UNIDAD 4: MODELOS DE CALIDAD DE SOFTWARE Y ASEGURAMIENTO DE LA CALIDAD

4.1 - Introducción a la calidad de software

1. Quién hace la calidad en un proyecto? Por qué es importante?

La calidad es responsabilidad de todos aquellos involucrados en el proceso de desarrollo de Software. Es importante por 2 razones: lo que se hace bien no tiene que hacerse dos veces (reduce costos y tiempo), y si se elaboran planes y gestiones de la calidad, puede repetirse y aplicarse en otros proyectos.

2. Qué es y que significa calidad?

La calidad del software es “el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario”.

La calidad se puede aplicar tanto al producto de software como al proceso de desarrollo de software.

3. Defina calidad desde el punto de vista del cliente y del producto. Por lo cual, cómo se logra la calidad?

Para el cliente, calidad significa que el producto debe adecuarse al usuario y satisfacerlo. Para el producto, la calidad se basa en cumplir las características y funciones que se han acordado.

4. Cuando se construye la calidad? De qué depende?

La calidad se construye a lo largo de todo el proyecto de desarrollo. Depende de tener procesos de desarrollo de software probados, maduros, formales, y estandarizados.

5. Como se define la Calidad de un producto? (atributos)

La calidad de un producto puede definirse en función de 4 atributos que debe cumplir:
- Cumplir satisfactoriamente la función que tiene asignada
- No presentar fallas o deficiencias
- Satisfacer al cliente que lo adquiere o utiliza
- Cumplir con las especificaciones técnicas establecidas

6. Cuál es la fórmula de la satisfacción del usuario? Ver recuadro en PPT

Satisfacción del usuario = producto que funciona + buena calidad + entrega dentro del plazo y presupuesto.

7. Qué es el aseguramiento de la calidad del software ACS? En qué se enfoca?

El Aseguramiento de la Calidad del Software (ACS) es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza adecuada en que el producto satisfará los requisitos dados de calidad.

Se enfoca en identificar y evaluar los defectos que puedan afectar al software.

8. Nombre las tres actividades principales de ACS?

Aseguramiento de la calidad: establecer un marco de trabajo que conduzca a la obtención de SW de alta calidad.

Planificación de la calidad: seleccionar procedimientos y estándares adecuados a partir del marco de trabajo, y adaptarlos para un proyecto de SW específico.

Control de la calidad: definir y aplicar procesos que aseguren que los procedimientos y estándares son seguidos por el equipo de desarrollo.

9. Cuáles son los tipos de estándares de ACS? (cuadro del producto y del proceso). Mencione un par de ejemplos de cada uno.

Los estándares de ACS son del **producto** (aplicados al producto a desarrollar), o del **proceso** (definen los procesos a seguir durante el desarrollo).

Ejemplos de estándares de **producto**: estructura del documento de requerimientos, formato del encabezado del procedimiento, estilo de programación en Java, formato del plan del proyecto, etc.

Ejemplos de estándares de **proceso**: sometimiento de documentos a revisiones, proceso de entrega de las versiones, proceso de aprobación del plan del proyecto, proceso de control del cambio, etc.

10. En que consiste IDEAL?

IDEAL/ciclo de Deming es un modelo elaborado por el SEI que provee un enfoque disciplinado de ingeniería para mejorar el proceso de software. Consiste en cuatro fases: **Planear -> Hacer -> Verificar -> Actuar (-> Planear)**. (la diferencia entre hacer y actuar, es que actuar se basa en analizar lo hecho y aplicar acciones correctivas, mientras que hacer significa ir adelante con lo planeado).

4.2-Modelo de Madurez de la capacidad

11. Qué beneficios otorga la mejora de un proceso?

Los beneficios de la mejora de un proceso son:

- Predictibilidad de la planificación y el presupuesto
- Tiempo de desarrollo
- Productividad
- Calidad (medida en n° de defectos)
- Satisfacción del Cliente
- Satisfacción de los empleados

12. Cómo se controla la variación para obtener un producto de alta calidad?

Se debe controlar la variación en los procesos, recursos y atributos de calidad del producto final (es decir, que no haya una gran diferencia). Esto significa reducir la diferencia entre:

- Recursos planificados vs reales
- Errores encontrados y ocultos
- Velocidad y precisión en las respuestas de soporte a clientes

13. Que significa que un equipo de proyecto tenga calidad?

Un equipo de proyecto tiene calidad cuando se capacitan y logran experiencia, trabajan en equipo, y cumplen compromisos.

14. Qué permite un sistema de certificación de calidad? Cuáles son sus pilares básicos?

Un sistema de certificación de calidad permite una valoración independiente que debe demostrar que la organización es capaz de desarrollar productos y servicios de calidad.

Sus pilares básicos son: metodología adecuada, medio de valoración de la metodología, y que los mismos estén reconocidos ampliamente por la industria.

