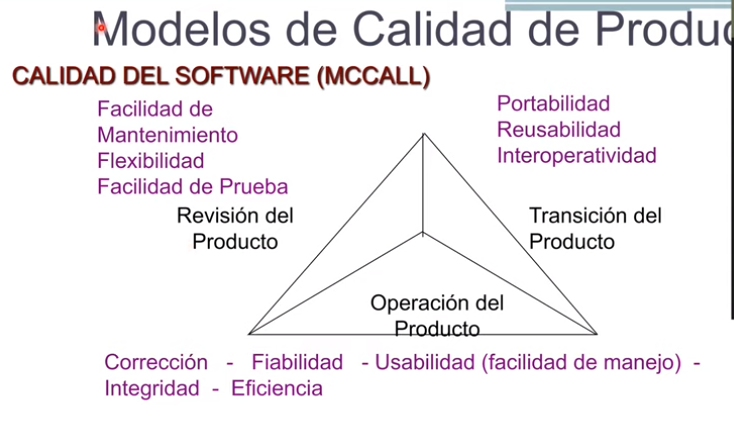
**INGENIERÍA DE SOFTWARE  
Tema: Calidad de producto.  
Calidad:** Todos los aspectos y características de un producto o servicio que se relacionan con la habilidad de alcanzar las necesidades manifiestas o implícitas.   
**Aseguramiento de calidad sobre el producto:** Se refiere a una comparación contra los requerimientos funcionales y no funcionales. El producto en sí tiene calidad siempre y cuando satisfaga las necesidades de los interesados. **Un SW de calidad satisface:** Las expectativas del cliente, del usuario, de la gerencia, del equipo de desarrollo y mantenimiento y de otros interesados. Y todas estas expectativas son todas distintas. Depende mucho de las expectativas y necesidades, la calidad es relativa a las personas, a su edad, circunstancias de trabajo, al tiempo. No solo está asociada a las características y capacidades del producto, sino también a toda la situación en la que el producto se desarrolla, se entrega, etc.  
Es una actividad más dentro del desarrollo de SW en la que se le asignan personas y recursos y todos deben hacer calidad en todas las etapas, no es una etapa puntual.   
La calidad tiene un costo, tiene que haber un balance entre los beneficios y los costos de la calidad. Si se produce un sistema de mala calidad, entonces nadie lo va a querer comprar. Por otro lado, si se dedica un gran tiempo y alto presupuesto al desarrollo del SW perfecto, entonces demorara tanto tiempo de esfuerzo y de desarrollo y costara tanto producirlo que estará fuera del mercado. Hay que buscar un punto medio en el cual el producto sea lo suficientemente bueno para cumplir con las necesidades del cliente y no sea tan perfeccionista como para que requiera mucho tiempo o costo para terminarlo.  
 **Calidad de producto:** La calidad de un producto no se puede sistematizar contra un modelo estándar, porque varían de producto a producto, cada producto tiene sus requerimientos, por eso, no existen certificaciones ni acreditaciones a la calidad del producto. Las evaluaciones de calidad se hacen contra los requerimientos de ese producto y la vas a poder hacer siempre y cuando exista una manera de formalizar cuales son esos requerimientos.  
 **MODELOS DE CALIDAD DEL PRODUCTO:  
  
ISO 25000:** Evalúa la calidad del SW. Es un conjunto de normas que tiene como objetivo la creación de un marco de trabajo para evaluar la calidad del SW.   
****  
**1. División de modelos de calidad:**   
**- Calidad en Uso:** Describe el modelo de calidad para el producto de SW y para su uso sobre que características de SW evaluar.  
**- Calidad de datos:** Define un modelo general para la calidad de los datos aplicable a aquellos datos que se encuentran almacenados de forma estructurada.  
  
**2. División de Mediciones de calidad:** Estas normas incluyen un modelo de referencia de la medición de la calidad de un producto, definiciones de medidas de calidad (interna, externa y en uso) y guías prácticas para su aplicación.  
  
**3. División de Requisitos de Calidad:** Este apartado incluye normas que proporcionan requisitos, recomendaciones y guías para llevar a cabo del proceso de la evaluación del producto de SW.

