Verificacion y Validación del Software

Tema 29. Técnicas.
Tema 30. Métodos de Pruebas. (press 16)
Tema 31. Estrategias de Pruebas. (press 17)
Tema 32. Sistemas Críticos. (somer 21)

1. INTRODUCCION

1. Definición: V&V, implicaciones, elementos software, responsables, impacto

- Definición
  - En la guía de verificación y validación del software de la PSS-05 se indica que la VVS son las actividades que chequean el software contra sus especificaciones.
  - Forma parte del proceso de control de la gestión del software
  - Es un proceso continuo a lo largo de todo el proceso de desarrollo
- Esto implica:
  - Comprobar que cada elemento software cumple con sus requisitos específicos.
  - Comprobar cada elemento software antes de emplearlo como entrada en cualquier otra actividad
  - Asegurar que las comprobaciones son llevadas a cabo por una persona distinta
  - Asegurar que los esfuerzos dedicados en VVS en cada elemento software son los apropiados para el uso que se le va ha dar a dicho elemento.
- Que es un elemento software:
  - Todos los elementos implicados en el desarrollo del software, incluyendo código, documentos, modelos, COTS etc.
- Quien es el responsable
  - El gestor de proyecto es el responsable de organizar las actividades de verificación y validación.
  - De definir los roles en el procesos de VVS (p.e líder de equipo de revisión)
  - Asignar personal a esos roles
- Impacto del VVS
  - Afecta a la calidad del software
    - En cuanto al fallo del software
    - No cumplir los requisitos
  - Involucra una cantidad importante de esfuerzo
    - Entre el 30-90% del coste del proyecto, dependiendo de la complejidad y criticidad del software

2. Diferencia entre Verificación y Validación, Nivel de confianza. Diferencia entre V&V y pruebas Clasificación de técnicas de V&V

- Según las definiciones del ECSS-E-40A
- Verificación
  - Confirmación mediante examen y evidencias objetivas que un elemento cumple con los requisitos específicos.
  - El proceso de verificación debe asegurar que para cada actividad existen las entradas y especificaciones adecuadas, y que las salidas de esa actividad son correctas y consistentes con dichas entradas y especificaciones.
  - ¿Estamos construyendo el producto correctamente?
- Validación
  - Confirmación mediante examen y evidencias objetivas que un Sw cumple con los requisitos particulares para un uso específico.
  - El proceso de validación asegurar que en el producto final Sw las funciones baseline y los rendimientos requeridos están correcta y completamente implementados.
  - Es una verificación end-to-end
  - ¿Estamos construyendo el producto correcto?
- Diferencia entre V&V y pruebas
  - La segunda es una técnica de V&V
- Objetivo final: tener confianza en que el software va a funcionar. El nivel de confianza requerido depende de:
  - Propósito del sistema:
    - cuan crítico es el software para la organización
  - Expectativas del usuario
    - En cuanto el Sw no puede fallar. La moral del usuario a este respecto es baja y pierde confianza.

- Entorno de mercado
  - Competidor, tiempo de puesta en el mercado.
- Clasificación de Técnicas
  - Tema->Técnicas

3. Planificación de las actividades de V&V: limitaciones de V&V, En desarrollo SW, principios y estructura

- Limitaciones
  - No es posible probarlo todo
  - Hay que medir la profundidad de la prueba
  - Hay que guiar las pruebas por los errores más peligrosos->categorización de la pruebas
  - Por el coste en recursos que tiene
  - Por el impacto en la planificación
- En el desarrollo Sw
  - Etapas más recientes del desarrollo
  - Limitar el impacto de encontrar un error
- Principios
  - Registros: para saber que se ha probado
  - Repetitividad: amplía el anterior, poder repetirlo.
  - Objetividad: amplía el anterior, tener los mismos resultados
  - Regresión: amplía el anterior, repetirlos de forma automática y comprobando que los cambios no producen efectos de lado no deseados.
- Estructura
  - Descripción del proceso de test
    - En sus aspectos más destacados
  - Traceabilidad de requisitos
    - Para orientar el test a que se cumplen los requisitos
    - Comprobar la tracebilidad de los requisitos de usuario contra los requisitos Sw
    - Comprobar que los componentes software son traceables contra los requisitos Sw
  - Elementos de test
  - Plan de trabajo de test
    - Asignación de tiempo y recursos
  - Procedimiento de registros
  - Requisitos Hw y Sw
    - Software tools y utilización del Hw
  - Restricciones
    - Tiempo, falta de personal, de conocimiento etc.