15. Cuáles son los principales síntomas a considerar en la realidad? (braimstorming)

Los principales síntomas son:
- Compromisos incumplidos
- Visibilidad inadecuada de la gestión
- Problemas de calidad
- Baja estima

16. Cuáles son los beneficios de la Mejora del Proceso?

(ver pregunta 11 xd)

Los beneficios de la mejora de un proceso son:

- Predictibilidad de la planificación y el presupuesto
- Tiempo de desarrollo
- Productividad
- Calidad (medida en n° de defectos)
- Satisfacción del Cliente
- Satisfacción de los empleados

17. Defina que es la Complejidad Ciclomática?

La complejidad ciclomática es una métrica del software en ingeniería del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Básicamente, más complejidad ciclomática, más complejo el código, más riesgo si se lo cambia, y más pruebas son necesarias para asegurar su calidad.

18. Nombre las 5 actividades de QA? (ver grafica)

- Desarrollar estándares y procesos
- Administrar los cambios
- Probar (xd)
- Inspección del Sw
- Controlar la calidad

19. Indique en qué se centra un QA comparado con un Tester o QC. Analice diferencias

Un QA se centra en el proceso de desarrollo del producto, todo el proceso. Testing y control de calidad (QC) solo se centra en el producto en sí mismo.

20. Cuáles son las acciones básicas para lograr la calidad?

Tenemos 3 tipos de acciones:
- Acciones preventivas: pensadas para prevenir la ausencia de calidad (normas, estándares, métodos, etc)
- Acciones detectivas: pensadas para detectar tempranamente la falta de calidad (inspecciones, revisiones, etc)
- Acciones correctivas: pensadas para descubrir y corregir la falta de calidad (testing)

21. Qué hace/es CMMI®? Qué indica? A qué se orienta?

CMMI es un modelo de referencia para la calidad en los procesos de desarrollo y mantenimiento de SW. Establece una serie de buenas prácticas que las empresas deben cumplir para ser consideradas “maduras”. Indica **qué** hacer, no **cómo** hacerlo.

22. Identifique las diferencias entre CMMI y PMBoK

PMBoK es un compendio de mejores prácticas, es esencialmente una guía de lo que normalmente un gerente de proyectos debe llevar acabo. No se supone que se la siga al pie de la letra.

Tanto CMMI como PMBoK son un compendio de buenas prácticas, pero PM se enfoca solamente en la gestión de proyectos, mientras que CMMI se centra en el desarrollo y mantenimiento de SW, y también engloba a otras disciplinas.

Además, PMBoK certifica personas, mientras que CMMI certifica y califica empresas.

23. Como se identifica una empresa inmadura de una madura? Ver cuadro comparativo

En general, una empresa inmadura no tiene muchos recursos, no planifica los procesos o lo hace mal, acarrea constantemente deuda técnica, y depende de “Crunch” y de “héroes” para salir adelante. Una empresa madura tiene procesos y responsabilidades definidas, resultados predecibles, y sus empleados y clientes están satisfechos.

24. Nombre por lo menos 3 objetivos iniciales de CMMI®

- Eliminación de inconsistencias
- Reducir duplicaciones
- Incrementar la claridad y comprensión

25. Qué son los niveles de madurez? En qué se diferencia con los niveles de capacidad?

Los niveles de madurez permiten predecir el funcionamiento futuro de una organización dentro de una disciplina o un conjunto de disciplinas. Esencialmente, se usan para definir cómo se puede mejorar respecto de un grupo de área de procesos específicas.

Los niveles de capacidad definen el mejoramiento de una empresa en cada área de proceso en particular. Básicamente, capacidad = área específica, madurez = conjunto de áreas.

26. Nombre los cinco niveles de madurez de CMMI® ordenados, a que se dirigen y a qué nivel de gestión apuntan (Saber gráfico)

1. Inicial: Impredecible y reactivo. El trabajo se completa pero se retrasa y supera el presupuesto.
2. Repetible: Los proyectos se planifican, realizan, evalúan y controlan. El éxito comienza a ser repetible.
3. Definido: Proactivo, en vez de reactivo. Estándares transversales a toda la organización para guiar los proyectos, programas, etc.
4. Gestionado: Medido y controlado. El éxito y rendimiento se cuantifica y se mide, permitiendo realizar ajustes predecibles y fundamentados con datos.
5. Optimizado: Estable y flexible. La organización se mejora constantemente y se adapta al cambio y a las oportunidades.

27. Qué son las áreas de proceso en CMMI®?

Es un conjunto de prácticas relacionadas con una zona que, cuando se implementan en conjunto, satisfacen un conjunto de objetivos. En cada área hay metas y prácticas específicas.

28. Qué diferencia hay entre prácticas específicas y genéricas?

Las específicas implementan el proceso, mientras que las genéricas institucionalizan el proceso.

29. Nombre las dos representaciones del modelo CMMI y en qué áreas de proceso están organizada cada representación?

Las dos representaciones son **continua** y **por etapas**.
La representación por etapas o escalonada engloba a todas las áreas de proceso de la organización y pretende que todas mejoren a la vez, mientras que en la continua, la organización elige qué áreas de procesos va a mejorar en qué momento, permitiendo avanzar a un ritmo definido por la misma organización y priorizar ciertas áreas.