****  
Modelo de calidad que busca reducir la brecha entre usuarios y desarrolladores enfocándose en factores de calidad. PAG 72  
Organiza los factores en 3 ejes o puntos de vista desde los cuales el usuario puede contemplar la calidad del producto basándose en 11 factores de calidad y a su vez cada factor, se descompone a su vez en una serie de criterios que determinan su calidad, estos criterios son evaluados sobre un conjunto de métricas, para cada criterio, deben fijarse unos valores máximos y mínimos aceptables para cada criterio.  
Plantea una categorización de factores que afectan la calidad del SW:  
**Operación del producto:** Características de calidad del producto en cuanto a su uso. Se espera que todos los productos de SW satisfagan estas características, el problema es como lo medís.   
**- Corrección:** ¿El SW hace lo que yo quiero?  
**- Fiabilidad:** ¿Lo hace de forma exacta todo el tiempo?  
**- Eficiencia:** ¿Se ejecutará sobre mi HW lo mejor posible?  
**- Integridad:** ¿Es seguro?  
**- Facilidad de uso:** ¿Puedo ejecutarlo?  
 **Revisión del producto:** Características de calidad que nos interesan a nosotros internamente como desarrolladores. Cuanto me va a costar mantener ese producto, localizar defectos, evolucionarlo, etc.  
**- Facilidad de mantenimiento:** ¿Puedo arreglarlo?  
**- Facilidad de Prueba:** ¿Puedo probarlo?  
**- Flexibilidad:** ¿Puedo modificarlo?  
 **Transición del producto:** Características de calidad del producto en distintos ambientes y la conexión del producto con productos externos.  
**- Interoperabilidad:** ¿Podré comunicarlo con otros sistemas?  
**- Portabilidad:** ¿Podré usarlo en otra máquina?  
**- Reusabilidad:** ¿Podré reutilizar parte de mi SW?  
  
**Las métricas:** auto documentación, capacidad de expansión, compleción de las funciones, complejidad, concisión, consistencia, modularidad, operatividad, seguridad, simplicidad, tolerancia a errores y trazabilidad.   
**Ventajas:** Se focaliza en el producto final identificando atributos claves desde el punto de vista del usuario. Focaliza en medidas precisas de alto nivel.  
**Desventajas:** Las características son por lo general propiedades abstractas y no siempre existe una relación lineal entre los valores de las métricas y las características a estimar.  
  
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Técnicas y Herramientas:** **Para el Aseguramiento de calidad:** El aseguramiento de calidad es una actividad continua que se realiza acompañando a la construcción del producto. Lo que quiere decir que la calidad no es una etapa puntual.   
 **- Revisiones Técnicas:** Las revisiones técnicas son un proceso de Validación y Verificación estático cuyo principal objetivo es detectar defectos y corregirlos en etapas tempranas del desarrollo. La hacen colegas pares.  
**INFORMALES:  
Walktroughs:** Técnica de análisis estático en la que un diseñador o programador dirige miembros del equipo de desarrollo y otras partes interesadas a través de un producto de software y los participantes formulan preguntas y realizan comentarios acerca de posibles errores, violación de estándares de desarrollo y otros problemas.  
**Objetivos:** Mínima sobrecarga, un proceso rápido. Capacitar a los desarrolladores. Rápido retorno.  
**Ventajas:**- Se pueden probar versiones completas de productos y encontrar errores.  
- Aprendizaje colectivo  
**Desventajas:**- No se documentan  
- Dificil encontrar métricas  
- Se ahorra tiempo cuando los participantes tienen experiencia sino se pierde tiempo  
  
**FORMALES:**   
**Inspecciones:** Objetivo es detectar y remover todos los defectos eficiente y efectivamente.   
**Características**: Proceso Formal. Se utilizan checklists para ir realizan la inspección. Se toman métricas. Es llevada a cabo por una Junta de Revisión (4 a 6 personas) que se dedican a la búsqueda de fallas en los productos de trabajo y cuyo éxito dependerá en gran medida de la planificación y el control que se realice sobre las mismas.   
**Objetivos:** Descubrir errores. Verificar que el software alcanza sus requisitos representado de acuerdo a ciertos estándares. Conseguir un software desarrollado de manera uniforme.   
**Son:**   
- La forma más barata y efectiva de encontrar fallas.  
- Una forma de proveer métricas al proyecto.   
- Una buena forma de promover el trabajo en grupo.  
- Método para mejorar la calidad del producto.   
**No son**:   
- Encuentran soluciones a fallas   
- Obtener la aprobación del producto   
- Para evaluar el desempeño de las personas.   
 **Métricas:** Esfuerzo x inspección. Esfuerzo x defecto. Total de defectos encontrados. Densidad de defectos.  
  
