Testing

* ¿Qué es Software en Testing?
  + Software Testing:
    - Disciplina de IS que permite identificar defectos en el software alcanzado
    - Va de la mano con el desarrollo no se hace al final
  + Historia:
    - Creada en 1960 en ese momento los softwares eran de baja calidad
    - La necesidad fue porque querían productos buenos
  + Objetivos:
    - Encontrar defectos o bug
    - Aumentar el nivel de software
* Los 7 Fundamentos del Testing: Según ISTQB

1. Ejecutar pruebas nos desvela la presencia de defectos
2. El testing exhaustivo es imposible
3. La verificación (pruebas testing) de un sistema debe iniciar rápidamente es decir debe empezar junto con el ciclo de vida del desarrollo
4. La mayoría de defectos están en componentes determinados: Hay partes del sistema que debemos priorizar en testear
5. La paradoja del pesticida: Ejecutamos los test una u otra vez en una parte de app y ya no de errores entonces pasar con otro componente
6. El testing es totalmente dependiente del contexto
7. La falacia de la ausencia de errores

* Importancia de las pruebas y la automatización
  + Pruebas: Posibilita los procesos para garantizar la calidad de cualquier desarrollo
  + Automatización: Utilizar programas, apis, scripts para que nos encuentre errores de manera rápida y efectiva para conseguir la eliminación de errores
* Conceptos Importantes
  + Miembros del equipo de desarrollo.
    - Líder del Equipo: Generalmente Senior con capacidad de organizativa Se encarga de redactar y mantener actualizado los requerimientos además de llevar la relación con el cliente y ver el avance con el proyecto
    - Desarrollador: Es un programador lleva los requerimientos a código bastante lógica Back-end
    - Diseñador Gráfico UX: Front-end
    - Líder de Calidad (Tester): Se encarga de desarrollar **Suite de Test** que verifiquen que el software cumpla con los requerimientos
    - Ingeniero de calidad (Tester): Programador encargado de automatizar el Suite de Test
  + Error, Defecto y Fallo:
    - Error:
    - Defecto: Es un desperfecto en algún componente
    - Fallo: Manifestación física o visual de un defecto
  + Requisitos y Calidad:
    - Requisitos: Es una circunstancia que el cliente desea este es de manera obligatoria, exigencias del cliente
      * Tipos de Requisitos
        + No Funcional: Rendimiento, estabilidad, seguridad, funcionalidad, Integridad
        + Funcional: Correctitud y completitud
    - Calidad: Grado de un componente que cumpla con los requisitos específicos, nivel hasta donde se han cumplido los requerimientos
  + Modelo de Desarrollo de Software
    - Modelo de Desarrollo Cascada: Solo podemos seguir a la siguiente fase si se acaba una
      * Requisitos: Análisis de las necesidades del cliente, se establecen los requerimientos
      * Diseño: Describe la estructura interna del software las interfaces y los procesos
      * Implementación: Programar
      * Verificación: En si es el testing
      * Instalación y Mantenimiento: Instalar la aplicación dar mantenimiento al sistema
    - Modelo “V”: Mejora del Cascada
      * Definición de Requerimientos
      * Diseño funcional del Sistema: Los procesos
      * Diseño Técnico del Sistema
      * Especificación de Componentes
      * Código
      * Pruebas Unitarias
      * Pruebas de Componentes
      * Pruebas de Sistema
      * Prueba de Validación
    - Modelo en Espiral: Una espiral que se repite hasta que se puede entregar el producto terminado
      * Determinar objetivos
      * Análisis de riesgos:
      * Desarrollar y probar
      * Planificación: Estrategias para la siguiente fase de desarrollo