4. Normas y herramientas, Workbench

- Normas
  - Validación: PSS-05-10, ECSS-E40A
  - Pruebas: BS 7925
  - The IEEE Standard for Software Verification and Validation (IEEE Std 1012-1998) contains information on software integrity levels, the V & V process, the Software V & V reporting, administrative, and documentation requirements, and an outline of the software verification and validation plan.
- Herramientas
  - Pruebas de cobertura: Rational Coverage - Purify
  - Busqueda de errores: depuradores
  - Medidas de rendimiento: gprof
  - Herramientas de generación de vectores de test o passwords

2. T29 - TÉCNICAS

5. Tipos de técnicas, Error/Fiabilidad, Análisis/Medida

- Tipos de técnicas:
  - Error/Fiabilidad
  - Análisis/Medida
  - Estáticas/Dinámicas
  - Manuales/automáticas
  - Fase de utilización en el ciclo de vida

- Error/Fiabilidad
  - Tipos de estrategia que son radicalmente distintas
    - Búsqueda del error
      - Pretende buscar en qué funciona mal el software
    - Análisis de la fiabilidad
      - Pretende determinar cuan bueno es el software
  - El error
    - Descubrir un error
    - Buena prueba es la que tiene una alta probabilidad de demostrar un error no descubierto hasta entonces
    - Una prueba que tiene éxito es la que descubre un error
    - Una prueba no demuestra que no hay errores
  - Técnicas de fiabilidad
    - Test estadístico
    - Se emplea para deteminar el rendimiento y la fiabilidad del programa.
    - Se realiza sobre las condiciones reales de trabajo y con las frecuencias de entrada y salida reales.
    - Después de la ejecución de los test, se realiza un análisis de los fallos o defectos encontrados para determinar la fiablidad del sistema (cuantas veces se ha colgado, no necesariamente errores de código).
    - Tema->SistemasCríticos
- Análisis/medida
  - Análisis
    - Técnicas que analizan la consistencia
    - El programa utiliza la sintaxis correcta, emparejamiento de parámetros, estilo, translación de especificaciones.
  - Medida
    - Técnicas que miden alguna propiedad de los elementos software
    - Tema->Métricas

6. Inspecciones de software o estáticas, Revisiones: revisión peer2peer, inspección fagan, walkthrough, auditorias

- Inspecciones de software o estáticas
  - Tiene como objeto: documentos, modelos, código
  - Se aplica en todas las fases, esto es muy importante por que permite iniciar las V&V antes de escribir una sóla línea de código, que es lo que hay que hacer.
- Revisiones:
  - Son técnicas estáticas para determinar la consistencia
- Revisiones peer2peer
  - Revisiones cruzadas
- Inspecciones -inspección de Fagan
  - Son inspecciones para evaluar el código de sentencia en sentencia.
  - Tiene las siguientes 6 fases:
    - Planificación:
      - Cuando el desarrollador completa su un ""producto de trabajo"", se forma un grupo de inspección y se designa un moderador. El moderador asegura que el ""producto de trabajo"" satisfaga el criterio de inspección. Se le asignan diferentes roles a las personas que integran el grupo de inspección, así como la planificación de tiempos y recursos necesarios .
    - Overview
      - Si los inspectores no están familiarizados con el desarrollo del proyecto, un vista general es necesaria en éste momento. Este es un paso opcional, pero no menos importante ya que en esta etapa se dará al grupo de inspección un "background" y el contexto a cubrir por las inspecciones.
    - Preparación
      - Los inspectores se preparan individualmente para la evaluación en la reunión, estudiando los productos de trabajo y el material relacionado. Aquí es aconsejable la utilización de listas de chequeos para ayudar a encontrar defectos comunes, y . El tiempo que pueda llevar esta etapa va a depender de cuan familiarizado esté el inspector con el trabajo que debe analizar.
    - Examen:
      - En esta etapa, los inspectores se reunen para analizar su trabajo individual en forma conjunta. El moderador deberá asegurarse que todos los inspectores están suficientemente preparados. La persona designada como lector presenta el "producto de trabajo", interpretando o parafraseando el texto, mientras que cada participante observa en busca de defectos. Es recomendable que este examen no dure mas de 2 horas ya que la atención en busca de defectos va disminuyendo con el tiempo. Al terminar con la reunión, el grupo determina si el producto es aceptado o debe ser retrabajado para una posterior inspección.
    - Retrabajo:
      - El autor corrige todos los defectos encontrados por los inspectores.
    - Seguimiento:
      - El moderador chequea las correcciones del autor. Si el moderador está satisfecho, la inspección está formalmente completa, y el "producto de trabajo" es puesto bajo el control de configuración.

