	Testing	Auditorias
Concepto	Actividad destructiva mediante el cual se somete a un software o componente a condiciones específicas con el objetivo de encontrar **defectos**, cuya presencia es asumida de antemano en el Sw.	Evaluación independiente de los productos o procesos de software para asegurar el cumplimiento de estándares, lineamientos, especificaciones y procedimientos incluyendo documentación que especifique: 1. La forma o contenido de los productos a ser desarrollados 2. El proceso por el cual los productos son desarrollados 3. Cómo debería medirse el cumplimiento con estándares o lineamientos.

	Testing	Auditorias
Caracteristicas	1. Se lleva entre el 30% y el 50% del costo total de los proyectos de SW 2. Su artefacto de salida mas importante son los casos de prueba 3. Es exitoso cuando se encuentran defectos 4. No garantiza que el producto no tiene errores. 5. Siempre es necesario	1. Dan visibilidad a la gerencia sobre los procesos de trabajo 2. Se clasifican en auditoría de Configuración **Física** (compara el código con la documentación de soporte), de Configuración **Funcional**(compara el software construido con los requerimientos de la ERS) y de **proyecto** (busca **verificar** la consistencia del producto a medida que evoluciona en el proceso de desarrollo). 3. Permite identificar áreas de mejora y áreas de insatisfacción potencial del cliente

	Testing	Auditorias
Diferencias	1. No garantizan la calidad del software 2. Sus resultados son un conjunto de casos de prueba junto con errores y defectos a ser corregidos 3. Se realiza por miembros del equipo y es parte integral del proceso de desarrollo de software.	1. Aseguran la calidad del software 2. Sus resultados son observaciones sobre condiciones que deberían mejorarse pero no requieren un plan de acción, Desviaciones que requieren un plan de acción y buenas practicas. 3. Se realiza por un agente externo al equipo que no participo en el desarrollo y se realizan en intervalos específicos.

Similitudes	1. Ambas buscan identificar problemas. El Testing se centra en **defectos** en el código y las Auditorías en **desviaciones/observaciones** de los estándares	1. Aunque con diferente alcance, ambos buscan verificar el cumplimiento de ciertos requisitos 2. Implican esfuerzo y costo para los proyectos

Planificación de pruebas para el software- Niveles y tipos de pruebas para el software.

¿Qué son los casos de prueba?

Son datos necesarios de seteo para determinar si se cumplen uno o mas criterios de aceptación, se basa en las US, los casos de uso y los requerimientos

¿Cuánto consume la fase de implementación (Desarrollo) del sw?

Entre el 33% y 35% del costo del proyecto

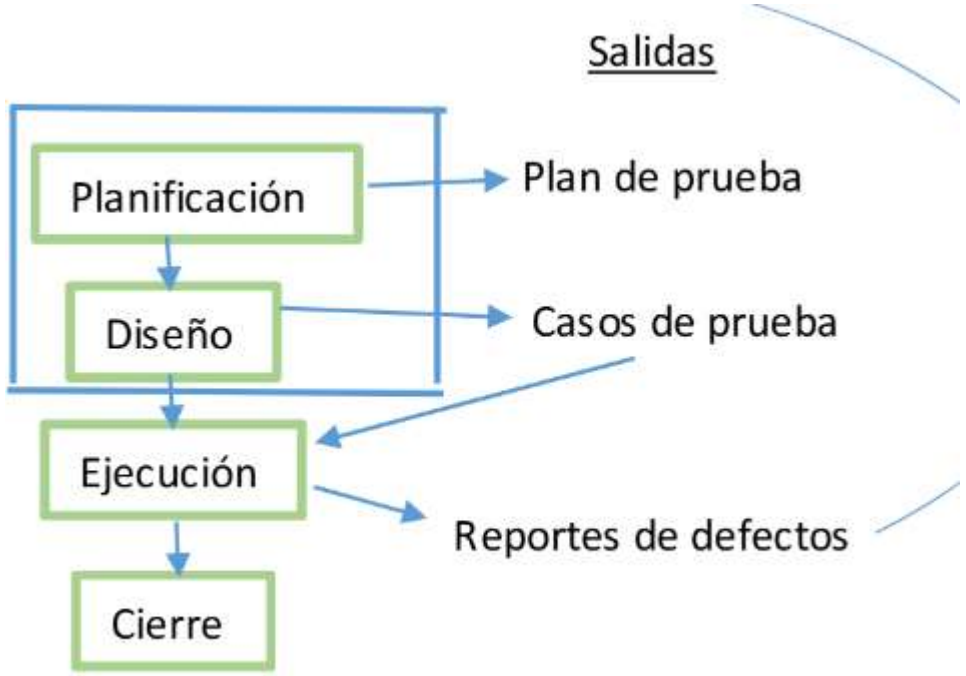
¿De que pruebas se encarga el P.O y de que pruebas se encarga el equipo de desarrollo?