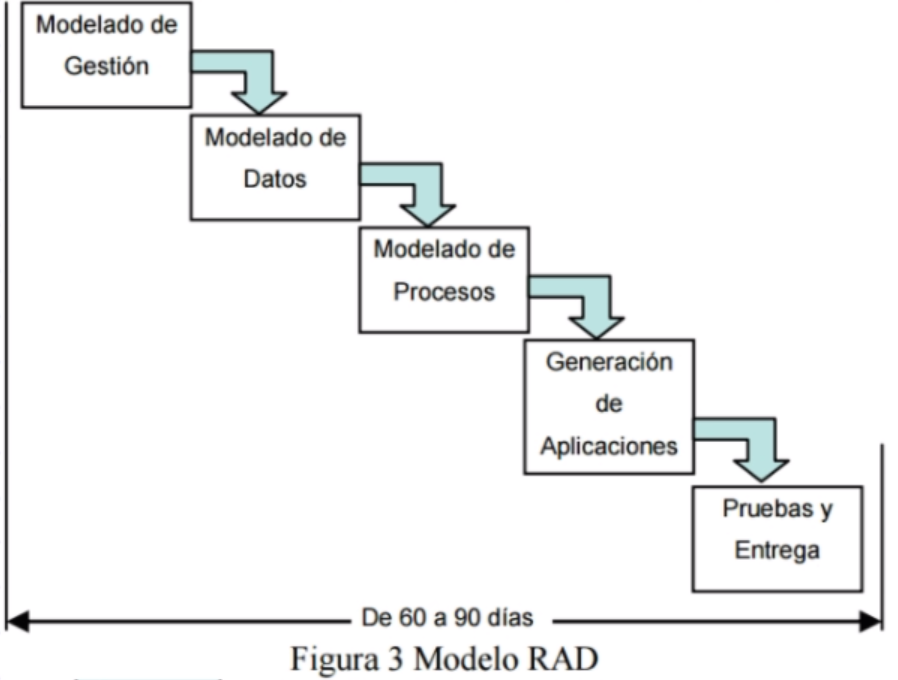


* + - Modelo Iterativo e Incremental (Modelo Scrum): Es un conjunto de tareas agrupadas en pequeñas etapas repetitivas, Un spring es toda una vuelta que tardara de 1 a 2 semanas