- Walkthrougs
  - Es una inspección como la de Fagan pero con las siguientes características:
    - El análisis es presentado por el propio desarrollador
    - El mecanismo para conducir la inspección es el desarrollo de un escenario sobre el código.
    - El desarrollo de la prueba sirve para iniciar la discursión.
- Walkthought cognitivo
  - Es un walkthrougt (ver tema anterior) pero guiado solo a través del análisis de los pasos de un GUI. Tomando los revisores el papel de usuarios y siguiendo el proceso a través de un escenario o de un test.
- Auditorias
  - Sobre todo realizadas por elementos externos. Buscan inconsistencia en todo el proceso (calidad, configuracion, manuales de estilo) y puede ser tambien una técnica de validación.

7. Estáticas - análisis: Análisis de confianza, Ejecución simbólica, pruebas formales, Traceabilidad, Checklist

- Análisis de confianza - Parecido al Análisis de Riesgos en seguridad
  - Identificar peligros y proponer métodos para reducir el riesgo de que estos riesgos se produzcan. No es exactamente el análisis de riesgos que se realiza en la etapa de planificación pues el primero está orientado a los riesgos para que el proyecto llegue a buen término, mientras que este está orientado a que los problemas puros de la operación del software
    - Análisis de los peligros - implica el uso de guías para identificar peligros, sus raices y posibles contramedidads.
    - Análisis del riesgo - Yendo más allá, identifica las consecuencias y la probabilidad de que ocurra.
  - Ejemplo, utilizar números aleatorios para la identificación de una comunicación (como en TCP/IP) y que se produzca una coincidencia.
- Ejecución simbólica
  - La ejecución simbólica es un método de análisis estático del código en el que se recorre la ejecución del código pero en vez de que con valores de un escenario, se realiza asignándo símbolos a las variables.
  - En las ramificaciones se analizan:
    - Las condiciones de camino
    - En cada condición de camino se define un camino de ejecución que es posible analizar.
- Pruebas formales
  - Formal methods means the use of mathematical and logical techniques to express, investigate, and analyze the specification, design, documentation, and behavior of both hardware and software.
- Traceabilidad
  - Hacia adelante y hacia atras
- Checklist

8. Pruebas del software o dinámicas

- Test del programa ejecutándose
- Tema->Métodos de prueba

9. Manual/Automática - Fases del ciclo de vida

- Manual/Automática
  - Chequeo automática
    - Tema -> SistemasCríticos->Seguridad
  - Regresión
- Utilización a lo largo del ciclo de vida
  - Tema->Estrategias de prueba
  - Utilización a lo largo del ciclo de vida

ESTÁTICOS	DINÁMICOS
Requirements analysis and functional specification	
walkthroughs, design reviews, checklists	
Top-level design	
walkthroughs, design reviews, checklists, formal proofs, fagan inspection	
Detailed design	
walkthroughs, design reviews, control flow analysis, data flow analysis, symbolic execution, checklists, fagan inspection, metrics	

Implementation	
static analysis	functional testing, boundary value analysis, structured-based testing, probabilistic testing, error guessing, process simulation, error seeding
Integration testing	
walkthroughs, design reviews, sneak circuit, analysis	functional testing, time and memory tests, boundary value analysis, performance testing, stress testing, probabilistic testing, error guessing
Validation	
Auditoria	functional testing