- El P.O de las **UAT (Pruebas de adaptación de usuario)** y el equipo de desarrollo de las demas

¿Cuál es el objetivo de TDD?

Adelantar la etapa de planificación y la etapa de diseño del testing para antes de la fase de implementación del proceso de desarrollo y así mejorar la calidad del producto

¿Cuáles son las salidas en cada una de las etapas del testing?



¿Cuál es la diferencia entre validación y verificación?

- **Validar:** Consiste en comprobar que el SW satisface las necesidades de los usuarios, es lo mas caro de corregir

- **Verificar:** Consiste en comprobar que el software cumple las **especificaciones** y se hizo según lo planificado

¿Cuál es la diferencia entre error y defecto?

La diferencia es el momento en el cual se detectan y solucionan.

- Un error es detectado y corregido en una misma etapa mientras que un defecto es un error que se traslada de una etapa a otra etapa posterior.

- El testing encuentra defectos, ya que son errores que surgieron en etapas anteriores, pero se detectan en la etapa de prueba.

- Ambos conceptos pueden o no generar fallas en el sistema. Por ejemplo, un error en los colores de una interfaz no es una falla, ya que no conllevan a un mal funcionamiento del sistema.

¿Qué es la severidad de un defecto, la prioridad y como se clasifica?

La severidad refleja el impacto que genera el defecto en el sistema y no el costo de corregirlo. Mientras que la prioridad la define el cliente (Alta media baja)

→ **Bloqueante o invalidante:** No se puede utilizar el sistema

→ **Grave:**

→ **Leve:**

→ **Cosmética:** Relacionada con aspectos de interfaz de usuario

¿Cuáles son los ambientes de SW?

→ **Desarrollo**

→ **Prueba**

→ **Pre producción:** Se incluyen las UAT

→ **Producción**

¿Qué son los ciclos de prueba del testing?

- Es la ejecución de un conjunto de casos de prueba en una versión determinada del producto.
- Generalmente se tienen 2 ciclos, el primero es conocido como ciclo 0 y siempre es manual, es donde se configura todo inicialmente y a partir del ciclo 1 ya se pueden automatizar las pruebas.
- En caso de detectar defectos, el producto vuelve a desarrollo para su corrección, se debe evitar grandes cantidades de ciclos, ya que generan retrabajo.

¿Qué estrategias de ciclos de prueba existen?

1. **Con regresión:** Se ejecutan exhaustivamente todos los casos de prueba incluyendo nuevos y anteriores para asegurar que las modificaciones no afecten a funcionalidades que ya fueron corregidas.

2. **Sin regresión:** Se enfoca únicamente en probar las nuevas funcionalidades y no ejecutar las pruebas anteriores del sistema

¿Cuánto testing es suficiente?

El testing exhaustivo es imposible por la cantidad de tiempo que requiere. El momento en que se deja de hacer testing depende del nivel de riesgo o costo del proyecto.

Los riesgos permiten definir prioridades de que se debe testear primero y con qué esfuerzo.

El criterio de aceptación para decidir si una determinada fase de testing ha finalizado puede ser definido en términos de:

1. Costos.

2. % de tests corridos sin fallas.

3. Inexistencia de defectos de una determinada severidad.

4. Pasa exitosamente el conjunto de pruebas

5. **Good Enough:** Cantidad aceptable de fallas no críticas

6. Defectos detectados es similar a la cantidad de defectos estimados.

¿Cuáles son los ambientes de testing?