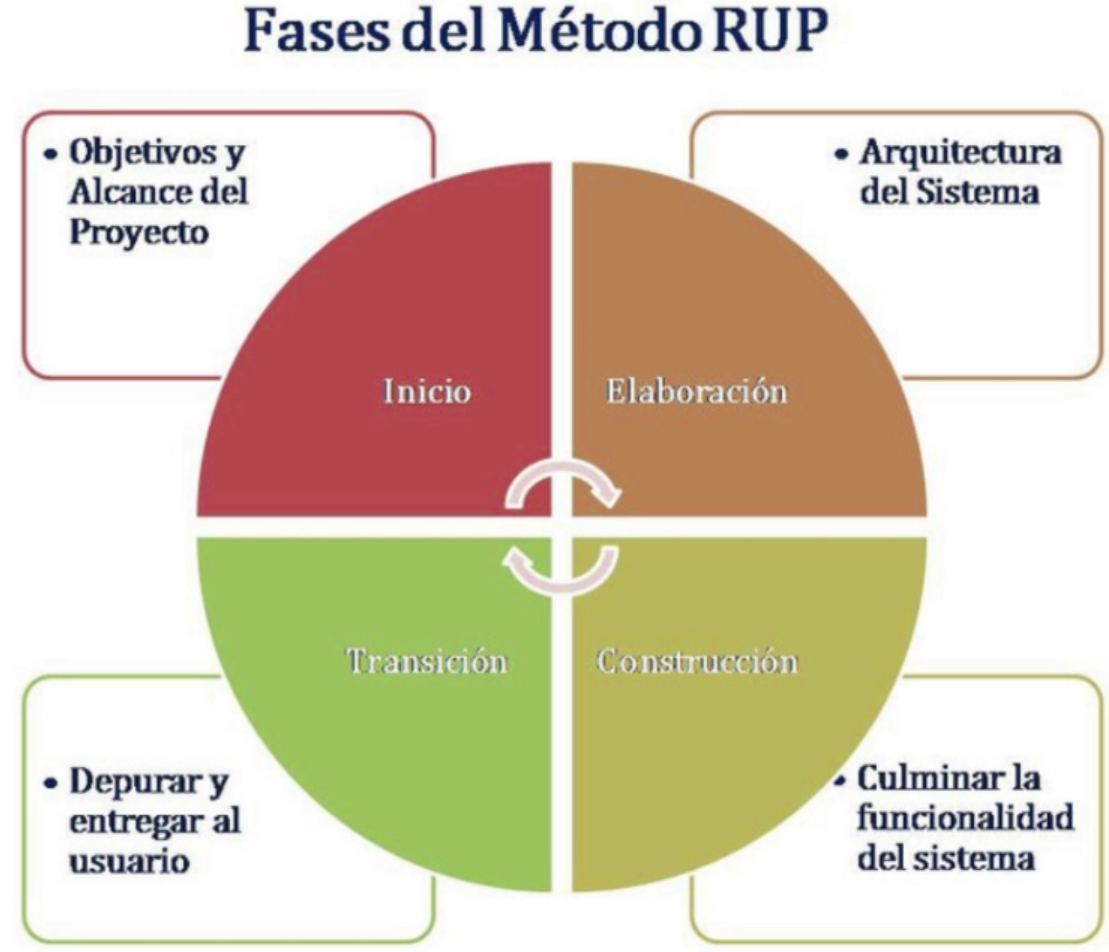




* + - Modelo RAD: Permite la construcción rápida de sistemas utilizables, es parecido a una cascada



* + - Modelo RUP (Proceso unificado de Rational):
      * Conjunto de metodologías adaptables al contexto y necesidades de cada organización.
      * Constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistema orientado a objetos
      * Rational: Es una empresa de IBM por tanto RUP es dueño de IBM