➢

3. T30 - MÉTODOS DE PRUEBAS

10. Principios, Perfiles de fallo, priorización y categorización, Facilidad, Tipos, Propósito

- Principios
  - Tiene que seguir una traceabilidad hasta los requisitos
  - Planificarse antes de que empiecen
  - El principio de Pareto es aplicable:
    - 80% de los errores en el 20% de los módulos. El problema es encontrar esos módulos
  - Las pruebas han de ser bottom-up
  - No son posibles las pruebas exhaustivas.
- Perfiles de fallo, Priorización y categorización del error
  - A medida que se van haciendo las pruebas y se recogen los datos de los errores, se pueden emplear los perfiles de fallos para dar prioridad y categorizar los errores descubiertos La prioridad indica la severidad del problema.
  - Las categorías de los fallos proporcionan una descripción de un error, de manera que se puedan llevar a cabo análisis estadísticos de errores.
- Facilidad de la prueba
  - En un proceso de prueba es muy importante seguir unos principios que hagan la prueba lo más sencilla posible para minimizar su coste:
    - Cuanto mejor funciona más fácil es de probar.
    - Lo que ves es lo que pruebas
    - A mayor control del software más fácil es el proceso de pruebas
    - Controlar el ámbito de pruebas hace que sea más fácil probar
    - Cuanto menos haya que probar más fácil es probarlo
    - Cuantos menos cambios menos interrupciones en las pruebas
    - Cuanta más información más inteligentes son las pruebas.
- Tipos
  - En cuanto al conocimiento del elemento a probar:
    - Caja blanca
    - Caja negras
- En cuanto a su propósito
  - Prueba de defectos
  - Análisis de la fiabilidad
    - Tema->SistemasCríticos

11. La complejidad ciclomática

- Básico para la comprensión del diseño de pruebas de caja blanca
- Software como grafo
  - V(G) = Aristas-Nodos+2
- Matrices de grafos
- Complejidad máxima aconsejada 10.

12. Pruebas de caja blanca: sentencia, condición, cobertura, flujo de datos, bucles, aleatorio

- Statement testing
  - Comprobar que se ha pasado al menos una vez por todas las sentencias.
- Pruebas de condición (branch/decision testing)

- Más allá que el anterior, pues implica testear las posibles condiciones (en positivo o negativo)
- Se pretende que todas las condiciones de salto o ruptura de secuencia sean ejecutadas.
- Esto no implica seguir todos los posibles caminos, pero sí ejecuta todas las sentencias y condiciones.
- No os muy apropiado cuando las condiciones son muy complejas.

➢ Cobertura de caminos
- Se asegura que todas las sentencias se han ejecutado una vez

➢ Prueba de flujo de datos
- Pretende asegurar que todas las variables se han usado en la prueba.
- Selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
- Cadena de definición – uso { Var, Sentencia de definicion, Sentencia de utilización}
- Es como camino de datos pero se ha de probar al menos una vez cada cadena DU

➢ Prueba de bucles
- Los bucles son la piedra angular de la mayor parte de los algoritmos implementados en Sw.
- Bucles simples
  - Pasarlo por alto
  - Pasarlo una vez
  - Dos veces
  - Un valor x 2<x<maximo
  - Hacer maximo-1 y maximo+1
- Anidados
- Concatenados
- No estructurados

➢ Test aleatorio

13. Pruebas de caja negra: grafos, particiones, límites, sintáctico, aleatorio, back2back

➢ No es una alternativa a las pruebas de caja blanca

➢ Métodos basado en grafos
- Se identifican objetos abstractos en la aplicación que se conectan y se evalua la progresión de la aplicación.
- Tipos
  - Modelado del flujo de transacción o de causa efecto
    - Se toman los requisitos y se analizan las condiciones (causas) y las subsecuentes acciones que ha de realizar el sistema. El grafo se translada entonces a una tabla de decisión.
  - Modelación del estado finito
    - Se modela el sistema como una maquina de estados con transiciones en función de eventos.
  - Modelación del flujo de datos
  - Modelación de la planificación – Medida temporal de la ejecución de procesos

➢ Particiones equivalentes

➢ Análisis de valores límite

➢ Prueba de comparación

➢ Analisis sintáctico
- Se analiza la sintaxis de los datos de entrada para generar casos de prueba
- Ej: analisis de la dirección de correo

➢ RandomTest

➢ Back to Back – prueba contra un prototipo -> Requisitos\Prototipos

➢ Pruebas de conjetura
- Toman en cuenta la experiencia
  - Es un caso de test negativo en el que se crean test para situaciones que en el pasado se sabe que han creado problemas.
- Precisan de tener en el proceso de desarrollo un registro de errores en otros proyectos.
- No es sistemático y solo se utiliza como complemento.

