es necesario hacer las pruebas de regresión, para volver a probar la funcionalidad. Después de cada modificacion en el sistema, volver a probar toda la funcionalidad.

Bien, después tenemos, como prueba complementaria también, las pruebas “Alfa” y “Beta”, esto, ustedes saben lo que son las versiones “Betas”, han escuchado, ¿si? Se entrega una primera versión al usuario, lista para ser probada por ellos, o sea, el que prueba al que hace la prueba “Alfa” y la prueba “Beta”es el usuario. Normalmente está plagada de defectos, ¿si?, y es una forma económica de identificar estos defectos, ya que el trabajo lo hace otro, lo hace un usuario.
Entonces, la prueba “Alfa” la hace el usuario, en nuestras instalaciones. Nosotros, nosotros ponemos en un servidor una aplicación para que el usuario lo pruebe. Nuestra la prueba “Beta”, es la que hace el usuario en sus propias instalaciones. Nosotros le vamos a pasar, digamos, una versión, por ejemplo, de “Windows 25”, ¿si?, que lo pruebe en su máquina, que lo pruebe cada uno en su computadora.
Bien, después están las pruebas complementarias que son de usabilidad, es decir, que el el software sea comprensible y amigable, o sea, que sea fácil de usar. Un sistema que es inentendible, funciona todo bien, pero es inentendible, la verdad que no tiene mucho utilidad. Entonces, es importante que el sistema sea usable, entonces, eso también se prueba.
Y también está la “Prueba de Smoke”, ¿se acuerdan que hicimos un “Smoke Test” nosotros? Es una revisión rápida del producto para comprobar que funciona y no tiene defectos que interrumpan su operación básica. Se hace una prueba básica del producto. Ahí está, yo les comentaba en aquel momento de la prueba de humo, como tenía que ver con los caños de plomería, y se inyectaba humo para ver la pinchadura, eso es la imagen que estamos viendo. Bien.
Y, bueno este “Smoke Test” que se realiza previa a la recepción del software, por el equipo de pruebas o por el usuario final. Por otra parte, tenemos también pruebas de compatibilidad, cómo reacciona el producto en diferentes entornos y plataformas, es decir, ¿va a funcionar bien para un teléfono esta aplicación?, ¿funciona correctamente?, ¿se ve correctamente en una tablet, en una computadora, en todo tipo de pantallas?, ¿ se ven bien en distintos navegadores?, bueno, pruebas de compatibilidad.

Otro tipo de pruebas son las pruebas de seguridad, ¿si? Esta prueba determina el nivel de seguridad de las aplicaciones y sistemas en términos de confidencialidad, integridad y disponibilidad. Hay algo que se llama “Owasp”, que se publica todos los años que son, digamos, las medidas de seguridad más comunes para defectos más comunes que existen, que se encuentran en los software, y es importante para alguien que está en el ambiente la seguridad, que esté interesado en la seguridad, es importante que lea esto, “Owasp”, están publicados, lo pueden encontrar en internet, ofrece herramientas, documentos, foros, capítulos gratuitos y abierto a cualquiera que esté interesado en mejorar la seguridad en las aplicaciones.

Y por otra parte tenemos el “Testing Exploratorio” que es un enfoque en el que simultáneamente se aprende sobre la aplicación, es decir, a medida que estamos aprendiendo sobre la aplicación y se diseñan casos de prueba y se ejecutan esos casos de prueba, todo a medida que vamos probando, ¿no?, o sea, lo vamos haciendo todo en el momento, ese es el “Testing Exploratorio”. No se prueba por probar, o sea, tiene un objetivo y un límite de tiempo para realizar esas pruebas, hay una misión que define qué se probará del producto, yo quiero probar que su factura correctamente. Los tipos de incidentes que se buscan, vamos a buscar todos inicidentes que sean bloqueantes, incidentes que no avanzan y los riesgos involucrados. Y la sesión indica el itinerario, cada sesión se le estima un tiempo de ejecución y percibe un cierto objetivo, que puedes poner el propio Tester, del equipo de Testing o el Test Manager.

Entonces, tiene un objetivo general y no se indican los pasos a seguir para conseguirlo, no hay un paso a paso, hay un objetivo genérico.

Existe documentación, que pueden ser los casos que se fueron probando o sólo los defectos encontrados, no es que hay mucha documentación, pero bueno, ese es un “Testing Explorer”, es un tipo de testing y es una prueba complementaria que se puede utilizar, digamos, está dentro de las pruebas complementarias. Enfoques es complementarios, además, técnicas de pruebas y Testers, se necesitan Testers de mente abierta, pensamiento crítico, observadores, creativos y curiosos, para detectar “bugs” más complejos y evaluar riesgos y esto es lo que lo que es necesario para hacer el Testing Exploratorio.

Bueno, con esto completamos la lista, ¿si?, de los que serían las pruebas complementarias más comunes y cerramos también, entonces, con esto, la unidad cinco.