Entonces ya vimos pruebas de caja negra, vamos ahora a repasar concepto de prueba de caja blanca, decimos que las pruebas de caja blanca son pruebas estructurales porque vemos la estructura del código, también son conocidas como class-box porque digamos como si fuera una caja transparente no, en caja blanca despues se puede ver internamente que lo que hace, se prueba lo que el software hace, se evalúa el resultado aprovechando el conocimiento interno del código para dirigir la prueba, aprovechamos lo que sabemos del código para poder dirigir la prueba hacia donde nosotros queremos, estas pruebas de caja blanca no garantizan el cumplimiento de las especificaciones funcionales, son pruebas complementarias a las funcionales, las funcionales son las de caja negra, estas son complementarios, en cada caso de prueba debe probar solo un camino de todos los posibles, por ejemplo si la función contiene una condición "if/else" si pasa esto entonces, si no sale otra cosa, se deben hacer dos casos de prueba, cuando hay una bifurcación en el código se deben hacer dos casos de pruebas, si aparece un "or" un "and" es decir, esto y además esto otro, si se cumple esta condición o esta otra condición se deben generar dos casos prueba para la misma condición y así, el tema es poder cubrir todo toda la lógica por donde va el código, que se acuerdan que habíamos hecho un flujo no, entonces el código puede ir para un lado y para el otro bueno hay que tratar de cubrir todo el árbol, esto tiene una estructura de árbol, con su nodo principal y su rama distintos nodos, distintas ramas que salen de ahí, bueno la idea es poder cubrir todo, todo, todo, todo ese lugar el desarrollador al conocer la estructura interna puede dirigirse directamente a las partes del código más riesgosas generalmente realizado este este test por el desarrollador o con el desarrollador, entonces acá tenemos el ejemplo el flujo de lógica, ahí tenemos un programita que dice bueno este hace un input de A y B estamos leyendo la parte izquierda de la pantalla e input A y B, ingresa dos valores al sistema cada valor es desde 1 hasta 100, ingresamos los números válidos de 1 hasta 100, pregunta si el primer número el A es mayor a 10 entonces hace la sentencia 1 y si B es menor a 25 entonces hace la sentencia 2 y si no hace la sentencia 3 y si no se cumple el de arriba el mayor a 10 hace la sentencia 4, esto es lo que está haciendo este este código, entonces si vamos al gráfico, a la derecha, tenemos la representación de esto mismo, preguntas el rombito siempre está está determinando que hay una pregunta entonces el rombo es una pregunta, si A es mayor a 10, si, si eso va a pasar cuando, va a pasar el 90% de las veces y entonces vamos vamos para ese lado si si, si no vamos para el otro lado si aparece cual, eso va a parecer el 10% de las veces, entonces se dividen o se divide en el flujo, uno más grueso porque van muchas más opciones por ahí, otro más finito porque van menos opciones por ahí, aparece cuatro cuantos van a ejecutar, nada más que mil casos, si si nosotros ponemos todos los números no, estamos haciendo una estadística pero si nosotros ingresamos por ejemplo tenemos un número A que puede ir de 1 a 100, el número B que puede ir de 1 a 100 estos son 10 mil combinaciones, entonces esas 10 mil combinaciones hay 1000 que van a ir por un lado y 9000 por otra y después pregunta si ves menor a 25 entonces también hay este esa una pregunta, un 75% que va por un lado y un 25% que va para el otro, se multiplica por digamos para saber la cantidad de casos se multiplican por lo que vinieron de la condición anterior, por que era un 09%, entonces tenemos 2.250 casos por un lado y 6.750 casos por el otro, aca tenemos como iría el flujo no adentro del sistema, si nosotros ingresamos números digamos, cubrimos todo el espectro, sabemos que hay 6.750 veces que va a ir por un lugar 2.250 por otro y 1.000 veces que va a ir por otro esto es importante saber para saber dónde con más probabilidad va a ir el sistema, este este es el nivel más bajo del testeo por cobertura establece que cada sentencia del software testeado debería ser ejecutada por lo menos una vez, yo me quedaría tranquilo si sé que cada cosa que está escrita en el código por lo menos pasa por ahí una vez el sistema y una vez se ejecute, entonces qué pasa, decimos que aunque se realice esta cobertura total cubrimos todo el código, se puede perder muchos caminos posibles, como es esto, a veces no es sencillo realizar la cobertura total, por

ejemplo en los casos límite donde el software maneja excepciones de tipo baja memoria, disco lleno, disco corrupto pero fíjense esto no, hablando de cobertura total, nosotros tenemos acá un código a la izquierda donde nos dice bueno a ver, X es igual a 0 pregunta, le aisgno un valor X es igual a 0 ¿cuál es el valor de X? 0, me pregunta si A es mayor a 0 dice que X es igual a sí misma o sea 0 + 1 y pregunta despues si B es igual a 3 dice qué Y es igual a 0 y después imprime X e Y eso es lo que hace este código, bien ahora si yo quiero lograr la cobertura total lo que podría hacer es decirle bueno pongo un valor de A mayor a 0 y un valor de B igual a 3, si yo pongo un valor de A mayor a 0 pasa por la primera condición y ejecuta el X es igual a X+1 y si pongo un B igual a 3 pasa por la segunda condición y ejecuta el Y igual a cero entonces acá ejecuta todo lo cual me la da cobertura total, ya pasé por todas las líneas de código del sistema pero me deja camino fuera del alcance donde