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Para poner un poco en contexto:** La auditoría de configuración es una de las Actividades de la Gestión de Configuración de SW (junto con la identificación de items, el control de cambios y los informes de estado). La gestión de configuración de cambios es una disciplina de soporte protectora cuyo objetivo es mantener la integridad del producto. Si bien nunca podemos garantizar la calidad total del producto, hay un montón de cosas para que el producto tenga más calidad.  
**Configuración de SW:** Conjunto de ítems de configuración con su correspondiente versión en un momento determinado.  
**Ítem de Configuración (SCI):** Son todos los artefactos que forman parte del producto o del proyecto que pueden sufrir cambios o necesitan ser compartidos entre los miembros del equipo y sobre los cuales necesitamos conocer su estado y evolución. O sea, es cualquier cosa que puede guardarse en un file system o en una computadora. Ej: Cualquier artefacto. Cada ítem de configuración debe tener un nombre único para garantizar su integridad.  
**Versión:** Estado de un ítem de configuración en un instante o contexto dado.   
**Línea base:** Ítem o conjunto de ítems de configuración con su estado de configuración que han sido revisados y aprobados (se los considera estables) y que para cambiarlos deben pasar por un proceso formal de control de cambios.  
**De especificación:** No tienen código, tiene información de ingeniería del producto (requerimientos, diseño).   
**De producto:** Tienen código que ya es operativo que han pasado por un control de calidad definido previamente.

**- Auditorias de Configuración:** La auditoría de configuración tiene como objetivo asegurar que la línea base o actualización se ha alcanzado realmente y que el software y la documentación son internamente consistentes para entregarlos al cliente. La auditoría es una revisión independiente y objetiva sobre el producto, por lo cual, el auditor siempre es alguien externo.   
**Beneficios de la auditoria:**  
- Aumenta la protección contra cambios innecesarios   
- Mejora de la visibilidad del estado del proyecto y sus componentes   
- Aumenta la auto responsabilidad   
- Disminuye los costos por re-trabajos   
- Disminuye el tiempo de desarrollo   
- Aumenta la calidad   
- Suministra visibilidad y rastreabilidad del ciclo de vida del producto de software   
**Auditoría Física:** Hace verificación (¿El producto funciona correctamente?). Vela por la integridad del repositorio para poder satisfacer los requerimientos, o sea, que el repositorio esté, que esté alojado donde se dijo que iba a estar, que los Items de configuración respeten las nombras de nombrado y que estén guardados donde se debe.  
**Auditoría Funcional:** Hace validación (¿Es el producto correcto?). Verifica si el producto es el producto correcto, es decir, si los Items de configuración hacen lo que los requerimientos dicen que tiene que hacer. Verificando que la funcionalidad y rendimiento reales de cada ítem sean consistentes con la especificación del requerimiento. Se hace por muestreo, es decir, se escoge un requerimiento a lazar y se analizan sus ítems de configuración. La matriz de rastreabilidad de requerimientos es una herramienta muy útil para saber qué caso de prueba corresponde a cada requerimiento.  
Primero se hace la Auditoría Física y luego la Funcional, ya que si la auditoría física no sale bien la funcional no se hace.  
Las auditorias necesitan un plan de gestión de configuración en donde esté la estructura de repositorio, el esquema de nombrado de los ítems, la formación del comité para poder auditar eso. Si no hay línea base no hay que controlar.  
En función del proceso de control de la auditoria se obtiene un **informe de auditoría** que muestra todas las desviaciones que se detectaron y luego el equipo hace un **plan de acción** para informar cómo piensa resolver dichas desviaciones.  
 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Control de la Calidad:   
- Testing:** Proceso destructivo cuyo propósito es detectar defectos en el SW cuya presencia se asume.   
El testing es exitoso cuando se encuentran defectos en el código.  
Es una actividad del aseguramiento de calidad que controla la calidad, pero llega tarde porque el producto ya está hecho, en todo caso se visualiza que calidad tiene el producto pero no le otorga calidad.

**Defecto:** Algo es un defecto cuando se lo encuentra en una etapa posterior a la que se lo introduce.  
**Error:** Algo es un error cuando se lo encuentra y corrige en la misma etapa.