30. Qué es institucionalizar un proceso y que beneficios trae institucionalizarlo? Nombre tres

Institucionalizar un proceso implica que el mismo se vuelva parte esencial de la forma en la que se ejecuta un trabajo, y existe un compromiso para ejecutarlo. Esto lleva a tener resultados consistentes en el tiempo.

3 beneficios podrían ser:
- Facilita la rotación de personas entre proyectos
- Medición del impacto en los cambios de requisitos
- Establece un plan que se puede seguir incluso en situación de crisis

31. Qué es SCAMPI? En qué consiste? Qué proporciona?

SCAMPI (Standard CMMI Appraisal Method for Process Improvement) define el método para identificar fortalezas, debilidades, y clasificación de la empresa respecto del modelo de referencia.

Proporciona un método de evaluación en el contexto de mejora interna de procesos, selección de proveedores, y seguimiento de procesos.

4.3 -Aseguramiento de la calidad del Software

32. Cuál es la diferencia entre control de calidad y aseguramiento de la calidad?

El control de calidad se centra en medir y verificar el producto o entregable, mientras que el aseguramiento de calidad se centra en los procesos que fueron utilizados para crear el entregable.

33. Qué se tiene que hacer para tener ACS?

1. Definir lo que es calidad del SW
2. Crear las actividades para garantizar la calidad
3. Desarrollar el control de calidad (y actividades que implementen este control de calidad)
4. Usar métricas para mejorar el proceso de SW (y también, implícitamente, el producto final)

34. Nombre 5 de los a10 elementos de aseguramiento de la calidad del software

1. Estándares: se imponen o se adoptan
2. Revisiones y auditorías
3. Pruebas
4. Administración del cambio
5. Administración de riesgos

35. Nombre las 4 metas que se buscan alcanzar al realizar las tareas del ACS

- Calidad de los requerimientos
- Calidad del diseño
- Calidad del código
- Eficacia del control de calidad

36. Qué es la confiabilidad del software? Como se mide?

La confiabilidad del software es la “probabilidad de que un programa realice su objetivo satisfactoriamente (sin fallos) en un determinado periodo de tiempo y en un entorno concreto”.

Se la puede medir utilizando datos históricos, como el tiempo de fallas, cantidad de fallas, tiempo de funcionamiento correcto, etc.

37. Qué proporciona un Plan de ACS? Cómo funciona?

Un Plan ACS proporciona un mapa para instituir el ACS, es decir, funciona como una plantilla para las actividades de ACS que se instituyen para cada proyecto de SW.

UNIDAD 5: PRUEBAS Y ESTRATEGIAS DE PRUEBAS DE SOFTWARE

5.1-Pruebas de Software

1. Completar: Las pruebas de software son un elemento crítico para…

… la garantía de calidad del SW y representa una revisión final de las especificaciones, del diseño y de la codificación.

2. Defina pruebas/testing

Las pruebas de SW son una parte del proceso de ACS.
Prueba/test: actividad en la cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificadas, los resultados se observan y registran y se realiza una evaluación de algún aspecto. Es una actividad cognitiva que involucra varias funciones mentales.
“Proceso de ejecutar un programa con el fin de encontrar errores”.

3. Cuáles son los tres objetivos de testear?

- Encontrar defectos
- Evitar defectos
- Ganar confianza acerca del nivel de calidad y proporcionar información

4. Definiciones: depuración, equivocación, cobertura, refactorizar y condición de prueba.

Depuración: proceso de localizar, analizar y corregir los defectos que se sospecha que contiene el software

Equivocación: una acción del se humano que produce un resultado incorrecto

Cobertura: una medida de qué tan completo fue el testing

Refactorizar: modificación de la estructura interna de un programa con la finalidad de obtener mejor legibilidad y escalabilidad, sin modificar el comportamiento observable del SW

Condición de prueba: elemento o evento de un componente o sistema que debería ser verificado por uno o más casos de pueba

5. Explique la relación entre error, defecto, fallo y bug, e identifique “cuando se detectan”

Un error (o equivocación) es una acción humana que produce un resultado incorrecto, se comete en la fase de programación.

Este error a la hora de programar causa un defecto en el componente, que puede causar que el mismo falle en sus funciones. El defecto está en el código, es “estático”.

Cuando se ejecuta ese componente, el mismo puede **fallar** debido a su defecto. El fallo es la manifestación visible de un defecto, y aparece cuando el software se está ejecutando (ha sido desplegado).

Un bug es una inconsistencia del SW encontrada en la etapa de pruebas. Cuando se detecta un defecto o fallo, el mismo se “clasifica” como bug.

6. Indique cómo se clasifican las incidencias (error, defecto o fallo) y en función de qué

Las incidencias se clasifican en función de su impacto sobre la funcionalidad. Por ejemplo: bug, usability, performance, security issue, maintenance, etc.

7. Desde el punto de vista de Piattini, Calvo-Manzano, Cervera y Fernández, las pruebas deben centrarse en cuáles objetivos? Qué se prueba?