Son todos los recursos de Hw y Sw que se requieren para poder trabajar con el producto. Existen distintos ambientes según la etapa de desarrollo en la que se encuentre el software y las necesidades de la misma.

1. **Desarrollo:** Implica hardware y software necesario para desarrollar y desplegar el producto. En este ambiente se realizan las pruebas unitarias.

2. **Pruebas** Es el ambiente que utilizan los testers para llevar a cabo las pruebas. En este se realizan las pruebas de sistema y de integración.

3. **Preproducción** Similar al ambiente de producción, Sirve para poder comprobar que el producto funcionará una vez desplegado en producción de manera correcta. En este ambiente se realizan las pruebas de aceptación.

4. **Producción:** Es la configuración de software y hardware que tienen los usuarios finales, y que utilizan para sus actividades. En este entorno no se realizan pruebas

Tipos de pruebas

- **Pruebas unitarias (Nivel 1):** Las hace el desarrollador y se realizan sobre un componente, de manera independiente a los otros. Se suelen reparar los **errores** sin registrarlos formalmente.

- **Pruebas de integración (Nivel 2):** Buscan verificar el correcto funcionamiento de dos o más componentes que se relacionan entre si a través de sus interfaces. Se realizan durante el testing

- **Pruebas de sistema o de versión (Nivel 3): :** Se realiza durante el testing sobre un versión de un producto o incremento de SW. Se busca de emular de la mejor manera posible el entorno en donde se usará el SW.

- **Pruebas de aceptación de usuario (Nivel 4): :** Se realizan en el despliegue y debería realizarlas el usuario para validar lo que el mismo requirió. Busca que el usuario/cliente se familiarice con el producto.

Filosofia LEAN – Metodo Kanban

¿Qué es LEAN?

Es una filosofía que busca maximizar el valor generado al cliente con el mínimo uso de recursos, argumentando que todo el esfuerzo que no cree valor, se considera desperdicio.

Principios LEAN

1. **Eliminar desperdicios: Se** busca detectar y eliminar todo lo que no aporta valor al cliente
   - Producir funcionalidades que luego no se usan
   - Retrabajo.
   - Producir Artefactos que se vuelven obsoletos antes de terminaros.

2. Amplificar aprendizaje: Se basa en poder aprender a partir del trabajo y que ese conocimiento sea accesible a toda la organización

3. Embeber la integridad conceptual: Mantener la coherencia y calidad del producto desde el principio. Encastrar todas las partes del producto, que tenga coherencia y consistencia, pensando en una estructura que sea mantenible.

4. **Diferir compromisos:** Tomar decisiones importantes lo mas tarde posible, para contar con la información necesaria

Principios LEAN

6. Ver el todo: Analizar el sistema completo para optimizar el flujo de valor de extremo a extremo. Tener una visión holística de conjunto (producto, valor agregado y el servicio del producto)

7. Entregar lo antes posible: Liberar versiones tempranas del producto para obtener retroalimentación rápida por parte del cliente.

¿Cuáles son los tipos de desperdicio?

Tipo 1: Desperdicio necesario: Actividad que no agrega valor directamente al cliente, pero es necesaria en el proceso actual ya que son tareas que no se pueden evitar, aunque se busca minimizarlas lo máximo posible.
Ejemplo: actividades de supervisión y de control.

Tipo 2: Puro desperdicio: Es una actividad que no agrega valor y tampoco es necesaria. Se busca eliminarlas por completo.

¿Cuáles son los desperdicios según LEAN?

1. **Defectos:** Son errores que se trasladan de una etapa a otra posterior en la cual se introdujo, provocando retrabajo, debido a que Testing debe "devolver" la funcionalidad a desarrollo para la resolución.

2. **Talento no utilizado:** No aprovechar las ideas, conocimiento o capacidades el equipo de trabajo.

3. **Esperas:** Tiempo muerto que una persona tiene que pasar para poder realizar su trabajo, puede ser causado pñor aprobaciones, dependencias o falta de insumos

4. **Movimientos:** Cambios constantes de contexto o de tareas que dispersan la atención de los equipos

5. **Transporte:** Transferencia de información entre equipos o herramientas que generan demoras o errores.