14. Métodos específicos:GUI, inyección fallos, C/S, Tiempo real, Docmt y ayuda

➢ GUI
- Métodos heurísticos
  - Proporciona un conjunto de checklist de las caracteríticas que tienen que cumplir el GUI para usabilidad. Como: simpleza, uso de jerga del usuario, realimentación, shortcuts etc.
  - También puede guiar la fase de diseño.
- Walkthought cognitivo
  - Es un walkthrougt (ver tema anterior) pero guiado solo a través del análisis de los pasos de un GUI. Tomando

los revisores el papel de usuarios y siguiendo el proceso a través de un escenario o de un test.

- Inyección de fallos
  - Fault injection is the intentional activation of faults by either hardware or software means to observe the system operation under fault conditions.
    - Hardware fault injection - Can also be called physical fault injection because we are actually injecting faults into the physical hardware.
    - Software fault injection - Errors are injected into the memory of the computer by software techniques. Software fault injection is basically a simulation of hardware fault injection.
- Arquitecturas Cliente-Servidor
- Documentación y ayuda
- Sistemas de tiempo real
  - Prueba de tareas
  - Prueba de comportamiento
  - Pruebas intertareas
  - Prueba de sistema

4. T31 - ESTRATEGIAS DE PRUEBAS

15. Que es, que incorpora, estrategia general V y prueba de unidad

- Una estrategia de prueba del software integra las técnicas y métodos de validación y verificación en una serie de pasos bien planificados que dan como resultado una correcta construcción del software.
  - Proporciona un mapa a seguir para el responsable del desarrollo Sw, organización y al cliente
  - Cuanto tiempo esfuerzo y recursos.
- Incluye:
  - Planificación de la prueba
  - Criterios para completar la prueba
  - Procedimiento de evaluación de los datos resultantes.
- Estrategia general de las pruebas
  - Está ligada al proceso de desarrollo Sw.
  - Modelo V de verificación y validación.
- Prueba de unidad
  - Qué es
    - Prueba de los módulos unitarios
    - Se considera parte del proceso de codificación
  - Entorno de pruebas
    - Requiere muchas veces el desarrollo de un entorno de pruebas:
      - Programa de invocación del módulo
      - Módulos dummy que son los invocados por el modulo bajo test
    - Este desarrollo a veces es bastante costoso

16. Prueba de integración: qué es y estrategias, Especiales: Tiempo real

- Qué es:
  - Método sistemático para construir la estructura del programa
  - Mientras se prueba la interacción de cada uno de los módulos
  - Hay que ver la estructura como en un gran arbol
- Estrategias
  - Big-bang
  - Incrementales - Descendente
    - El módulo principal es el conductor de la prueba.
    - Necesita emplear módulos dummy para suplir los que no se han desarrollado todavía.
    - Se van sustituyendo los módulos dummy por los reales uno a uno
    - Se sigue una estrategia de regresión
    - Tipos:
      - Primero en profundidad
      - Primero en anchura
  - Ascendente
    - Los módulos inferiores (las hojas del arbol) se asocian en grupos por función.
    - A estos grupos se les añade un controlador
    - Y se sigue el proceso hacia arriba
  - Combinación guiada por los módulos críticos
- Tiempo Real

- Tema->STR->Características->Pruebas

17. Prueba de sistema: delegación, recuperación, seguridad, resistencia, Especiales: C/S

- Delegación de culpabilidad
  - No solo incumben al proceso Sw sino al Hw tambien.
  - Como el equipo es interdisciplinar cuando no funciona un componente hay una delegación de culpabilidad.
  - Estrategias para manejar este problema:
    - Diseñar caminos de manejo de errores de toda la información procedentes de otros sistema.
    - Simular previamente pruebas que introduzcan datos corruptos en el sistema o errores de interfaz.
    - Participar en el proceso de diseño de pruebas del sistema para asegurar que el Sw se prueba de acuerdo a especificaciones.
- Prueba de recuperación
  - Cuanto tarda e recuperarse de una caida del sistema
- Prueba de seguridad
  - Piratas
- Prueba de resistencia
  - Pruebas de carga
  - Situaciones anormales
  - Prueba de sensibilidad
    - Descubrir combinaciones de datos dentro de un margen de entradas válidas que pueda producir inestabilidad o un proceso incorrecto
- Prueba de rendimiento
  - Muchas veces similares a las de resistencia
  - Medida de recursos