2 objetivos de las pruebas:
- Probar si el software **no** hace lo que debe
- Probar si el software **hace lo que no debe**

(más fácil pensar en lo que queremos lograr, que se haga lo que se debe, y que no se haga lo que no se debe. Las pruebas buscan probar lo contrario de las dos afirmaciones anteriores)

Lo que se prueba:
- Componentes del sistema
- Documentación
- Datos
- Requerimientos

8. Compare: verificación y validación. (tener en cuenta: qué comprueba, entorno, que se comprueba, quién, y las preguntas que se hacen)

Verificación: evaluar si el sistema o uno de sus componentes satisface los requerimientos funcionales y no funcionales. Se realiza “contra” el entorno de desarrollo o del proyecto. Se ha construido el producto correctamente? Lo hace la computadora.

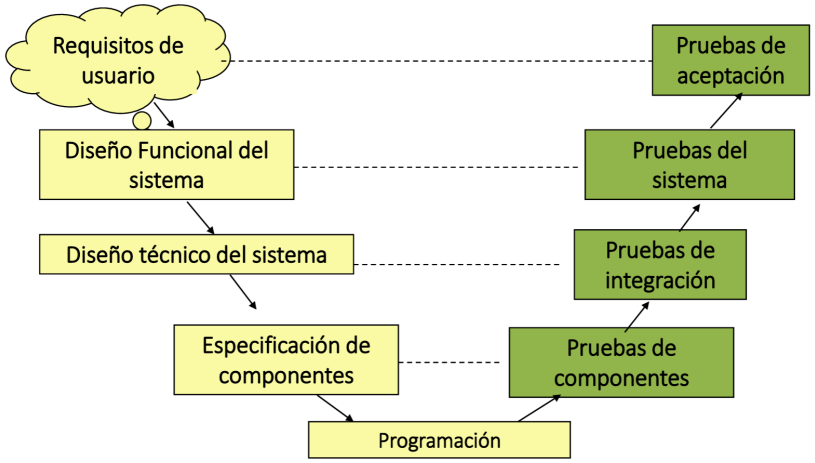
Validación: evaluar si el sistema o uno de sus componentes satisface los requisitos marcados por el usuario. Se realiza contra el entorno cliente (producción). Hemos construido el producto correcto? Lo hacen los humanos.

9. Qué condiciones se deben cumplir para que se produzca una falla?

Para que se produzca un fallo, debe cumplirse alcanzabilidad (el código debe llegar a ejecutar la línea que tiene el defecto), necesidad (la línea defectuosa debe producir un estado diferente del que produciría la línea correcta) y propagación (el estado incorrecto debe propagarse de modo que el resultado final sea distinto del esperado).

10. Relación entre productos de desarrollo y niveles de prueba: Modelo en V del ciclo de vida (gráfica). Y **ESTUDIAR LOS TIPOS DE TESTING DEL EXCEL.**

Hay una relación entre la rama izquierda (desarrollo) y la derecha (pruebas):

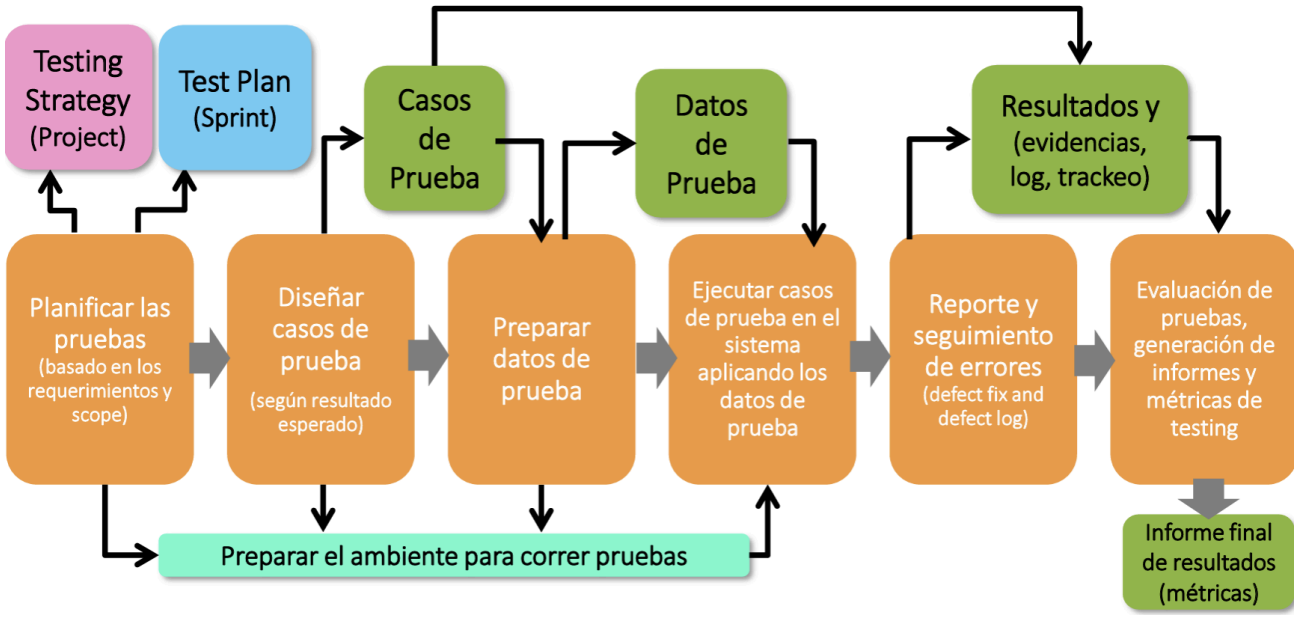


11. Cuando tenemos un buen caso de prueba según Myers? Cuándo tiene éxito?

    Un buen caso de prueba es aquel con alta probabilidad de mostrar un error no descubierto antes. Tiene éxito si lo descubre.