6. **Procesos extra:** Pasos innecesarios que no agregan valor al producto, se busca identificarlos y eliminarlos.

7. **Sobreproducción:** Desarrollar mas funcionalidades de las que el cliente necesita o

¿Qué es Kanban?

Es un método de trabajo para estionar y optimizar el flujo de trabajo como servicios profesionales o actividades en las que interviene la creatividad y el diseño.

Principios de Kanban

1. **Gestión de cambios:** Kanban promueve la evolución gradual en lugar de cambios drásticos. El proceso se mejora de forma continua, limitando el trabajo en curso y ajustando el flujo según la capacidad del equipo.

2. **Entrega de servicios:** Se centra en cumplir compromisos de forma predecible y continua, gestionando el flujo de trabajo como un servicio que agrega valor al cliente. Se busca optimizar el tiempo de entrega y mantener la calidad.

Practicas de Kanban y principios LEAN ASOCIADOS

1. **Limitar el trabajo en curso:** Evitar que haya demasiados tareas a medio hacer o en progreso para evitar cuellos de botellas permitiendo **eliminar desperdicios** ya que se evita la sobrecarga, multitarea y espera.

2. Hacer políticas explicitas: Definir claramente las reglas como DoD, DoR, limite de wip, o criterios de pull. Esto permite **embeber la integridad conceptual** y **dar poder al equipo**.

3. Visualizar el trabajo: Usar el tablero para visualizar el flujo de trabajo y los riesgos. Permite **eliminar desperdicios** identificando fácilmente los cuellos de botella y **ver el todo** (sistema completo).

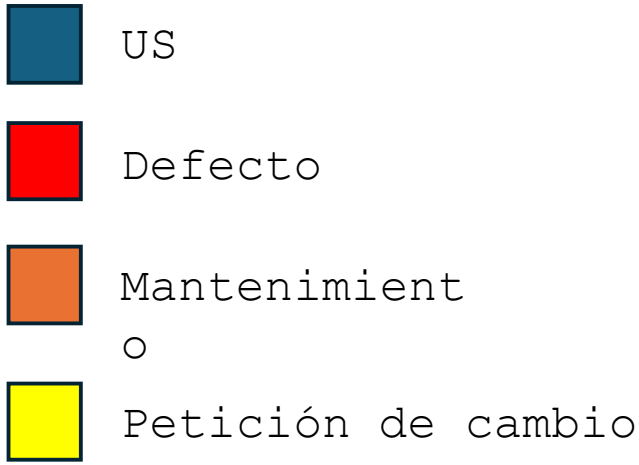
4. Establecer ciclos de retroalimentación: Fomentando la mejora y el conocimiento, permite **amplificar el aprendizaje**

5. Mejorar colaborativamente, evolucionar experimentalmente: Ejecutar experimentos para avanzar o aprender mediante el uso de el método científico. Esto permite **Amplificar el aprendizaje**

5. Gestionar el flujo: Observar cuán rápido se mueven las tareas y eliminar cuellos de botellas. De esta manera **eliminamos los desperdicios**

Diseñe el tablero Kanban

✓ El equipo:
1 Analista Funcional
1 Arquitecto
2 Desarrolladores
2 Analistas de Prueba
1 Responsable de Despliegue



✓ Acuerdos:
- El cliente valida funcionalidades antes de producción
- Revisión técnica al código es parte del "criterio de hecho"

Cola de tareas	Analisis		Diseño		Desarrollo		Revisión tecnica		Listo para build	Testing		Evaluación con el cliente		Despliegue		En producción
	En progreso	Hecho	En progreso	Hecho	En progreso	Hecho	prog	hecho		En progreso	Hecho	En progreso	Listo	En pro	listo	

Defina los tipos de trabajo que utilizará, justificando la respuesta y al menos 5 políticas de calidad

1. **User stories: Se centran en el usuario, son independientes, testeables, estimables, etc.**

2. **Defectos:** Asegura que el **trabajo no planificado** se vuelva visible, permitiendo analizar el impacto de los mismos en el flujo de trabajo.