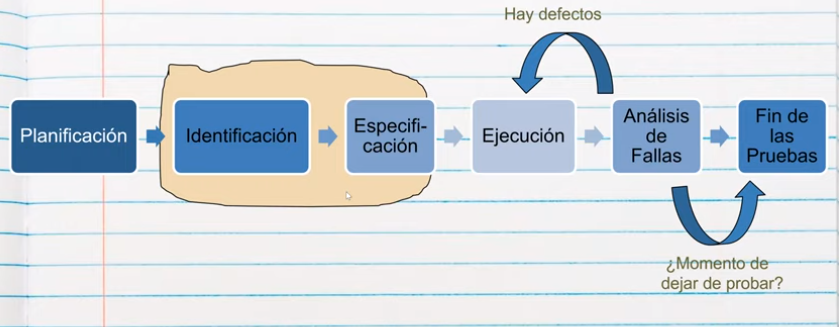
**Verificación:** Si el producto funciona correctamente. **Validación:** Si es el producto correcto o no.

**Clasificaciones de los Defectos:  
- Severidad:** Que tan grave es un defecto. Que tanto esfuerzo voy a necesitar para remediarlo. Evalúa que tanto daño le hace al SW ese defecto. La identifica el técnico de testing.   
1. Bloqueante  
2. Crítico  
3. Mayor  
4. Menor   
5. Cosmético  
**- Prioridad:** Que tan rápido se debe corregir un defecto. La define el cliente. Evalúa que tan importante es para el negocio la resolución del efecto.  
1. Urgencia  
2. Alta  
3. Media   
4. Baja

**Niveles de Prueba:   
- Pruebas de unidad:** El programador prueba cada componente (clase o método) del código de manera individual e independiente del resto de los componentes. No se deja registro.  
**- Pruebas de integración:** El equipo de desarrollo verifica si las partes del sistema que funcionan bien por separado lo hacen también conjuntamente. Se pueden automatizar mediante las prácticas de integración continuas.   
**- Pruebas de sistema o pruebas de versión:** Testing de los requerimientos RF y RNF de una versión o de un sistema como un todo. Un desarrollador nunca debe probar su propio código. Se hace en el WF de Prueba.  
**- Pruebas de aceptación de usuario:** El usuario hace la prueba de aceptación del producto para ver si el producto satisface sus necesidades. Se hace en el WF de despliegue.

**Ambientes para la construcción del SW:** Cuando instalamos el SW en otro lugar donde se desarrollo puede haber problemas, por lo cual, hay que probar la configuración y la escalamiento de la versión y que funcione en el ambiente del cliente.  
Hay que tener ambientes separados por que uno tiene que tener un ambiente limpio en el cual los desarrolladores no puedan inferir ni modificar nada. Además para asegurar la calidad del producto y no salir a producción con problemas.  
**- Desarrollo.  
- Prueba:** Se hacen las pruebas de sistemas. Están preparados con datos de prueba y las hacen la gente de testing. **- Pre-producción:** Se hace un ambiente muy similar al de producción para ejecutar el SW y detectar defectos. En este ambiente se deberían hacer las pruebas de aceptación para que cuando estén bien se pueda pasar a producción. No todas las empresas lo hacen pero los otros 3 ambientes deben estar garantizados. **- Producción.**

**Caso de prueba:** Condiciones o variables bajo las cuales un tester determinará si el software está funcionando correctamente o no. Se diseñan, por lo cual, una buena definición de los casos de prueba ayuda a reproducir los defectos.  
**Partes:** Objetivos, datos de prueba y resultados esperados.  
Los criterios de aceptación de las US se toman como base para definir qué condiciones de prueba voy a probar en cada escenario en particular.   
Para poder ejecutarlo tenemos que tener bien especificado:  
- Que queremos probar  
- El estado del sistema   
- Set de datos que se van a usar para hacer las pruebas con los resultados esperados.  
- El éxito del testing depende de que tan bien diseñados están los casos de prueba y que tan bien conformadas están las bases de datos para hacer pruebas.  
 **Derivación de los casos de prueba para hacer** **validación y verificación?** Desde los requerimientos para poder ver si el producto se comporta como el cliente espera que lo haga.  
- Si lo sacamos del código: Nunca podríamos hacer validación, solo podríamos hacer verificación.  
 **Ciclo de prueba:** Abarca la ejecución de la totalidad de los casos de prueba establecidos aplicados a una misma versión del sistema a probar. Cada vez que se encuentra un defecto se documenta en el **reporte de defectos.**  
**Estrategias:**  
**Testing con o sin Regresión:** El mejor testing es con regresión pero se usa sin regresión por el tiempo y el costo. Conforme vamos automatizando las pruebas es más factible poder hacer testing con regresión. **- Con Regresión:** Cuando todos los ciclos de prueba se van a ejecutar como si fueran el ciclo 0  **- Sin regresión:** Si ya estoy por el ciclo de prueba 1 y tomo el reporte de defectos del ciclo de prueba 0 y sólo testeo los errores del reporte estoy realizando una prueba sin regresión. Puede haber errores ocultos.