18. Prueba de validación: caja negra, configuración, alfa-beta, Especiales: Estadísticas y análisis de fiabilidad

- Construcción de las pruebas de aceptación
  - Pruebas de caja negra
- Revisión de la configuración
- Alfa y Beta
  - Alfa: Prueba desarrollada por el usuario pero en el lugar de desarrollo
  - Beta: Prueba desarrollada en el lugar de despliegue

19. Cleanroom

5. T32 - SISTEMAS CRÍTICOS

20. Qué es un sistema crítico-STR. Diferencia de las pruebas de otros sistemas, Importancia del proceso: fiabilidad, safety, security, STR, Garantía de proceso, Roles

- La V&V de sistemas críticos ha de ser entendida en el contexto de cualquier otro sistema con algunas características o procesos adicionales debidos a:
  - Coste del fallo
  - Validación de requisitos no-funcionales (fiabilidad, seguridad (safety), seguridad (security).
- Importancia del proceso:
  - Un mayor coste de las actividades de V&V
  - La utilización de normas de aseguramiento del proceso (ISO 9000)
  - Esto es lo más importante para aseguramiento de los sistemas críticos.
- Se estudiarán los procesos particulares para sistemas críticos en cuanto requisitos de:
  - Fiabilidad
  - Safety
  - Security - Protección
- Garantía de proceso
  - Importante para todos los críticos pero sobre todo para safety.
    - Los accidentes son raros en estos sistemas y es muy dificil simularlos.
    - Los requisitos de safety son del tipo "no-debe". Es muy dificil hacer test para ello.
  - El ciclo de vida ha de contemplar en todas las etapas que se contemplan los requisitos de safety. Entre ellos:
    - Crear un documento en donde se refleje el log de riesgos alcanzados y llevar ese mismo log adelante.
    - Se usa en cada etapa del proceso de desarrollo para determinar que los riesgos se han tendio en cuenta de

forma apropiada.

- Roles
  - Ingeniero de safety del proyecto.
  - No envuelto en el desarrollo.
  - Hacer seguimiento de todos los procesos de seguridad.
  - Certificado.

21. Validación de fiabilidad: prueba estadística(etapas); predicción de la fiabilidad (modelos escalón, escalón aleatorio, continuo, poisson-beneficios)

- Prueba estadística.
  - Implica tener un método para medir la fiabilidad.
  - Este sistema, al contrario que los sistemas de test, no busca el fallo sino todo lo contrario Tiene cuatro etapas:
    - Obtención de un perfil operativo de los sistemas en uso.
      - El perfil operativo es el estudio de conjunto de posibles entradas.
    - Se construye un conjunto de entrada de test en función del perfil anterior
    - Se prueba el sistema contra estos datos y se computa:
      - Número de fallos
      - Tiempo o frecuencia de los errores
    - Después de obtener un conjunto de valores estadísticamente significativos se puede computar la fiabilidad con la oportuna métrica
  - Dificultades de realizar en la práctica
    - Incertidumbre del perfil operativo
    - Gran coste en la generación de datos de pruebas
    - Incertidumbre estadística cuando se requiere una gran fiabilidad.
- Predicción de la fiablidad
  - Principios
    - Parar el test tan pronto como sea posible
    - Detenerse para cuando se obtenga el nivel de fiabilidad deseado.
  - Problemas
    - Puede que la predicciones indiquen que nunca se alcanzará el nivel de fiabilidad deseado.
    - Implica re-escribir el software o negociar con el cliente.
  - Modelo de crecimiento de la fiabilidad
    - Modelo de como cambia la fiabilidad del sistema durante el proceso de test.
    - Precisa de un modelo matemático para poder realizar una predicción.
    - Como medida se emplea el ROCOF (rate of ocurrence of software failures)
  - Tipos de modelos
    - Función escalón
      - Se incrementa la fiablidad por cada error descubierto y reparado.
      - La escala temporal no es constante pues varía en función de cuando se ha encontrado el error.
      - Problema: reparar un error en realidad suele introducir otro.
      - Problema: supone que cualquier fallo descubierto tiene la misma incidencia en la fiabilidad
    - Escalón aleatorio
      - El incremento en fiablidad puede ser positivo o negativo en función de considear que se introducen errores en las reparaciones.
    - Métodos continuos
      - Necesarios para la predicción
      - Aproximación simplística, comparar los datos con un modelo conocido.
      - Se extrapola hasta la fiabilidad deseada.
    - Ejemplo de modelo continuo - el modelo logaritmico de Poisson
      - Tiene la forma de una asíntota en un eje de coordenadas de ROCOF y el tiempo.
      - Esta en función inversa a un medida de la intensidad de reducción de fallo por tiempo de prueba.
      - Esta en función logaritmica de los fallos en la primera prueba multiplicado por el tiempo.
      - Cuando se inicia el proceso de prueba se van tomando muestras para ajustar los parámetros de dicha función.
  - Beneficios de los modelos predictivos
    - Permiten la planificación de los test
    - Conocer cuando terminarlos
    - Permiten la negociación con el cliente