12. Complete el gráfico del proceso de pruebas y sus principales entregables (Se dará el gráfico

para completar lo faltante).



13. Mencione los principales entregables del proceso de pruebas y su definición.

- Test Strategy: guía orientada al proyecto para lograr el objetivo de la prueba y la ejecución de los tipos de prueba mencionados en el Test Plan
- Test Plan: orientada al sprint, se define el enfoque de la prueba y el alcance del proyecto
- Casos de prueba: items específicos que serán probados, con los pasos detallados que serán seguidos para verificarlo. Especificación de las entradas a la prueba y de la salida esperada.
- Datos de prueba: entradas seleccionadas para probar el sistema.
- Resultados (evidencias, log, trackeo)
- Informe final de resultados (métricas)

14. Nombre las diferencias entre un Test Plan & Test Estrategy

- Test Plan: orientado al sprint, se describen los métodos a utilizar para la prueba y habla de la cobertura de las pruebas. **Es el destino.**
- Test Strategy: orientado al proyecto, habla del enfoque, objetivo, entorno y herramientas de las pruebas. También habla de las estrategias de testeo, de automatización y el análisis de riesgos. **Es el mapa.**

15. Nombre qué se debe revisar frecuentemente al probar

- Objetivos
- Equipo
- Shift left testing (trazabilidad entre testing, desarrollo y requerimientos)
- Shift right testing (info de Producción que obtenemos para mejorar las pruebas)
- Estrategia de Testing Funcional (sin dividir entre manual y automatizado)
- Pruebas no funcionales (revisar atributos de calidad)
- Entornos y plataformas
- Calidad de código
- CI/CD

16. En que consiste el modelo Holístico de pruebas y en qué ayuda?

Consiste en realizar pruebas a lo largo del CVDS e implica un enfoque en los riesgos. Ayuda a definir “dónde se debe aplicar qué tipo de testing”

- Pruebas continuas
- Shift-left y shift-right

- Automatización del ciclo de entrega
- Foco en UX
- Calidad en todo el ciclo de desarrollo
- Monitoreo y analítica

17. Defina en qué consiste un Test cases y los datos de prueba. Cuál es su estructura o contenido de un caso de prueba ver cuadro anterior o de definiciones. **NOTA: APLICACIÓN PRACTICA PARA EL GLOBAL INTEGRADOR**. Ver material del TP: Gestión de Casos y datos de prueba para desarrollar casos de pruebas + y -, según un caso.

Los casos de prueba son especificaciones de las entradas a la prueba y de las salidas esperadas del sistema, más una declaración de lo que prueba. Los datos de prueba son las entradas seleccionadas para probar el sistema.

El contenido de un caso de prueba es:

- Precondiciones
- Conjunto de valores de entrada
- Conjunto de resultados esperados
- Poscondiciones esperadas
- Identificador único
- Dependencia de otros casos de prueba
- Referencia al requisito
- Forma en la cual se debe ejecutar y verificar los resultados (opcional)
- Prioridad (opcional)

18. En qué se basa el diseño de pruebas?

    Se basa en técnicas: particiones de equivalencia, valores límites, combinaciones por pares, tablas de decisión o máquinas de estado.

19. Fórmula para armar un caso de pruebas

    Entradas + precondiciones + resultado esperado = salidas + postcondiciones + evidencias

20. Nombre los pasos para llevar a cabo un caso de prueba

Pasos para llevar a cabo un caso de prueba:

1. Entender el requerimiento y los criterios de aceptación
2. Definir escenarios (camino feliz, flujos y subflujos alternativos, etc.)
3. Identificar condiciones de entrada
4. Definir clases de equivalencia
5. Realizar casos de prueba

21. Cuál es la diferencia entre plan y procedimiento de pruebas

El plan de pruebas establece el tipo de pruebas que se llevarán a cabo, explicitando el alcance, enfoque, recursos, responsables, etc. Por su parte, el procedimiento de pruebas define los casos de prueba específicos que se llevarán a cabo.

22. Nombre cómo se pueden generar los Datos para los Test Cases

Los datos se pueden generar mediante 4 estrategias:

- Por Diseño: datos inventados, dirigidos a probar una funcionalidad en concreto, basados en la experiencia en el negocio o hipótesis del futuro
- Datos reales: se toman datos que realmente se hayan registrado
- Simulados: creados a partir de un algoritmo
- Object Mother: cada entidad del sistema tendrá un mecanismo para generar objetos de prueba. Para entidades compuestas por varias entidades, los mecanismos se activan recursivamente (ej: entidad de prueba Cliente que genera nombre y apellido por default, y le pide a país que genere una entidad de prueba País, y así)

23. Cuadro de escenario, casos y script de prueba: definición, de qué deriva y el nivel de detalle

El **escenario de prueba** es una secuencia de pasos que un usuario o el sistema pueden realizar para alcanzar un objetivo específico. Describe el objetivo general sin entrar en detalles de cada paso. Se deriva de la **especificación de los requerimientos**.