**- Actividades:  
  
Planificación:**   
- Se plantean los objetivos  
- Se hace el plan de pruebas (riesgos, objetivos, estrategia, recursos, criterios de aceptación, etc.)  
**Diseño (Identificación y especificación):** Se priorizan y diseñan los casos de prueba y el entorno de pruebas.  
**Ejecución:   
-** Creación de los datos de estado necesarios.  
- Automatizar  
- Ejecutar los casos de prueba  
- Registrar los resultados.  
**Evaluación y reporte:** Reporte de los resultados de las pruebas y evaluación de las actividades del testing.  
  
**Proceso de Pruebas:** Es importante tener casos de prueba diseñados y ejecutar pruebas sistemáticas basadas en un proceso (metodológicas) porque si no se encuentran defectos que no se pueden reproducir, es decir, no sabes qué pasos hiciste para lograr esos defectos (testing and hock).  
  
**Ciclo de vida del testing:** Se prueba siempre de forma inversa a la que se desarrolla. De lo particular a lo general, es decir, se empieza probando a nivel de prueba de unidad, después la integración, después el sistema y después la aceptación.  
Ya se está en condiciones de realizar planes de prueba cuando estoy en las actividades de planificación y diseño aunque se esté desarrollando la versión del producto para que cuando venga la etapa de ejecución ya tenga gran parte realizada.

**Tipos de prueba:** Apuntan a un aspecto o característica (RF o RNF) de un producto de SW que queremos probar. Están pensados para probar RF o RNF: **- Testing Funcional:** Las pruebas se basan en funciones y características (descripta en los documentos o entendidas por los testers) y su interoperabilidad con los sistemas específicos.  
**Tipos:  
Basado en requerimientos:  
 - Prueba de operación normal:** Esperar que el sistema sea exitoso donde los usuarios van a hacer todo lo que tienen que hacer y el sistema se va a comportar como se debe comportar.  
 **- Pruebas negativas:** Tomando la definición de un requerimiento esperamos que el sistema no permita hacer lo que estamos planteando en ese escenario.  
**Basado en procesos de negocio:** Evaluó un circuito completo de negocio haciendo énfasis en las reglas de negocio. Pruebas de integración porque integramos varios escenarios de distintos CU para conformar el proceso de negocio.  **- Testing No Funcional:** Es la prueba de “cómo” funciona el sistema. Tienen sentido si se tiene la configuración de HW en la que va a funcionar o algo similar.   
**Tipos:** Todos estos tipos de prueba tienen que ver con las características de calidad que suelen ser factor común en casi todos los productos de SW.   
**- Performance Testing:** Tiempo de respuesta y utilización de recursos de HW. Se hace para las características y requerimientos que tienen marcado un tiempo de respuesta.   
**- Pruebas de carga:** Se somete al sistema a todo el esfuerzo al que se comprometió. Se espera que el sistema las pase. Ej: si el sistema debe soportar 500 usuarios simultáneamente hay que probarlo.  
**- Pruebas de estrés:** Se somete al sistema a un esfuerzo mayor al que se comprometió para ver hasta dónde llega. Ej: si el sistema debe soportar 500 usuarios simultáneamente, probamos con 700. No se espera que el sistema las pase pero si esperamos ver en qué estado queda el sistema y cuanto nos cuesta que el sistema esté operativo nuevamente.  
**- Pruebas de usabilidad:** Es la experiencia de usuario. Están basadas en las definiciones del producto de SW que se hicieron en la etapa de diseño de interacción humano-máquina y lo que se intenta ver es que tan cómodo es para el usuario hacer su trabajo con el producto de SW. Se analiza la usabilidad.  
**- Pruebas de confiabilidad o fiabilidad:** Seguridad física y lógica. Problemas de acceso o accesos indebidos.  **- Pruebas de mantenimiento:** Cuanto nos cuestaa nosotrosintroducir un cambio en el producto de SW. **- Pruebas de portabilidad:** Como soporta el SW los cambios de ambiente, es decir, mover el producto de SW a espacios de HW diferentes (equipos, SO, Servidores, etc.).  
**-** **Pruebas de interfaces de usuario:** Hay una gran complejidad respecto a este tipo de pruebas porque hay un usuario en el control y miles de combinaciones   
**-** **Pruebas de volumen:** El propósito fundamental es probar el sistema desde el punto de vista de los datos, es decir, como se comporta el producto cuando hay mucho crecimiento en la base de datos, cuando la BD está llena, cuando la BD está vacía.   
**- Pruebas de configuración:** Ver si el SW se contrasta contra el manual de configuración, o sea, ver si siguiendo el manual de configuración el SW se puede configurar. La configuración tiene que ver con los datos iniciales, permisos de usuario, tablas de soporte, etc.  
**- Pruebas de instalación:** Ver si el SW se contrasta contra el manual de instalación, o sea, ver si siguiendo el manual de configuración el SW se puede configurar.