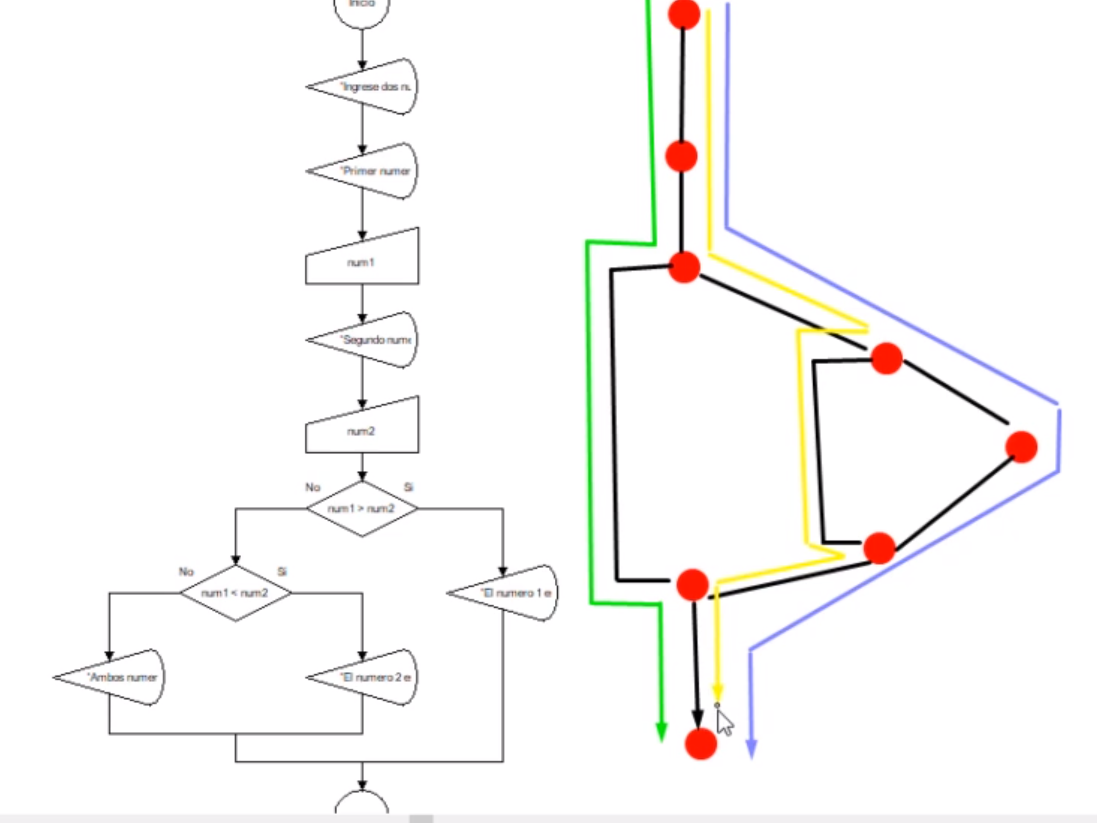


* + - * Es iterativo y presentado al cliente, en la parte de transición se hace el testing
* Procesos Fundamental del Testing:
  + Fase 1 Estudio de Requisitos: Las funcionalidades del producto son evaluadas
  + Fase 2 Análisis y diseño de pruebas
    - Objetivos:
      * Verificar la funcional del proyecto es dar un vistazo rápido del proyecto
      * Revisar los Requisitos del proyecto
      * Crear una guía de plan de test (**Guideline**)
      * Considerar una guía técnica para la Escritura del test plan (**Tests Analysis**)
    - Guideline: Documento que debe generarse y presentar al equipo de pruebas para generar los test case y sus elementos que debe contener son:
      * Normas generales para la definición de títulos del test plan
      * Indicaciones para la definición de las carpetas del test plan
      * Normas generales para la definición de los títulos del test cases
      * Propuesta para la definición del cuerpo de la prueba
    - Tests Analysis: Documento con las conclusiones y análisis que se hagan en esta fase iran en una o varios Test Analysis y los elementos que llevan son
      * Descripción de las funcionalidades del proyecto
      * Análisis de requisitos
      * Herramientas para utilizar para hacer pruebas como selenium
      * Estructura del test plane
    - Las Guideline son documentos que ayudan a hacer test case
    - Los test Analysis son documentos que ayudad para Test Plane
  + Fase 3: Definición de Estrategia de Pruebas
    - Definir la estrategia de prueba y generar un documento que la describa (**Test Strategy**) y sus elementos son
      * Descripción de las funcionalidades a probar para saber que probar y que no
      * Definir los tipos de prueba así definir la forma
      * Definir el scope de pruebas
      * Definir herramientas para testing y gestión
      * Calendario de pruebas
      * Entorno de pruebas (entorno virtualizado como Android studio o real en un maquina especfica)
      * Riesgos
      * Revisiones y aprobaciones que se pueden hacer al software
      * Definición de los ciclos pruebas hasta donde hacer las pruebas, en que momento terminar, cuantas veces repetir, cuantas veces hacerla
      * Definición del proceso de incidencias haciendo referencia a un suceso no esperado
      * Definición de métricas se refiere todo lo que se va ir preparando durante toda la ejecución para luego re subirlas en un documento
      * Definición de los criterios de calidad
  + Fase 4: Escritura del **plan de pruebas:** Unificar todos los documentos anteriores también deberíamos generar la matriz de trazabilidad es un mapeo entre las pruebas escritas y los requisitos que cubren cada prueba
  + Fase 5: Ejecución del **plan de pruebas** se empieza a ejecutar cada una de las pruebas sobre el software , se recolectan los documentos e implementan las pruebas las actividades de la fase son los siguientes
    - Definición y presentación de los GuideLines de ejecución
    - Personas de referencia en el proyecto
    - Presentación del flujo del ejecución de los ciclos
    - Seguimiento y reporte de progreso
  + Fase 6: Evaluación: Sirve para crear un documento con las conclusiones totales entonces ya se ejecutaron todas las pruebas este documento se llama **TestWare** y tiene
    - Las funciones probadas dentro de la App o sistema
    - Equipos de testing para saber quiénes se encargaron de hacer las pruebas ,quienes documentaron ,quienes hicieron los test case y los otros documentos
    - Métricas
    - Conclusiones: saber si se cumplieron los objetivos si se saltaron pasos y capaz nuevas mejorar para el software que se puedan hacer
  + Fase 7: Cierre y Mantenimiento : Para finalizar el proyecto debe hacer una firma de cierre con los encargados de hacer el software y los stakeholder
    - El mantenimiento debe cobrarse aparte si es el tipo de mantenimiento que se cobra y no estuvo en el contrato
* Diseño de las Pruebas:
  + Test Case: Conjunto de condiciones para saber si un componente es satisfactorio en si es una prueba de testing documentada que se realiza a los componentes y la estructura del Test Case es
    - Introducción y visión general
    - Actividades
    - Resultados
  + Testing Unitario: Es comprobar una unidad de código como una función, modulo recordad no es todo el código fuente, es una forma de comprobar el correcto funcionamiento de una unidad de código
    - características son:
      * Automatizable
      * Completas es decir de comprobar la mayor cantidad de codigo
      * Repetible
      * Independiente
      * Profesionales
    - Ventajas
      * Fomenta el cambio
      * Documenta el codigo
      * Los errores están más acotados y son más fáciles de localizar
    - Desventajas
      * No se descubren todos los errores
      * No se descubrirá los errores de integración
  + Testing Integración: Verificar si las comunicaciones entre los distintos componentes presentes en el software son funcionales
    - Características:
      * Comunicación del lado del Hardware y Software si es estable, que tanto recursos utiliza
      * Se verifican los datos de entrada y salida
      * Inicia justo después del Testing Unitario
  + Testing de Sistema: Ejercitar profundamente el sistema comprobando la integración global del proyecto