El **caso de prueba** es el conjunto de condiciones de entrada/salida, acciones y resultados esperados que deben ser documentados para validar y verificar una característica del sistema. Tiene un nivel de detalle alto, especificando cada paso y su resultado esperado. Se deriva del **escenario de prueba**.

El **script de prueba** es un conjunto de instrucciones en lenguaje entendible por computadora, que automatiza la ejecución de un caso de prueba (o un conjunto de ellos). Es extremadamente detallado, tiene instrucciones precisas para la automatización. Se deriva del **caso de prueba**.

24. Cuando usar escenarios?

Los escenarios son útiles cuando se presenta alguna o varias de las siguientes características:
- Las reglas de negocio son complejas
- El sistema cambia de un momento a otro
- No se pueden reutilizar los test cases
- La cantidad de combinaciones para probar la aplicación o funcionalidad son muchas

25. Entornos: ¿Qué permiten cada uno de ellos?

- Desarrollo: permite el modelado y diseño de aplicaciones, el análisis estático y dinámico del código, el análisis de rendimiento de la aplicación y el mejoramiento del código de acceso a la base de datos.
- Pruebas: permite simular situaciones complejas, como una migración de servidores, Base de Datos bajo estrés, implementación de clustering, etc.
- Producción: permite la monitorización constante de la aplicación, la identificación de cuellos de botella, y el mantenimiento y evolución de la aplicación.

26. Nombre de qué depende el esfuerzo de las pruebas?

El esfuerzo de una prueba puede depender de las características del producto (tamaño, complejidad, calidad de la especificación, etc.), las características del proceso de desarrollo (herramientas, estabilidad de la organización, presión del tiempo, procesos de test, etc.), y/o los resultados de las pruebas (cantidad de defectos, cantidad de trabajo que se debe rehacer).

27. Nombre las características y las habilidades de un buen tester

Características de un buen tester:
- Creativo
- Observador
- Pensamiento crítico
- Curioso
- Pensamiento Lateral (Out of the Box)

Habilidades de un buen tester:
- Conocimiento técnico (herramientas, codificación, tecnologías Web y Cloud, técnicas de pruebas)
- Conocimiento del negocio (comunicación con los usuarios y stakeholders)
- Habilidades personales / blandas (escucha activa, pensamiento analítico / crítico, comunicación, etc.)

28. Diferencias entre: analista QA y QC (arme un cuadro)

	QA (Quality Assurance)	QC (Quality Control)
Interacción con los defectos	Proactivo, intenta prevenir los defectos	Reactivo, busca encontrar los defectos que ya se produjeron
Enfoque	Centrado en el proceso de	Centrado en el producto

	desarrollo	desarrollado
Momento	Todo el proceso, desde el principio de la planificación hasta el producto final	Entra en juego una vez ya se ha desarrollado (parcial o totalmente) el producto

29. Diferencias entre testing estático y dinámico (cuando se ejecuta y si tiene que ver con verificación/validación)

El testing estático se realiza sin ejecutar el código, y verifica el código, diseño y los requerimientos, por lo que se lo considera una **verificación.**

El testing dinámico se realiza ejecutando el código, y analiza el comportamiento del sistema y el resultado que produce. Es un tipo de **validación**. Se puede realizar testing dinámico funcional y no funcional.

30. Enfoque de pruebas de caja blanca, negra y gris. **ESTUDIAR CUADRO DEL EXCEL solapa completa**

Pruebas de Caja negra: su enfoque está en las entradas y salidas del sistema, considera su funcionamiento externo sin preocuparse de la implementación interna. Está basado en requisitos y especificaciones, priorizando la experiencia de usuario y la funcionalidad externa. Útil para pruebas de UI, integración de sistemas y UAT.

Requiere conocimientos de los requisitos funcionales y no funcionales, conocimiento del negocio (flujos, procesos, resultados, reglas, etc.) y técnicas de diseño de casos de prueba.

Pruebas de Caja blanca: su enfoque está en la lógica interna. Se basa en analizar el código fuente, considerando su estructura y seguridad. Es útil para realizar optimizaciones, corregir errores lógicos, de control de flujo, de inicialización, y verificar la integridad y robustez del sistema.

Requiere conocimientos técnicos de lenguajes de programación, estructuras de datos, algoritmos, paradigmas, y en general ser capaz de entender la lógica del código para crear casos de prueba apropiados.