**Métodos:** Como no podemos garantizar una cobertura del 100% del testing se busca realizar la **menor cantidad de casos de prueba eficientes y efectivos**   
**Cobertura:** Porcentaje de pruebas que se hacen / El total de pruebas que debería hacerse para cubrir la totalidad de las condiciones posibles en cada caso de prueba.

**- Caja Negra:** Es un testing dinámico que se enfoca en el que, es decir que no tiene en cuenta la funcionalidad interna:  
**- Basado en especificaciones:** Usan la documentación de las especificaciones del producto.  
¬ Partición de equivalencias: Un producto bajo prueba se espera que se comporte igual que los de su clase.   
1. Identificar las clases de equivalencia válida y no válida.  
2. Diseñar los casos de prueba.  
¬ Análisis de valores límites: No se selecciona una clase de equivalencia sino los bordes.  
**- Basados en la experiencia:** La experiencia de quien determina las entradas es fundamental.  
¬ Adivinanza de defectos: Se intuyen defectos comunes y se prueban.  
¬ Testing exploratorio: El tester mientras usa el SW lo va a aprendiendo a usar y va generando pruebas.  
  
**- Caja Blanca:** Es un testing estático porque no ejecuta el código, solo lo mira por dentro analizando la estructura interna. Puede garantizar el testing coverage (que se han recorrido todos los caminos de un módulo).- **Cobertura de sentencias:** Recorrer todos los caminos lógicos.  
- **Cobertura de decisión:** Recorrer cada decisión al menos una vez x ambas ramas.  
- **Cobertura de condición:** Cada condición en una decisión tenga todos los resultados posibles.  
-**Cobertura de decisión/condición:** Cada condición tenga las salidas y cada punto de entrada sea activado.  
- **Cobertura múltiple:** Ejecutar todas las combinaciones posibles de resultados de condición en cada decisión al menos una vez.

**Testing Ágil:**  
  
**Valores:**- Testing durante sobre testing al final  
- Prevenir defectos por sobre encontrar defectos  
. Entender lo que se esta probando  
- Construir mejor el sistema por sobre probarlo  
- Equipo responsable de calidad y no solo el tester.  
  
**Principios:**  
1. Mueve hacia delante el proyecto  
2. El testing no es una fase  
3. Todos hacen testing   
4. Reducir la latencia del feedback: cliente involucrado  
5. Las pruebas presentan expectativas del usuario  
6. Mantener el código limpio y corregir los defectos rápidos  
7. Reducir la documentación en las pruebas  
8. Las pruebas son parte del done

Cuadrantes del testing  
1. Pruebas unitarias  
2. Pruebas de rendimiento y seguridad  
3. Pruebas funcionales  
4. Pruebas de aceptación.  
  
SOPORTE AL DESARROLLO VERIFICAN  
EVALUACION DEL PRODUCTO VALIDAN  
PERSPECTIVA TECNOLOGICA  
PERSPECTIVA DE NEGOCIO

**Tester Ágil:**   
Usar las pruebas para documentar los requerimientos y para conducir el desarrollo.  
Vela por la construcción del mejor sistema posible.  
Se relaciona con el cliente para hacerlo entender cuáles son las necesidades que tienen.  
Le agrega valor al producto.  
  
1. Proveer retroalimentación continuamente  
2. Agregar valor a los clientes  
3. Posibilitar la comunicación cara a cara  
4. Coraje  
5. Simplicidad  
6. Mejora continua  
7. Responder a cambios  
8. Auto organizados  
9. Foco en la gente  
10. Disfrutan