3. **Mantenimiento:** Este tipo de trabajo previene fallos o ralentizaciones que pueden resultar muy costosas en el futuro

4. **Peticiones de cambio:** Generalmente son **externas** por lo que tienen valor directo para el cliente. Su visibilidad permite priorizarlas adecuadamente

Ejemplos de Políticas de calidad

1. **Política:** Las tarjetas solo pueden ser añadidas por el Product Owner o líder de equipo, para asegurar relevancia.

2. **Política:** No se incorporan nuevas tareas al tablero sin descripción o criterios de aceptación.

3. **Política:** Se realiza cada día una TEAM Kanban Meeting de 15 minutos máximo frente al tablero para observar el flujo de trabajo e identificar bloqueos o cuellos de botella y decidir como proceder.

4. **Política:** La fase de testing de un producto de trabajo termina al pasar el 80% de tests sin fallas y no hayar defectos críticos, bloqueantes o graves.

5. **Política:** Hay un limite para el WIP, cada columna cuenta con un límite máximo de tareas, y si se supera, el equipo debe priorizar la liberación de los bloqueos.

Metricas en Agile-Kanban-Tradicional

A. Metricas Agiles

¿Qué son las métricas?

Las métricas son resultados no procesos, se debe medir lo necesario y nada mas

Principios agiles que guían la elección de métricas

- Nuestra mayor prioridad es satisfacer al cliente por medio de entregas tempranas y continuas de software valioso
- El Software funcionando es la principal medida de progreso

A. Metricas Agiles

1. Velocidad: Es la cantidad de story points que el P.O acepta en el sprint Durante la review, permite corregir errores de estimación

2. Capacidad: Se determina durante el sprint Planning. En un equipo maduro se puede medir en story points, caso contrario se mide en horas ideales.

- **Horas de Trabajo Disponibles por día** (WH) X **Días Disponibles Iteración** (DA) = Capacidad

WH x DA = Capacidad

3. RTF: Cantidad de features testeadas que están funcionando, es decir, cuantas piezas de producto se terminaron y están en ejecución.

B. Metricas de Kanban

1. **Lead Time (vista del cliente):** Tiempo que pasa desde que una tarea entra al sistema hasta que se entrega al usuario

2. **Cycle Time (vista interna):** Tiempo desde que se empieza a trabajar efectivamente en una tarea hasta que se termina

3. **Touch Time:** Tiempo en el que las tareas están "En progreso"

4. **Eficiencia del ciclo de proceso:** Se calcula como Touch time / Lead time. Si el resultado es cercano a 1, significa que la mayor parte del tiempo la tarea estuvo siendo trabajada activamente. Si el valor es bajo, una pequeña parte del tiempo se trabajó realmente la tarea.

C . Metricas Tradicionales

- **Métricas de proceso:** Permite saber en términos de la organización como estamos trabajando, independientemente de un proyecto en especifico. Por ej: Porcentaje de proyectos terminados con éxito, porcentaje de proyectos cancelados.

- **Métricas de proyecto:** Permiten saber si un proyecto de SW en ejecución esta cumpliendo o no de acuerdo a lo planificado. Se consolidan para crear métricas de proceso. Ej: Variación del calendario real vs planificado

- **Métricas de producto:** Tienen una relación directa con el producto de software que estamos construyendo. Ej: Tamaño del Producto, cantidad de errores encontrados

Metricas → Pregunta de parcial

Suponiendo un desarrollo ágil, analice el siguiente gráfico sobre el comportamiento del proyecto manifestado y explique detalladamente la interpretación que hace luego del análisis

El burndown Chart

Dado el siguiente gráfico explique la razón por la cual todo lo que está fuera de coincidencia es considerado desperdicio

- **Calidad programada:** Los alcances de producto planificados

- **Calidad realizada:** Lo que realmente se desarrolló del producto

- **Calidad necesaria:** Mínimas características que el producto debe tener, para satisfacer los requerimientos

Todo lo que está **fuera de la zona central** se considera **desperdicio** porque representa un **recurso, esfuerzo o tiempo** que no se tradujo en el valor real que el cliente esperaba, o no se ajustó a lo planificado o necesario