podría haber defectos, por ejemplo qué pasa si A es mayor a 0 y B no es igual a 3, si A no es mayor a 0 y B tampoco es igual a 3 y si A no es mayor a 0 y B si es igual a 3, en esos casos no estamos, no se cubre toda la cobertura pero ¿ y si hay un error con eso? Por ejemplo si nosotros seguimos este código y le ponemos como valores un A mayor a 0 sí, pero B le ponemos igual a 4 entonces hace la primer parte X igual a 0 pregunta ¿hay mayor a 0? Si X es igual X+1, o sea, X es igual a 1 ahora, después pregunta B es igual a 3, no entonces Y no es igual a 0, nunca llega nunca ejecuta el Y igual a 0 y después quiere imprimir, quiere imprimir X que X sabe que tiene valor uno y quiere imprimir Y al cual le desconoce el valor porque nunca llegó a asignarle valor a esta variable Y entonces va a dar un error cuando quiere imprimir, entonces digamos tener la cobertura total no nos garantiza que no haya errores, puede haber errores justamente por no haber pasado por algún lugar. Bien después podemos hablar acerca de otra técnica que la cobertura de decisiones se prueban los casos para que cada decisión sea evaluada por lo menos una vez, en vez de ocuparnos de que todas las sentencias sean ejecutadas ahora lo que me interesa es cada triangulito o cada decisión sea evaluada por lo menos una vez, el verdadero y una vez el falso, entonces esto este pero a cada salida de un “if” o un “WHILE” o de alguna cosa que tenga una condición, tampoco nos asegura la cobertura de todos los caminos pero si yo tengo este código que es el mismo que teníamos recién bastaría con dos casos de prueba para cubrir todas las decisiones, por ejemplo si yo pruebo con algo que sea verdadero y verdadero ahí estoy esté probando que entre por el primer if y que entre por el segundo if entonces en este caso estoy probando cada uno de sus if cuando sus condiciones trucos, condiciones verdaderas y después para poder probar que los dos if sean falsos con otro ejemplo, entonces le pasó datos como para que los dos den falso, entonces ya probé cada uno de los if por su valor verdadero y por su valor falso pero no perdemos, qué pasa si tenemos un true falso o un falso true, en realidad sabemos por lo que decíamos hace un rato que un true false es va a generar un error y eso no lo detectamos haciendo la cobertura de decisión, bien y después tenemos la cobertura de rama, en caminos, también llamada prueba del camino básico se prueban todos los caminos independientes, o sea, sus funciones, la cobertura de rama es deseable pero muy costosa, como que probamos todos los caminos pero es muy costoso, realizar la prueba con herramientas de automatización abarata los costos de hacer este tipo de pruebas porque éste estar haciendo manualmente, probar cada uno digamos de los caminos es muy muy complicado y estos son los esto que tenemos acá gráficado son los distintos flujos de control representados por grafos, donde tenemos con el if/else tenemos la selección múltiple, el while que es algo repetitivo una secuencia que es lo más simple y el repetir hasta son las distintas estructuras que podemos usar y que tienen condiciones salvo la secuencia que tienen condiciones que tenemos que evaluar que son las que nos van a agregar complejidad a nuestro código, bien por otro lado hay algo que se llama prueba de caja y se lo menciono porque quizás lo escuchan, combina elementos de caja negra y caja blanca, un mix, se conoce parte de la implementación, estructura interna y se generan condiciones y casos que nos genera una prueba de caja negra, el conocimiento es parcial, no total porque si fuera un conocimiento total sería caja blanca, o sea, que conocemos algo hice algo y conocemos los guías para poder probar, eso se llama caja gris y por último les voy a

contar acerca de los sistemas mutantes, como el esto, cuando nosotros trabajamos con caja blanca significa que tenemos acceso al código y dominio del código de programación, entonces tenemos a ver un un sistema que tenemos ahí a la izquierda, que tenemos que probar, bien entonces nosotros le damos un input a este sistema y el sistema da una salida que es un output, agregamos le decimos una entrada de dato y nos da una respuesta, ahora bien tenemos este sistema pero yo digo qué pasa si modificamos el sistema, hacemos un sistema paralelo, otro sistema, basado en este pero con alguna modificación, entonces nos basamos en el sistema principal y lo mutamos, hacemos un mutante de ese que es muy parecido al sistema original pero tiene un pequeño cambio, tiene un pequeño cambio, entonces qué pasa nosotros le vamos a dar el input el mismo input a los dos sistemas, al original y al mutante y vamos después a ver las salidas de los dos, si las salidas son distintas entonces está todo bien porque las salidas tienen que ser distintas porque si los sistemas son digamos hacen cosas distintas la salida tiene que ser distinta, ahora el problema lo tenemos si las salidas son iguales, si las salidas son iguales este que pasa ahí pasó algo raro, porque son iguales las salidas, porque estamos teniendo el mismo resultado entonces ahí hay evidentemente por un lugar que no está pasando el código, no no está yendo por ahí, entonces eso a nosotros nos sirve para detectar dónde puede haber conflictos, muy bien entonces hasta acá terminamos lo que es esta estas técnicas de caja blanca y de caja negra, espero que les haya gustado y ahora vamos a comenzar una nueva unidad, en la próxima unidad vamos a conversar y charlar y ver cómo se utilizan las bases de datos.