22. Validación de la safety: relación con la fiabilidad, chequeo RT, demostraciones lógicas, análisis simbólico, revisiones

- En estos casos se puede hablar más de garantía de la safety
- Común con la fiabilidad

- Se pueden emplear los métodos empleados para sistemas de muy alta fiabilidad.
- El problema es que para alcanzar los grandes niveles de seguridad se necesitan un número de test inviable.
- Los test solo proporcionan una evidencia más, que ha se ser complementada con revisiones y análisis estático.

➢ Diferencia con la fiabilidad
- La fiablidad acepta medidas cuantitativas
- La safety precisa de medidas cualitativas

➢ Chequeo en tiempo real
- Programación defensiva
- Incluir sentencia redundantes en el programa para comprobar que se cumplen los requisitos de safecty.
  - Envía excepciones cuando se encuentran condiciones de error

➢ Demostraciones lógicas
- Se puede realizar un análisis lógico de la corrección del programa
  - Esto es extremadamente costoso.
- En ciertas aplicaciones lo que si se puede hacer son pruebas formales que acrecientan la confianza en el software.
- Para hacer eso se siguen los siguientes pasos:
  - Se identifican los requisitos de safety del sistema.
  - Se asocian a estados no seguros en nuestro software
  - Se intenta probar por contradicción que no se puede alcanzar los estados inseguros.
  - Para ello se analiza el código
  - Se determina las condiciones necesarias para poder alcanzar el estado inseguro desde cualquier otro estado..
- Esta demostración no implica que el sistema funcione correctamente, sólamente que no alcanza un estado inseguro.

➢ Análisis simbólico
- Tema de técnicas

➢ Revisiones es un sistema safety
- Se consideran fundamentales en cinco etapas:
  - Son correctas las funciones deseadas
  - Estructura mantenible y comprensible
  - Verificar que el algoritmo y las estructuras de datos son consistentes con el comportamiento especificado.
  - Consistencia del codigo con los dos anteriores
  - Adecuación de los casos de test de sistema

23. Validación de la seguridad: Diferencia con los anteriores, Análisis de riesgos, Estrategias: Experiencia, herramientas, tiger, formal (objetivos, políticas, principios, técnicas y amenazas). Criterios de evaluación

➢ Similar al anterior en la medida en que son requisitos del tipo "no-debe"
- Pero es más difícil definir el comportamiento que debe seguir el sistema como simples restricciones que deben ser chequeadas en el sistema.
- La violación del sistema es muchas veces maliciosa.
- Evoluciona rápidamente en el tiempo (RSA cracked en el 99) con nuevos tipos de amenazas.

➢ Aproximaciones (complementarias)
- Validación basada en la experiencia
  - Se analiza contra tipos de ataques
- Validación soportada por herramientas
  - Chequeadores de passwords
- Tiger Teams
  - Se crea un equipo dentro del desarrollo destinado a eso
- Verificación formal
  - Se chequea contra modelos formales de seguridad BLP

➢ Criterios de Evaluación
- PKI->Evaluación y acreditaciónç

24. Métodos formales: Panorama controversia, niveles, desventajas, desconsejo, Especificación formal, cleanroom

➢ Qué son
- Requisitos->EspecificaciónFormal

➢ Panorama
- Requisitos->EspecificaciónFormal

- Especificación formal
- Cleanroom