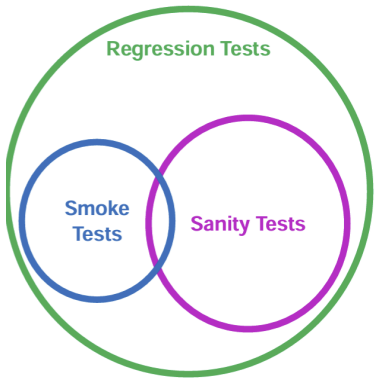
Se subdivide en 3 tipos:

- Análisis de flujo de control: casos de prueba que incluyen todas las rutas posibles de control de flujo del programa (todas las ramificaciones, bucles y condiciones).
- Pruebas de cobertura de código: usa herramientas que permiten medir qué porcentaje del código fuente ha sido ejecutado durante las pruebas, intentando cubrir lo más posible.
- Pruebas de caja de cristal: examina las funciones internas a través de sus entradas y salidas, para verificar que cada componente funcione correctamente.

Pruebas de Caja gris: un punto medio, pone el enfoque en la funcionalidad externa pero tiene cierto conocimiento de la lógica interna. Se basa en una combinación del código fuente y las especificaciones y requisitos asociados al mismo. Es útil cuando se tiene información limitada del funcionamiento interno, o cuando se quiere complementar a las pruebas de caja negra y blanca. Permite encontrar una amplia gama de errores, incluyendo funcionales, de diseño, o de implementación.

Requiere una combinación de conocimientos de requisitos, negocio y arquitectura global, como también requiere conocimientos técnicos de programación, algoritmos, estructuras de datos, etc.

31. Tipos de Testing: FUNCIONAL, NO FUNCIONAL y las que pueden ser funcionales o No funcionales.
    CUADRO DEL EXCEL y PPTs. Estudiar cuales son, tipos y descripción:
    a. Funcionales: que buscan (qué hace el sistema), descripción y enfoque que validan
       Buscan comprobar que el software haga exactamente lo esperado, validando la funcionalidad, los requisitos y los CU, con un enfoque en una característica específica, en qué hace el software y si hace lo que se supone.
        1. Pruebas E2E y sus subtipos
            a. Definición y diferencias
               Verifican el flujo completo de una transacción.
               Regresión: los cambios no afectaron otras funcionalidades existentes
               Smoke: sobre las funcionalidades críticas para verificar estabilidad básica
               Sanity: busca errores obvios introducidos por un cambio
            b. PPT: Ver gráfica (círculos) de la ppt de E2E dónde muestra Smoke, Sanity y Regresión



            c. PPT: Ver cuadro comparativo de Smoke, Sanity y regresión: las diferencias y aspectos comunes entre las tres.
               Hay diferencias principalmente en lo que se quiere probar, quién lo prueba, a partir de qué, y cuándo.
        2. Pruebas Exploratorias y Ad-Hoc: definición y diferencias
           Exploratorias: planificadas, resultado del estudio y aplicación del sistema en el contexto adecuado, basadas en la experiencia del tester.
           Ad-Hoc: sin planificación, para encontrar defectos rápidamente.
        3. Las ya vistas en pregunta del modelo en V
           Del sistema: comportamiento del sistema completo en un entorno simulado
           De integración: correcta integración entre diferentes componentes o sistemas
           De componentes / unitarias: correcta funcionalidad de unidades individuales de código
    b. No Funcionales: definición (cómo lo hace el sistema)
       Evalúan los atributos de calidad de un software que no están directamente relacionados con funcionalidades específicas, centrándose en cómo lo hace el sistema.
        1. Pruebas de Rendimiento y ejemplos de sus subtipos
           Evalúan la capacidad de un sistema para responder a la demanda de sus usuarios y mantener un nivel de servicio aceptable. Ejemplos:
            - Performance
            - Estrés
            - Carga
            - Volumen
            - Pico
            - Resistencia
            - Configuración

2. Pruebas de Seguridad: qué identifican y cuáles hay

   Identifican vulnerabilidades, verificando la confidencialidad, integridad y disponibilidad. La hay de:

   - Análisis dinámico
   - Análisis estático

c. Funcionales o No Funcionales

1. Pruebas de interfaz: GUI, API, UI y Cross browser

   GUI: centradas en la interacción del usuario con la interfaz visual, verifican la coherencia, la facilidad de navegación y la respuesta a las acciones del usuario.

   API: centradas en la interacción entre diferentes sistemas, verifican la funcionalidad de las API, la calidad de los datos intercambiados y la seguridad de las comunicaciones.

   UI/SI: centradas en la interacción entre la interfaz del usuario y el sistema backend, verifican que la interfaz presente la información correcta y responda de manera adecuada a las interacciones del usuario

   Cross browser: evalúa la compatibilidad con diferentes navegadores y dispositivos.

2. Pruebas de Aceptación: UAT, Alfa y Beta

   UAT: las realizan los usuarios finales para verificar que se cumple con sus necesidades y expectativas.

   Alfa: las realizan un equipo interno en un ambiente controlado, antes que las beta.

   Beta: las realizan un grupo de usuarios externos en un ambiente real para obtener feedback.

3. Pruebas de Mantenimiento: cuáles hay

   Ciclomáticas y mantenimiento: miden la complejidad lógica de un fragmento de código

   Mantenimiento correctivo: el defecto se corrigió sin introducir nuevos errores

   Mantenimiento adaptativo: el software se adapta a cambios en el entorno

   Mantenimiento perfectivo: las nuevas funcionalidades funcionan como se espera y no afectan a las existentes

   Mantenimiento preventivo: periódicamente, se buscan y corrigen problemas antes de que afecten a los usuarios.