    - Pruebas Funcionales: Todo el sistema de información realice las funciones que el cliente pidió
    - Pruebas de Comunicación: Todas las interfaces ya sea de dispositivos remotos como locales funcionen entre si
    - Pruebas de Rendimiento: Es el tiempo en que se tarda en responder el resultado
    - Pruebas de Volumen: Pruebas donde se hacen un gran de volumen de datos y usuarios para saber cómo trabaja
    - Prueba de sobrecarga: Poner al límite la aplicación por ejemplo si la aplicación tiene como límite 1gb de RAM hay que probarla en ese entorno de un 1 solo gb de RAM
    - Prueba de Disponibilidad de Datos: Como se recupera el sistema después de un facho improvisto accidente
    - Pruebas de Facilidad de Uso: Que tan intuitivo es la aplicación, que tan fácil es de usar para el usuario
    - Pruebas de Operación:
    - Pruebas de Seguridad:
    - Pruebas de Entorno:
  + Testing de Implantación: Comprobar el funcionamiento correcto del sistema integrado de hardware y software en el entorno de operación en si llevar el sistema al cliente donde lo va usar
  + Testing de Aceptación: El usuario utilice el software y apruebe su respectivo rendimiento y funcionalidad
  + Testing de Regresión: Comprobar que los cambios sobre un componente de un sistema de información no introducen un comportamiento no deseado este solo pasa cuando el cliente pide un cambio o un arreglo porque algo fallo donde este testing lo que hará es que cuando se hagan cambios dichos cambios no generen errores
    - Características:
      * Repetir test case ya realizados
      * Revisión manual antes de realizar los cambios
      * Diccionarios de Datos impresos
* Técnicas para realizar las pruebas: Existen 2 tipos de pruebas Técnicas estáticas y técnicas dinámicas, las técnicas son comprobaciones que se efectúan sobre un software estas técnicas también se les puede llamar tipos entonces lo que vemos es como se realizan las pruebas como cuales son los métodos cuales son las formas
  + Pruebas Estáticas: Son las pruebas que se realizan sin ejecutar el código fuente del programa, entonces es revisar el código sin ejecutar esta busca defectos
    - Revisiones: Es la más usada para pruebas Estáticas, son pruebas que permite detección de defectos de defectos en etapas tempranas
      * Se toma el código fuente y se analiza todos los posibles defectos como
        + Defecto de Requisitos
        + Defectos de Diseño
        + Especificación de Interfaz incorrectas
        + Desviaciones de los estándares: Normas prácticas, malas prácticas de programación
      * Revisión Formal: Detectar y registrar los defectos de un producto intermedio y que se ajusta a los estándares establecidos
        + Planificación
        + Definición de Criterios: Definir las medidas para que la revisión sea de calidad
        + Kick-off: Reunión donde se habla del proyecto
        + Identificación de defectos
        + Reunión de Revisión: Hablar con el cliente los avances del testing debe estar el usuario como el equipo total
        + Reconstrucción: Documentado los procesos a llevar la prueba
        + Seguimiento: Nos dará a saber si se llegó a resolver el defecto o no
        + Comprobación de Criterios
      * Revisión Informal: Los tester no tiene instrucciones escritas, el obejtivo es obtener defectos de manera barata y rápida, además lo puede hacer el mismo desarrollador y puede que sea documentada o no
        + Revisión Guiada: Pueden variar entre muy formales y hasta poco formales llevadas por los autores de los documentos
        + Revisión Técnica: Comprobar que el software se ajuste a especificaciones y que se este elaborando según las normas y guias aplicables al proyecto llevada por el jefe del proyecto o el tester
    - Inspección: Se basa en leer documentos hacer un examen visual para detectar defectos sus encargados son el revisor que verifica las anomalías, escriba es el que registra el defecto y moderador es responsable de organizar la prueba y de las actividades en la inspección, sus características son :
      * Es un proceso documentado
      * Con objetivos claros
      * Informes con listas de conclusiones
    - Análisis Estática: Pruebas en el código para hallar defectos rápidos esto generalmente es hecho por medio de un software de tercero estos tipos de defectos son
      * Referencia a una variable con un valor indefinido
      * Interfaces inconsistentes en el modulo
      * Variables que no se utilizan o declaran de forma correcta
      * Código inaccesible y ausencia de lógica
      * Infracciones de los estándares
  + Pruebas Dinámicas: Son las pruebas que requieren la ejecución del código y de la aplicación y esta busca fallos
    - Técnica de Caja Blanca: El fin es ver la operatividad de cada uno de los componentes viendo su lógica es decir encajar módulos y verificar que funcionen de manera interna
      * Objetivos:
        + Garantizar que todas las rutas de código sean probadas
        + Revisan condiciones lógicas
        + Revisan estructuras de datos
      * Técnicas
        + Cobertura de Sentencia: Requieren que se ejecuten una vez cada sentencia del programa y detectar bloques de código no usado, pero no comprueba todos los caminos que existen

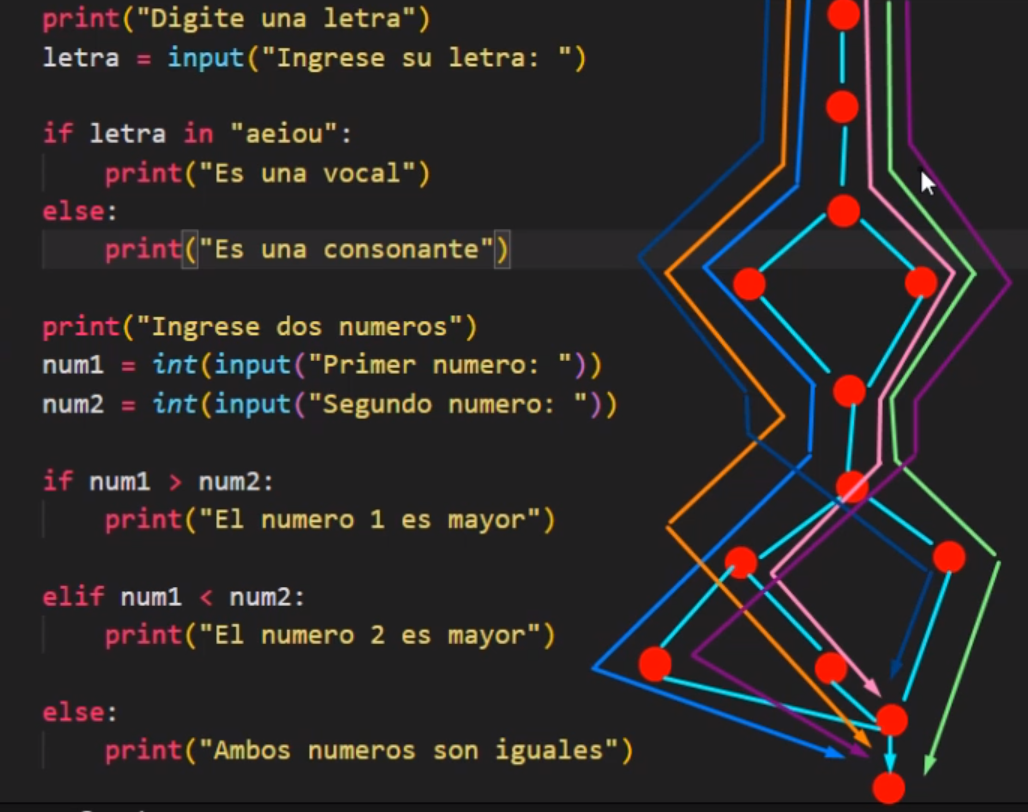
Fórmula para saber que cobertura alcanzamos con la prueba nota cada línea es una sentencia

* + - * + Cobertura de Decisión: Es necesario escribir un número suficiente de casos de prueba como para que cada decisión tenga por lo menos un resultado verdadero o falso