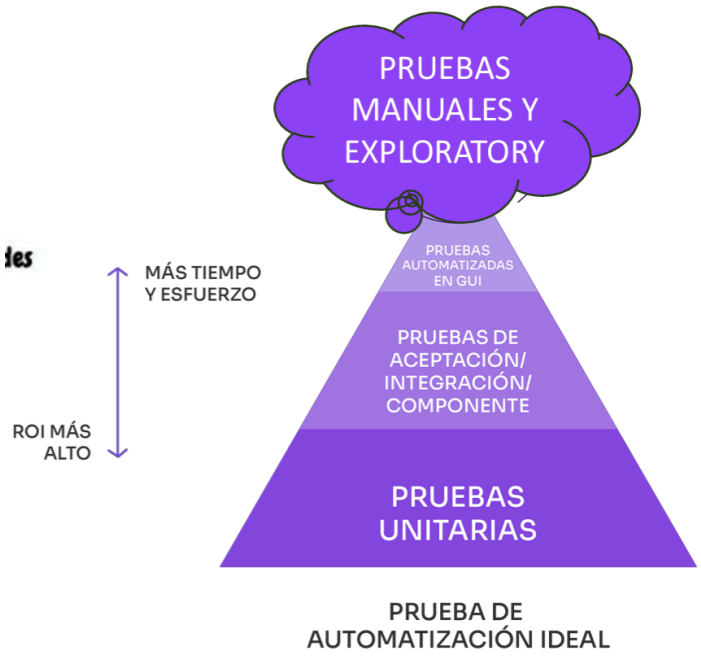
32. Diferencia entre pruebas funcionales vs E2E (cuadro de ppts)

Funcionales	E2E
Validan una sola pieza de código	Validan todo el sistema principal y los subsistemas conectados
Énfasis en verificar las características y cada funcionalidad del sistema	Énfasis en verificar el flujo del proceso de prueba
Asegura cumplir los criterios de aceptación	Garantiza la continuidad en funcionamiento tras cambios
Prueba cómo 1 usuario interactúa	Prueba cómo varios usuarios trabajan
Validan el resultado para entradas y salidas	Validan que cada caso se haya completado

33. Pirámide de automatización: qué es? ¿Cómo es la pirámide? (gráfica)

Organiza ciertos tipos de testing, colocando más cerca de la punta aquellas más complejas y de mayor

alcance, logrando un mejor ROI (Return Of Investment) con las pruebas en la base.



34. Automatización de pruebas: en que consiste, beneficios y riesgos.

Consiste en que una máquina logre ejecutar los casos de prueba en forma automática, leyendo la especificación del mismo de un script (lenguaje genérico o propio de la herramienta), una planilla de cálculo, un modelo, etc.

Algunos riesgos de automatizar:

- Sobredependencia de la herramienta de automatización
- Expectativas irreales para la herramienta
- Subestimación del tiempo, coste y esfuerzo necesarios para aprender la herramienta, y para mantener los recursos de pruebas generados por la herramienta

35. Nombre 4 beneficios de la automatización

- Se reducen los trabajos repetitivos
- Mayor consistencia y capacidad de repetición
- Valoración de objetivos, cobertura y comportamiento mediante métricas
- Fácil acceso a la información sobre los tests o pruebas (ej: tasa de éxito y/o fallo)

36. Cuándo automatizar?

La clave está en automatizar cuando el **costo de automatizar será menor que el costo de ejecutar las pruebas manualmente**. Para esto, tenemos que tener en cuenta la calidad del test caso generado, su vida útil (cuantas veces se ejecutará), el tiempo necesario para programar el script, para ejecutar las pruebas, para analizar los resultados, para mantener los test cases, etc.

37. Qué es un falso positivo y un falso negativo en el resultado de una prueba automatizada?

Cuando hablamos de pruebas automatizadas, un falso positivo ocurre cuando la prueba dice que hay un error pero en realidad no lo hay. Esto causa una pérdida de tiempo y recursos al buscar el error en el código fuente cuando en realidad está en la prueba.

Un falso negativo es el caso contrario: la prueba indica que no hay errores pero en realidad sí puede haberlos, el test no está cubriendo o probando la funcionalidad correctamente.

38. Describa el proceso de depuración. Cuál es la diferencia entre depurar y testear?

La depuración es la actividad de desarrollo que identifica la causa de un defecto, permitiendo reparar el código y verificando que el defecto ha sido corregido correctamente. La prueba solo muestra fallos causados por los defectos, el debugging o depuración permite encontrar la causa de la falla y reparar ese defecto.

Usualmente, se detecta la falla mediante una prueba y a raíz de esto se inicia un proceso de depuración para encontrar y eliminar el error.

39. Qué es ISTQB?

Es una norma internacional que certifica la calidad de los profesionales de testing de alto nivel. Es una de las únicas certificaciones a nivel personal que tiene cierta importante en el nivel de calidad de software.