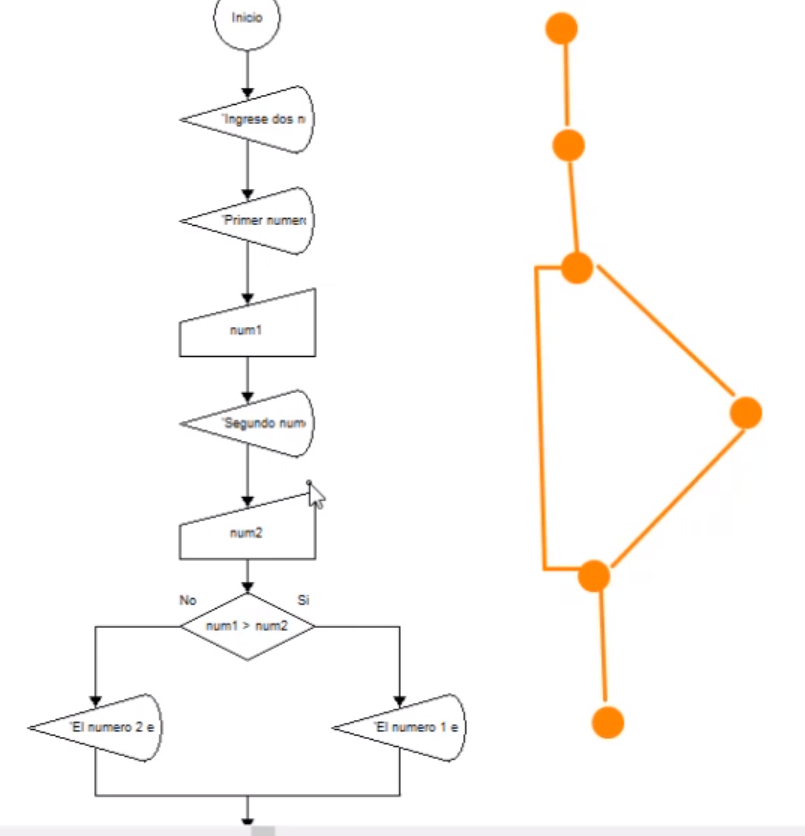
Formula



* + - * + Cobertura de Caminos: Se comprueban los caminos lineales que son independientes, es decir probar todo los caminos posibles como se dijo imposible



* + - Técnica de Caja Negra: Demostrar que las funciones son operativas de manera externa
      * Funciones incorrectas o faltante
      * Errores de iniciación o finalización
      * Errores de Interfaz
      * Errores en las Estructuras
  + Grafos de Flujo



Caja Blanca:

Caja Negra:

Como crear el test Case:

* ID
* Nombre del caso
* Descripción
* Precondición
* Pasos
* Resultado esperado
* Resultado Actual
* Comentario del teste
* Referencias

Selenium: Codigo abierto

Compatibles con varios lenguajes como java,C#, php,ruby,JS y Python

Multiplataforma: Window ,Mac, Linux

Varios navegadories: Chrome,safary,Edge ,Opera…

Herramientas que facilitan la documentación como

Herramientas de despliegue: Jenkins, Maven y Docker.

Solo funciona para aplicaciones web mas no nativas de escritorio

Metodos Xpath:

Identificar componentes con selenium que pueden ser componentes que no poseen atributos

XPath es un lenguaje que permite construir expresiones que recorren y procesan un documento XML.

Extensión ChroPath:

<https://chrome.google.com/webstore/detail/chropath/ljngjbnaijcbncmcnjfhigebomdlkcjo>

// -> significa ruta relativa que iniciamos una ruta relativa

//viñeta[@atributo=’alorAtributo’] -> ruta relativa completa

Ejemplo en youtube: *//input[@id="search"]*

Ruta Absoluta es aquella que jamás va a cambiar:

Ejemplo del buscador de youtube: */html[1]/body[1]/ytd-app[1]/div[1]/div[1]/ytd-masthead[1]/div[3]/div[2]/ytd-searchbox[1]/form[1]/div[1]/div[1]/input[1]*

**Unitest:**

Es un modulo que permite la creación de pruebas, para automatización usando POO.

Respuestas de Cada Test:

* OK: Exitosa
* FAIL: Indica que la prueba no ha ocurrido de manera correcta
* ERROR: Cuando no ha ocurrido correctamente pero tampoco ha sido una aserción

Cada método dentro de la clases es un test case

Todo los métodos de la clase para que sea *test case* tienen que empezar con la palabra test

Los assertions son implementos importantes

Para ejecutar se debe usar el método main()